Design and Optimization of Heterogeneous Networks

BY

SHANYU ZHOU
B.Eng. University of Electronic Science and Technology of China

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Chicago, 2019

Chicago, Illinois

Defense Committee:

- Hulya Seferoglu, Chair and Advisor
- Rashid Ansari
- Emir Halepovic, AT&T Labs Research
- Balajee Vamanan, Department of Computer Science
- Besma Smida

ACKNOWLEDGMENTS

My graduate study is a *marriage* to research.

I would like to deeply thank my *Marriage Officiant*, Ph.D. advisor, Prof. Hulya Seferoglu, for introducing me to research, offering me tremendous help during our journey, and guiding me to enjoy every sweetness and bitterness with research.

I would like to thank my *Marriage Witnesses*, my Ph.D. defense committee members, Prof. Rashid Ansari, Dr. Emir Halepovic, Prof. Balajee Vamanan and Prof. Besma Smida, for providing their valuable time and comments.

I would also like to thank my *Groom's Party*, my labmates and friends, Yuxuan, Haoyu, Brian, Haiyang, Yang, Chang, Raheleh, Yasaman, Elahe, Pengzhen, Maaz and Usama, for their valuable companion along this journey.

I would also like to extend my respect and gratitude to my *Marriage Organizers*, the staff and faculty of the Department of Electrical and Computer Engineering at University of Illinois at Chicago, who offered seamless help during our journey.

Last but not least, I would like to thank my parents for their tremendous support and sacrifice.

Yes I do. I am grateful to *marry* research. Our journey never ends and I want it to be simple and happy.

SZ

CONTRIBUTION OF AUTHORS

A version of Chapter 2 has been published in ACM CarSys Workshop (Zhou, S. et al., 2016) and IEEE Allerton Conference (Zhou, S. et al., 2015). I was the primary author and major driver of research. Prof. Hulya Seferoglu helped me with the development of main ideas and improvement of algorithms.

A version of Chapter 3 has been published in IEEE ITA Workshop (Zhou, S. et al., 2016). I was the primary author and major driver of research. Prof. Hulya Seferoglu and Prof. Erdem Koyuncu helped me with development of main ideas and improvement of writings.

A version of Chapter 4 has been accepted in IEEE LANMAN Conference (Zhou, S. et al., 2019). I was the primary author and major driver of research. Prof. Hulya Seferoglu, Prof. Balajee Vamanan, Dr. Emir Halepovic and Dr. Vijay Gopalakrishnan helped me with development of ideas and improvement of simulations. Muhammad Usama Chaudhry helped me with ns-3 simulations.

Chapter 5 represents a series of my own unpublished work directed at optimal coded computation for distributed systems with hard deadlines. I anticipate that this line of work will be continued after I leave and will ultimately be published as part of a co-authored manuscript.

TABLE OF CONTENTS

CHAPTER **PAGE**

1 **INTRODUCTION** 1
1.1 Motivation 1
1.2 Thesis Contributions 5
1.2.1 Connectivity-Aware Traffic Phase Scheduling for Heterogeneously Connected Vehicles 6
1.2.2 Blocking Avoidance in Transportation Systems 7
1.2.3 Flow Control and Scheduling for Heterogeneous (Per-Flow and FIFO) Queues over Wireless Networks 8
1.2.4 Managing Background Traffic over Cellular Network Using *Sneaker* . 8
1.2.5 Managing Background Traffic over Cellular Network Using *Legilimens* 9
1.2.6 Optimal Coded Computation with Hard Deadlines 10
1.3 Thesis Organization 11

2 **CONTROL AND OPTIMIZATION OF HETEROGENEOUSLY CONNECTED VEHICLES** 12
2.1 Connectivity-Aware Traffic Phase Scheduling Algorithm 13
2.1.1 Background 13
2.1.2 Related Work 17
2.1.3 System Model 18
2.1.4 Connectivity-Aware Traffic Phase Scheduling 20
2.1.4.1 CAMW: Connectivity-Aware Max-Weight 20
2.1.4.2 Expectation 22
2.1.4.2.1 Calculation of $E(K_i^{\phi}(t))$ for `Queue I` 22
2.1.4.2.2 Calculation of $E(K_i^{\phi}(t))$ for `Queue II` 26
2.1.4.3 Learning 30
2.1.5 Performance Evaluation 31
2.1.5.1 The baseline: max-weight algorithm 31
2.1.5.2 Evaluation of CAMW for `Queue I` 32
2.1.5.3 Evaluation of CAMW for `Queue II` 33
2.1.6 Conclusion 36
2.2 Blocking Avoidance in Transportation Systems 37
2.2.1 Background 37
2.2.2 Related Work 40
2.2.3 System Model 42
2.2.3.1 `Queue I` 43
2.2.3.2 `Queue II` 43
2.2.4 Average Waiting Time Analysis 44

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

2.2.4.1 Average Waiting Time for `Queue I` 45
2.2.4.1.1 All Vehicles Have Communication Abilities 45
2.2.4.1.2 None of the Vehicles Have Communication Abilities 46
2.2.4.1.3 A Percentage of Vehicles Have Communication Abilities 48
2.2.4.2 Average Waiting Time for `Queue II` 50
2.2.4.2.1 All Vehicles Have Communication Abilities 50
2.2.4.2.2 None of the Vehicles Have Communication Abilities 51
2.2.4.2.3 A Percentage of Vehicles Have Communication Abilities 53
2.2.5 Shortest Delay Algorithm 55
2.2.6 Performance Evaluation 56
2.2.6.1 Evaluation of Average Waiting Time at Isolated Intersections 57
2.2.6.1.1 `Queue I` 57
2.2.6.1.2 `Queue II` 58
2.2.6.2 Evaluation of the Shortest Delay Algorithm 59
2.2.6.2.1 Baselines 59
2.2.6.2.2 `Queue I` 61
2.2.6.2.3 `Queue II` 62
2.2.7 Conclusion 63

3 FLOW CONTROL AND SCHEDULING FOR HETEROGENEOUS (PER-FLOW AND FIFO) QUEUES OVER WIRELESS NETWORKS 64
3.1 Background 64
3.2 Related Work 68
3.3 System Model 70
3.4 Support Region 72
3.4.1 Single-FIFO Queue 73
3.4.2 Arbitrary Number of Queues and Flows 77
3.4.3 A Convex Inner Bound on the Support Region: 86
3.5 Flow Control and Scheduling 88
3.6 Performance Evaluation 99
3.6.1 Baselines 99
3.6.2 Single-FIFO Queue 101
3.6.3 Two-FIFO Queues 102
3.7 Conclusion 105

4 MANAGING HETEROGENEOUS TRAFFIC FOR INTERNET OF THINGS OVER CELLULAR NETWORKS 106
4.1 *Sneaker*: Managing Background Traffic in Cellular Networks 107
4.1.1 Background 107
4.1.2 Related Work 109
4.1.3 System Model 111
4.1.4 Interaction of TCP with Cellular Networks 113

4.1.4.1 Design of *Sneaker* 115
4.1.4.2 NUM Formulation When Foreground and Background Flows Coexist 115
4.1.4.3 Optimal Dropping Rate 118
4.1.4.4 Practical Dropping Rate 119
4.1.5 Implementation of *Sneaker* 119
4.1.5.1 Design Parameters and Signalling for *Sneaker* 119
4.1.5.1.1 Scheduling Probability 119
4.1.5.1.2 Local Signalling 124
4.1.5.1.3 End-to-end Signalling 125
4.1.5.2 Implementation of *Sneaker* on ns-3 125
4.1.6 Evaluation of *Sneaker* 126
4.1.7 Conclusion 131
4.2 *Legilimens*: An Agile Transport for Background Traffic in Cellular Networks 132
4.2.1 Background 132
4.2.2 Related Work 135
4.2.3 Challenges 137
4.2.3.1 The Proportional Fair (PF) Scheduler 137
4.2.3.2 LEDBAT and TCP-LP in Cellular 139
4.2.4 *Legilimens* 141
4.2.4.1 High-level Overview 141
4.2.4.2 Estimating Capacity and Busyness 143
4.2.4.2.1 Intuition: Optimal Low Priority Data Rate 143
4.2.4.2.2 Practical Issues 146
4.2.4.2.3 Capacity and Busyness Estimation Algorithm 148
4.2.4.3 Congestion Control 150
4.2.4.3.1 Congestion Window ($cwnd$) Adaptation 150
4.2.4.3.2 Fairness 151
4.2.4.3.3 Summary and Discussion 152
4.2.4.4 Deployment Model 153
4.2.5 Evaluation Using a Real Implementation 153
4.2.5.1 Methodology 154
4.2.5.1.1 Real-network Test-bed 154
4.2.5.1.2 PhantomNet Test-bed 155
4.2.5.1.3 Workload and Relevant Metrics 155
4.2.5.1.4 Protocol Implementations 157
4.2.5.2 Results from the Real Network 157
4.2.5.2.1 Mixed Workload During Idle Hours 157
4.2.5.2.2 Mixed Workload During Busy Hours 161
4.2.5.2.3 One-on-one Workload While Stationary 163
4.2.5.2.4 One-on-one Workload While Moving 164
4.2.5.3 Results from PhantomNet 166

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

4.2.6 Evaluation Using Simulations 166
4.2.6.1 Methodology 166
4.2.6.2 Results from Simulation 167
4.2.6.2.1 Scale 167
4.2.6.2.2 Fairness 168
4.2.6.2.3 Sensitivity 169
4.2.6.2.4 Queuing Behavior 170
4.2.7 Summary and Discussion 171
4.2.8 Conclusion 173

5 OPTIMAL CODED COMPUTATIONS WITH HARD DEADLINES 174
5.1 Background 174
5.2 Related Work 178
5.3 System Model 180
5.4 Matrix-vector Multiplication with Hard Deadlines 181
5.4.1 Uncoded Computation 182
5.4.2 MDS Coded Computation 183
5.4.3 Optimal Coded Computation for Matrix-vector Multiplication and Its Approximation 184
5.5 Matrix-matrix Multiplication with Hard Deadlines 190
5.5.1 Uncoded Computation 191
5.5.2 One-dimensional MDS Coded Computation 192
5.5.3 Two-dimensional MDS Coded Computation 194
5.5.4 Optimal Coded Computation for Matrix-matrix Multiplication and Its Approximation 197
5.5.4.1 Optimal and Approximated k for One-dimensional Coded Computation 197
5.5.4.2 Optimal and Approximated M and k for Two-dimensional Coded Computation 198
5.6 Evaluation and Simulation 199
5.6.1 Matrix-vector Multiplication 199
5.6.1.1 Exponential Distribution 200
5.6.1.2 Shifted Exponential Distribution 203
5.6.2 Matrix-matrix Multiplication 207
5.6.2.1 Exponential Distribution 207
5.6.2.2 Shifted Exponential Distribution 210
5.7 Conclusion 215

6 CONCLUSION 216

APPENDIX 219

CITED LITERATURE 221

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

VITA 243

LIST OF TABLES

TABLE **PAGE**

I PARAMETER VALUES 152

II SUMMARY OF RESULTS (*L-PASSIVE*) 172

LIST OF FIGURES

FIGURE **PAGE**

1 An example intersection with four possible traffic phases. 14

2 An example single-lane intersection, where vehicles are going straight, turning left and turning right respectively. 15

3 Two queuing models considered in this chapter, where λ_1 and λ_2 are the arrival rates of straight-going and left-turning traffic, respectively.(a) Single-lane traffic model. (b) One+two lane model. 19

4 An illustrative example of communicating vehicles in a queue at a time slot. Communicating vehicles are at labeled locations; $v_1, v_2, \cdots, v_T$. 22

5 Four possible configurations (Conf. I to Conf IV) for the first three vehicles in `Queue II`, where L and S denote that the intention of the vehicle is to turn left or go straight, respectively, while E denotes that the location is empty (due to previous blocking). 27

6 The average queue size versus time for `Queue I` . Each green phase lasts for two time slots. The arrival rate is $\lambda_1 = 0.18$ and $\lambda_2 = 0.12$ to each of the queue in the intersection. 32

7 Intersection efficiency versus total arrival rate to each queue with different communication probability ρ for `Queue I` . Each green phase lasts for two time slots and each queue has the same arrival rate and $\lambda_1 = 1.5\lambda_2$. 34

8 The evolution of the average queue size of the intersection using our algorithm and max-weight algorithm for different communication probability ρ for `Queue II`. The arrival rate to each queue is $\lambda_1 = \lambda_2 = 0.2$ and each green phase lasts for two time slots. 35

9 Intersection efficiency versus total arrival rate to each queue with different communication probability ρ for `Queue II`. Each queue has the same total arrival rate and $\lambda_1 = \lambda_2$, and each green phase lasts for two time slots. 36

10 An example intersection with multiple vehicles with different routes. 38

LIST OF FIGURES (Continued)

FIGURE **PAGE**

11 Representation of the south-north bound queue in the intersection demonstrated in Figure 10 using (a) single-lane, (b) one+two-lane, and (c) two+three-lane queuing models. λ_L, λ_S, and λ_R are the arrival rates of vehicles to the queue with destinations on the left, straight, and right, respectively. 39

12 The queuing model with two vehicles in the head-of-line (HoL). The arrival rates of right-turning and straight-continuing vehicles are merged as λ_1 and the arrival rate of left-turning vehicle is λ_2. 44

13 Four possible configurations for the first three vehicles in `Queue II`. 51

14 Vehicle configurations expansion; Configuration II and Configuration III in Figure 13 for different traffic phases. (a) Configuration II: Left turn phase is ON. (b) Configuration II: Go straight phase is ON. (c) Configuration III: Left turn phase is ON. (d) Configuration III: Go straight phase is ON. 52

15 The Markov chain for states S_i $(i = 1, \dots, 6)$, where α_i $(i = 1, 2)$ denotes the probability that the HoL vehicle is going straight or left and p_i $(i = 1, 2)$ denotes the probability that the traffic light choose go-straight or turn-left phases. 54

16 `Queue I.` (a) Average waiting time for different total arrival rates with different communication probabilities p_t. (b) Average queue size versus communication probability p_t. The arrival rates are $\lambda_1 = \lambda_2 = \lambda_3 = 0.1$. 57

17 `Queue II.` (a) Average waiting time for different total arrival rates with different communication probabilities p_t. (b) Average queue size versus communication probability p_t. The arrival rates are $\lambda_1 = \lambda_2 = 0.15$. 58

18 An illustrative transportation network with four nodes. The total arrival rate to node 2 and node 3 is λ_{n2} and λ_{n3}, respectively. 60

19 `Queue I.` (a) Estimated traveling time vs communication probability p_t at node 2. (b) Estimated traveling time vs arrival rate at node 2. 61

20 `Queue II.` (a) Estimated traveling time vs communication probability p_t at node 2. (b) Estimated traveling time vs arrival rate at node 2. 63

21 Queueing structure of one-hop downlink topology with (a) per-flow queues, and (b) a FIFO queue. 66

LIST OF FIGURES (Continued)

FIGURE **PAGE**

22 The wireless network model that we consider in this thesis. N FIFO queues share a wireless medium, where the nth FIFO queue, $\mathcal{Q}_n$ carries $\mathcal{K}_n$ flows towards their respective receiver nodes. The arrival rate of the kth flow passing through the nth queue is $\lambda_{n,k}$. 71

23 (a) Single-FIFO queue; $\mathcal{Q}$ is shared by K flows. (b) Support region of a single-FIFO queue as well as per-flow queues with two flows. 73

24 Markov chain for the single-FIFO queue system shown in Figure 23(a). 75

25 The state transition diagram for the states $H_m = k$, $\forall k \in \mathcal{K}_m$ and for the mth queue. Note that this state transition diagram only shows a subset of state transitions for clarity. 82

26 (a) Two FIFO queues; $\mathcal{Q}_n$ and $\mathcal{Q}_m$ are shared by two and one flows, respectively. (b) Three dimensional support region with $\lambda_{n,1}$, $\lambda_{n,2}$ and $\lambda_{m,1}$ for the two-FIFO queues scenario shown in (a). 85

27 Single-FIFO queue shared by two flows when $p_1 = 0.1$, $\beta = 1$, and $U_k(\lambda_k) = \log(\lambda_k)$. (a) Per-flow rates vs. p_2. (b) Total flow rate vs. p_2. 102

28 Single-FIFO queue shared by multiple flows. p_k is selected randomly between $[0, 1]$, $\beta = 1$, and $U_k(\lambda_k) = \log(\lambda_k)$. (a) Average flow rate versus number of flows. (b) Percentage of throughput improvement of qFC over max-weight. . . 103

29 Two FIFO queues with four flows. (a) Total flow rate versus β when $p_{n,1} = 0.1$, $p_{n,1} = 0.5$, $p_{m,1} = 0.1$, $p_{m,2} = 0.5$, and log utility is employed, *i.e.*, $U_{n,k}(\lambda_{n,k}) = \log(\lambda_{n,k})$. (b) Total rate versus $p_{n,2} = p_{m,2}$ when $p_{n,1} = p_{m,1} = 0.1$ and $\beta = 2$. 104

30 Per-flow rates versus $p_{m,2}$ for the scenario of two-FIFO queues with four flows when $p_{n,1} = p_{n,2} = p_{m,1} = 0$ and $\beta = 2$. (a) dFC and qFC. (b) Max-weight. 105

31 The service architecture in LTE network 111

32 Example cellular system setup with *Sneaker* 112

33 LTE protocol stack 126

34 Comparison of different approaches to prioritization 127

35 Foreground flows benefit from *Sneaker* 128

LIST OF FIGURES (Continued)

FIGURE **PAGE**

36 Fairness among background and foreground flows 129

37 Large-scale results at moderate load (50%) 130

38 Large-scale results at low load (20%) 131

39 Cellular vs. wired or Wi-Fi networks 138

40 Behavior of existing protocols 140

41 High-level overview 142

42 Scheduler and timestamps 145

43 Scheduler with batching 147

44 *Legilimens* congestion control 150

45 *Legilimens* congestion window evolution 154

46 Typical U_{PRB} and workload impact 158

47 FCT for short flows in the real network 158

48 Throughput in the real network 160

49 Cell load and utilization during busy hours 162

50 One-on-one workload in real network 163

51 Throughput of *L-passive* (background) vs. CUBIC (foreground) while moving . 164

52 Throughput in PhantomNet 165

53 Throughput in a simulated network 168

54 Fairness analysis among *Legilimens* flows 169

55 Queuing behavior of protocols 171

LIST OF FIGURES (Continued)

FIGURE **PAGE**

56 An canonical example of a distributed computing system consisting of one master device and four helper devices. There is a hard deadline T set for the whole task and master device applies (a) uncoded computation, (b) coded computation using (4,3) MDS codes and (c) coded computation using (4,2) MDS codes. 176

57 Illustration of one-dimensional MDS coded computation for matrix-matrix multiplication A^TB. 193

58 Illustration of two-dimensional MDS coded computation for matrix-matrix multiplication A^TB. 196

59 Probability of success versus different deadlines for the case of uncoded and MDS coded computation. The processing time on each helper follows exponential distribution with mean $\frac{1}{\lambda} = 1$. 201

60 Probability of success versus k in MDS coded computation. The processing time follows exponential distribution with mean $\frac{1}{\lambda} = 1$. The deadline is $T = 1$. 201

61 Optimal and approximated k for coded computation. The processing time follows exponential distribution with $\lambda = 1$. 202

62 Probability of master device meeting the deadline using coded computation with optimal k^*, approximated $\hat{k}$ and uncoded schemes. The processing time follows exponential distribution with $\lambda = 1$. 203

63 Probability of success versus different deadlines for the case of uncoded and coded computation. The processing time follows shifted exponential distribution with $\lambda = 1$ and $s = 2$. 204

64 Probability of success versus k in coded computation. The processing time follows shifted exponential distribution with $\lambda = 1$ and $s = 2$. The deadline is $T = 5$. 205

65 Optimal and approximated k for coded computation. The processing time follows shifted exponential distribution with $\lambda = 1$ and $s = 2$. 206

66 Probability of success using coded computation with optimal k^*, approximated $\hat{k}$ and uncoded schemes. The processing time follows shifted exponential distribution with $\lambda = 1$ and $s = 2$. 207

LIST OF FIGURES (Continued)

FIGURE **PAGE**

67 The probability of success using one-dimensional MDS, two-dimensional MDS and uncoded computation schemes. In two-dimensional MDS coded computation, the number of groups is $M = 11$. The processing time follows exponential distribution with $\lambda = 1$. 209

68 The probability of success using one-dimensional, two-dimensional and uncoded computation schemes. In both one-dimensional and two-dimensional MDS coded computation, $k = 2$. The processing time follows exponential distribution with $\lambda = 1$. 210

69 The approximated solution of k and M using one-dimensional MDS coding and two-dimensional MDS coding, and optimal solution of two-dimensional MDS coding. 211

70 Probability of success using approximated solution to one-dimensional MDS coding and two-dimensional MDS coding, and optimal solution to two-dimensional MDS coding. The processing time follows exponential distribution with $\lambda = 1$. 212

71 The probability of success using one-dimensional MDS, two-dimensional MDS and uncoded computation schemes. In two-dimensional MDS coded computation, the number of groups is $M = 11$. The processing time follows shifted exponential distribution with $\lambda = 1$ and $s = 2$. 213

72 The probability of success using one-dimensional, two-dimensional and uncoded computation schemes. In both one-dimensional and two-dimensional MDS coded computation, $k = 5$. The processing time follows shifted exponential distribution with $\lambda = 1$ and $s = 2$. 214

73 The approximated solution of k and M using one-dimensional MDS coding and two-dimensional MDS coding, and optimal solution of two-dimensional MDS coding. 214

74 Probability of success using approximated solution to one-dimensional MDS coding and two-dimensional MDS coding, and optimal solution to two-dimensional MDS coding. The processing time follows shifted exponential distribution with $\lambda = 1$ and $s = 2$. 215

LIST OF ABBREVIATIONS

CAMW	Connectivity Aware Max Weight
CPS	Cyber-Physical System
D2D	Device to Device
dFC	deterministic FIFO Control
FIFO	Fist In First Out
FCFS	Fist Come First Service
HoL	Head of Line
IoT	Internet of Thing
LPT	Low Priority Transport
M2M	Machine to Machine
MDS	Maximum Distance Separable
OWD	One-Way Delay
PK	Pollaczek Khinchine
qFC	Queue-based FIFO Control
RED	Random Early Detection
RTT	Round Trip Time

SUMMARY

The motto of *Internetting everything, everywhere, all the time* is becoming reality thanks to the increasing number and diversity of devices with Internet connectivity such as smart phones, tablets, wearable devices, and connected vehicles. The booming of devices with diverse applications has a significant social and economic impact. Yet, existing resource allocation and network coding mechanisms do not address the full range of challenges, specifically heterogeneity; these newly emerging devices and applications are highly heterogeneous and dynamic in nature. The goal of this thesis is to develop new networking and coding mechanisms that make better use of available resources by taking into account the heterogeneity.

The first chapter of this thesis focuses on transportation systems of heterogeneously connected vehicles. Due to technical equipment constraints, security and privacy concerns, and lossy communication channel qualities, vehicles may not have connectivity all the time. In this context, it is crucial to take into account this heterogeneous connectivity while developing network control mechanisms. In this chapter, we (i) focus on an isolated intersection and develop the connectivity-aware traffic phase scheduling algorithm for heterogeneously connected vehicles that increases the average number of vehicles passing the intersection and (ii) focus on the transportation system and develop shortest routing algorithm with minimum traveling delay for heterogeneously connected vehicles.

The second chapter investigates the performance of wireless networks of devices with heterogeneous (per-flow and FIFO) queues. In particular, we consider a scenario where there are arbitrary number of heterogeneous queues (per-flow and FIFO queues) shared by arbitrary number of flows. These queues share the same transmission medium such that only one queue can transmit data at a time. In this setup, we formulate the support region, which is characterized by the set of arrival rates that can be stably supported in the network. In general, the support region of this system is non-convex, which makes it difficult to obtain the optimal operating point for the system. Therefore, we further develop a convex inner-bound on the support region, which can be proved to be tight in certain cases. With this convex inner bound, we are able to develop a centralized resource allocation scheme; dFC. Based on the structure of dFC, we develop a stochastic flow control and scheduling algorithm; qFC. We approve that qFC converges to the optimal operating point in the convex inner bound.

The third chapter focuses on managing heterogeneous traffic generated by Internet of Things (IoT) devices over cellular networks. Nowadays, a large variety of traffic, time-sensitive “foreground” traffic (*e.g.,* web browsing) and time-insensitive “background” traffic (*e.g.,* software updates), compete for the scarce cellular bandwidth, especially on the downlink. While there is limited in-network support for traffic prioritization, existing end-to-end, “low priority transport protocols” exhibit sub-optimal performance in cellular networks. In this chapter, we develop management schemes for heterogeneous priority traffic to fully utilize available resources. In particular, we propose (i) *Sneaker*, which yields to time-sensitive foreground traffic during periods of congestion and enables time insensitive background traffic to efficiently utilize any spare capacity, and (ii) *Legilimens*, which is an agile TCP variant for cellular downlink transfers, and is able to deliver traffic using only the *spare* capacity on the downlink.

The last part of this thesis focuses on optimal coded computation for distributed computing network with heterogeneous helper devices. In this chapter, we focus on (n, k) MDS coded computation for matrix-matrix multiplication with hard deadline. We analyze the probability of master device meeting the deadline and characterize the optimal k that maximizes this probability by taking into account heterogeneity of helper devices. Obtaining the optimal k requires integer programming which could be time consuming. To speed up the calculation of k, we further develop approximated solution k that has much reduced time complexity and close optimal performance.

CHAPTER 1

INTRODUCTION

The contents of this chapters are based on our work that is published in [1–4].©2016 IEEE. Reprinted, with permission, from [3]. ©2015 IEEE. Reprinted, with permission, from [2]. ©2019 IEEE. Reprinted, with permission, from [4].

1.1 Motivation

Recent years have witnessed the dramatic growth of connectivity in heterogeneous networks such as connected vehicles, cellphones and computers. This trend has posed tremendous challenges for current networks with limited resources and advocated new network control mechanisms to address a number of critical issues. Specifically, the heterogeneity in these emerging devices that arises with their dynamic nature has to be addressed in order to better understand current networks. In this thesis, we particularly focus on transportation systems and wireless networks and the goal is to develop new networking mechanisms that make better use of available resources by taking into account the heterogeneity.

In transportation systems, the rapidly increasing number of vehicles in metropolitan transportation systems, has introduced several challenges including higher traffic congestion, delay, accidents, energy consumption, and air pollution. For example, the average of yearly delay per auto commuter due to congestion was 38 hours, and it was as high as 60 hours in large metropolitan areas in 2011 [5]. The congestion caused 2.9 billion gallons of wasted fuel in 2011, and this figure keeps increasing yearly [5], *e.g.,* the increase was 3.8% in Illinois between years 2011 and 2012 [6]. This trend poses a challenge for efficient transportation systems, so new traffic management mechanisms are needed to address the ever increasing transportation challenges and eliminate the inefficiency.

A straightforward approach to address the congestion problem is to enhance the capacity of transportation systems, which requires significant investment. On the other hand, it is extremely important to understand the capacity of existing as well as future transportation systems so that (i) available resources are effectively and fully utilized, and (ii) new transportation systems are developed based on the actual need. Capacity characterization of transportation systems and utilizing available capacity are getting increasing interest recently [7], [8], [9]. This is thanks to connected vehicle, which makes utilization of available capacity possible with the communication and coordination abilities of vehicles. In general, the connection among vehicle are enabled by the Internet connection via cellular network [7], [8], [9] or device-to-device (D2D) connection such as Bluetooth and WiFi-Direct [10]. In this context, to possibly take advantage of connected vehicles, two critical tasks, which are the focus of this thesis, have to be addressed. First, it is crucial to understand how heterogeneous communication affects the performance of transportation systems. And second, it is crucial to take into account practical constraints that arise from real transportation systems while characterizing capacity to fully utilize underlying resources in transportation systems.

In wireless data networks, the recent growth in mobile and media-rich applications continuously increases the demand for wireless bandwidth, and puts a strain on wireless networks [11], [12]. This dramatic increase in demand poses a challenge for current wireless networks, and calls for new network control mechanisms that make better use of scarce wireless resources. Furthermore, most existing, especially low-cost, wireless devices have a relatively rigid architecture with limited processing power and energy storage capacities that are not compatible with the needs of existing theoretical network control algorithms. One important problem is that low-cost wireless interface cards are built using First-In, First-Out (FIFO) queueing structure, which is not compatible with the per-flow queueing requirements of the optimal network control schemes such as backpressure routing and sheduling [13]. Per-flow and FIFO queues coexist in current wireless network, and our focus on this problem is to investigates the performance of wireless networks of devices with heterogeneous (per-flow and FIFO) queues.

The backpressure routing and scheduling paradigm has emerged from the pioneering work [13], [14], which showed that, in wireless networks where nodes route and schedule packets based on queue backlogs, one can stabilize the queues for any feasible traffic. It has also been shown that backpressure can be combined with flow control to provide utility-optimal operation [15]. Yet, backpressure routing and scheduling require each node in the network to construct per-flow queues. When a FIFO queue is used instead of per-flow queues, the well-known head-of-line (HoL) blocking phenomenon occurs. Although HoL blocking in FIFO queues is a well-known problem, achievable throughput with FIFO queues in a wireless network is generally not known. In particular, the network support region, which is characterized by a set of feasible arrival rates that can be stably supported (*i.e.,* not overflowing buffers), as well as the resource allocation schemes to achieve optimal operating point in the support region are still open problems.

In cellular networks, Internet of Things (IoT) has emerged as a new paradigm in which a large number of heterogeneous devices such as smart phones, wireless sensors, smart meters, health monitoring devices, etc., remain connected to the Internet. As the staggering growth of IoT devices continues, it is estimated that we will have billions of IoT devices in the next five years [16], [17]. Such an exponential growth of IoT devices will have a significant impact on cellular networks over which a vast number of such devices with diverse throughput, latency, and signaling requirements will communicate. Therefore, we need smarter network mechanisms to manage the heterogeneous traffic demand from these applications, especially during peak hours. Our goal is to develop a solution for real-time prioritization of background traffic while achieving high network utilization without affecting foreground traffic. With our solution, the background traffic must quickly yield to foreground traffic when the network is busy but must quickly recapture spare capacity when the network becomes lightly loaded. Our stated goal cannot be accomplished with trivial transport layer modifications due to the scale and complex cross-layer interactions between transport (TCP) and link (LTE) layers.

In recent years, the increasing number of machine learning algorithms require computationally intensive calculations, which could be challenging to be carried out by a single device. Fortunately, distributed computation provides promises to address this challenge, where a master device can divide computation intensive tasks into small sub-tasks and allocates sub-tasks to a group of helper devices. As a result, such distributed computing frameworks such as Spark [18] and MapReduce [19] could support large scale tasks on data size at the order of petabytes. However, such distributed computation frames also suffer from some challenges due to the heterogeneous nature of helper devices, one of which is the straggler effect. This fact harms the performance of many time-sensitive applications where the whole computing task has to be done by a hard deadline. Coding strategies have been applied in distributed computation systems to provide resiliency against the stragglers effect that could speed up the whole computation process. However, when there is a hard deadline set for the computation tasks, arbitrary coding strategies could delay the process and result in missing the deadline, which cause the whole computation tasks to fail. Therefore, it is crucial to develop optimal coding strategies for a distributed computation system with heterogeneous helper devices by taking into account the deadline.

1.2 Thesis Contributions

The thesis considers design and optimization of heterogeneous networks. In particular, we consider the intelligent transportation system and develop optimal traffic phase scheduling and shortest routing algorithms. In addition, we consider wireless networks and develop optimal resource allocation and scheduling algorithms for heterogeneous (per-flow and FIFO) queues. Moreover, we consider cellular networks and develop (i) in-network traffic controller and (ii) low priority transport protocol to support heterogeneous priority traffic. Finally, we consider distributed computing network with heterogeneous helper devices and develop optimal coded computation schemes to maximize the probability of master device meeting the deadline. More specifically, the contributions are the following:

- We study the traffic phase scheduling decisions at isolated intersections and develop a connectivity-aware traffic phase scheduling algorithm for heterogeneously connected vehicles [1].
- We study the impact of the blocking problem to the waiting time at intersections of a transportation system and develop a shortest delay routing algorithm [2].
- We study the performance of heterogeneous (per-flow and FIFO) queues over wireless networks and develop optimal flow control and scheduling algorithms [3].
- We study the heterogeneous priority traffic over cellular networks and develop (i) in-network controller (*Sneaker*) and (ii) low priority transport protocol (*Legilimens*) to managing background traffic transmission [4].

- We study the distributed computing network with heterogeneous helper devices and develop optimal coded computation scheme to maximize the probability of master device meeting the deadline.

Next, we describe each contribution with more details.

1.2.1 Connectivity-Aware Traffic Phase Scheduling for Heterogeneously Connected Vehicles

The increasing population and growing cities introduce several challenges in metropolitan areas, and one of the most challenging areas is transportation systems. Traditional traffic light scheduling algorithm does not take into account vehicles' connectivity and thus waste much time on scheduling traffics. Thanks to the large scale of connectivity in today's transportation network, vehicles are able to transmit and receive information, which has potential of reducing congestion, delay, energy, and improving reliability. However, it is crucial to understand how heterogeneous communication affects the performance of transportation systems.

In this thesis, we investigate the impact of heterogeneous communication on traffic phase scheduling problem in transportation networks. Specifically, we model arriving and departing vehicles at an intersection as a queuing model. In particular, we investigate two queuing models; single-lane model and one+two lane model. We develop a connectivity-aware traffic scheduling algorithm, which we name Connectivity-Aware Max-Weight (CAMW), by taking into account the congestion levels at intersections and the heterogeneous communications. The crucial parts of CAMW are expectation and learning components. In the expectation component, we characterize the expected number of vehicles that can pass through the intersections by taking into account the heterogeneous connectivity. In the learning component, we infer the directions of vehicles even if they do not directly communicate. The expectation and learning components collectively determine the number of vehicles that can pass through the intersections. We evaluate CAMW via simulations, which confirm our analysis, and show that our algorithm significantly improves intersection efficiency as compared to the baseline; the max-weight algorithm.

1.2.2 Blocking Avoidance in Transportation Systems

We investigate the impact of the blocking problem to transportation systems. The blocking problem naturally arises in transportation systems as multiple vehicles with different itineraries share available resources. Under that the assumption of heterogeneous communication among connected vehicles, we consider that different vehicles, depending on their Internet connection capabilities, may communicate their intentions (*e.g.,* whether they will turn left or right or continue straight) to intersections (specifically to devices attached to traffic lights). We consider that information collected by these devices are transmitted to and processed in a cloud-based traffic control system. Thus, a cloud-based system, based on the intention information, can calculate waiting times at intersections.

In this thesis, we investigate the impact of blocking problem in transportation systems by modeling arriving and departing vehicles at an intersection as a queuing model. Again, we investigate two queuing models; single-lane model and one+two lane model. For each model, we characterize average waiting times by taking into account the vehicles that can communicate their intentions (to turn left, right, or go straight) and blocking probability. We then design an algorithm that finds the routes (or set of intersections) between a starting and ending points with shortest delay. The shortest delay algorithm that we design takes into account the average waiting times at intersections, hence blocking probabilities. Lastly, we evaluate our algorithm via simulations for a multiple-intersection transportation network.

The simulation results confirm our analysis, and show that our shortest delay algorithm significantly improves over blocking-unaware schemes.

1.2.3 Flow Control and Scheduling for Heterogeneous (Per-Flow and FIFO) Queues over Wireless Networks

The dramatic increase of demand in resources poses a challenge for current wireless networks, and calls for new network control mechanisms that make better use of scarce wireless resources. Though backpressure routing and scheduling paradigm can provide utility-optimal operation for systems with per-flow queues, achievable throughput with FIFO queues in a wireless network is still an open problem.

In this thesis, we consider a general scenario where per-flow and FIFO queues coexist. We investigate the performance of these heterogeneous queues over wireless networks, and characterize the support region of the network where an arbitrary number of heterogeneous queues are shared by an arbitrary number of flows. The support region of the queueing system under investigation is non-convex. Thus, we develop a convex inner-bound on the support region, which is provably tight for certain operating points. We then develop a resource allocation scheme; dFC, and a queue-based stochastic flow control and scheduling algorithm; qFC. We show that qFC achieves optimal operating point in the convex inner bound. Lastly, we evaluate our schemes via simulations for multiple heterogeneous queues and flows. The simulation results show that our algorithms significantly improve the throughput as compared to the well-known queue-based flow control and max-weight scheduling schemes.

1.2.4 Managing Background Traffic over Cellular Network Using *Sneaker*

With the dramatic growth of connected devices in Internet of Things, new control mechanism are urgently needed to utilize the considerably limited resources. Furthermore, due to the dynamic and heterogeneous nature of different types of devices, it is critical to develop efficient prioritization traffic management mechanisms to ensure different Quality of Service (QoS) requirement are met.

In this thesis, our goal is to develop in-network controller, called *Sneaker*, that co-exists with end-to-end protocols (*i.e.,* TCP) without requiring changes to existing schedulers and manages background traffic to yield priority to regular traffic. To achieve these goals, we first study the interaction of TCP with common cellular schedulers. Then, we formulate the problem as a Network Utility Maximization (NUM) problem to determine the *optimal* transmission rate of background flows. Using this optimal transmission rate, we derive an optimal dropping rate for background flows. Because the optimal dropping rate is hard to realize in practice, we identify a close approximation to the optimal rate, which is easy to implement and works in harmony with end-to-end protocols. We show that our *practical* dropping rate avoids TCP timeouts for background flows and achieves the intended prioritization of foreground flows. Extensive ns-3 simulations confirm our analysis and show that *Sneaker* outperforms an aggressive baseline that gives strict priority to foreground traffic. Further, we also show that *Sneaker* performs better than existing low priority transport protocols.

1.2.5 Managing Background Traffic over Cellular Network Using *Legilimens*

Another way of managing background traffic in cellular network is to design a solution for low priority data transport while keeping the existing network infrastructures (*e.g.,* scheduler, QCI management, etc.) unchanged.

In this part of thesis, our goal is to develop a low priority transport protocol, called *Legilimens*, for real-time prioritization of background traffic while achieving high network utilization without affecting foreground traffic. We first studied the interaction of TCP with cellular network and identified critical reasons why existing low priority transport protocols do not work as expected. Then based on the analytical guidelines, we develop a two-phase rate control mechanism, *Legilimens*, that achieves the optimal resource allocation for low priority users. We focus on optimizing *Legilimens* for cellular downlink traffic. To that end, we design a novel algorithm that quickly estimates capacity and load based on packet inter-arrival times, not round-trip time (RTT) or one-way delay (OWD). We implement *Legilimens* in Linux as a sender-only modification to the network stack, enabling simpler and incremental deployment, without changes to the cellular infrastructure. We conduct simulations and experiments in `ns-3`, PhatomNet and real network, and results show that *Legilimens* is superior to other protocols in transferring large volumes of data without interfering with regular user traffic.

1.2.6 Optimal Coded Computation with Hard Deadlines

Coded distributed computing networks provide system robustness against stragglers due to heterogeneous nature of devices. However, when given a deadline, arbitrary coding strategies may delay whole calculation process and cause the task to fail.

In this thesis, we consider the problem of coded distributed computing with hard deadlines. In particular, we consider a distributed computing framework with one master device and n helper devices, whose task is to solve one of the most important tasks in machine learning algorithms: matrix multiplication, by a given deadline. Due to heterogeneous nature of helper devices, we assume random task processing time on each device and characterize the performance of uncoded and (n, k) MDS coded computation with hard deadlines. Moreover, we develop optimal k to maximize the probability of meeting the deadline. Obtaining the optimal k requires integer programming which could be time consuming. To speed up the calculation of k, we develop approximated solution with much reduced time complexity. Simulation results show that coded computation outperforms uncoded computation and baseline, and our approximated solution achieves very close performance as compared to the optimal solution.

1.3 Thesis Organization

The rest of the dissertation is organized as follows. In Chapter 2, we present our work on control and optimization of heterogeneously connected vehicles, namely, the connectivity-aware traffic phase scheduling algorithm and the shortest routing algorithm in a transportation system while avoiding blocking. In Chapter 3, we present the optimal flow control and scheduling algorithm for heterogeneous (per-flow and FIFO) queues over wireless networks. In Chapter 4, we present our work on managing background traffic to support heterogeneous priority data transmission in cellular networks. In Chapter 5, we present our work on optimal coded computation for distributed computing network with heterogeneous helper devices by taking into account hard deadlines. And in Chapter 6, we conclude the thesis.

CHAPTER 2

CONTROL AND OPTIMIZATION OF HETEROGENEOUSLY CONNECTED VEHICLES

The contents of this chapter are based on our works that are published in the proceedings of the 2016 ACM CarSys workshop [1] and 2015 IEEE Allerton conference [2].

We consider a transportation system of heterogeneously connected vehicles, where not all vehicles are able to communicate. Heterogeneous connectivity in transportation systems is coupled to practical constraints such that (i) not all vehicles may be equipped with devices having communication interfaces, (ii) some vehicles may not prefer to communicate due to privacy and security reasons, and (iii) communication links are not perfect and packet losses and delay occur in practice. In this context, it is crucial to develop control algorithms by taking into account the heterogeneity. In this part of the thesis, we develop (i) a connectivity-aware traffic phase scheduling algorithm for heterogeneously connected vehicles that increases the intersection efficiency (in terms of the average number of vehicles that are allowed to pass the intersection) by taking into account the heterogeneity, and (ii) a shortest delay algorithm that calculates the routes with shortest delays between two points in a transportation network.

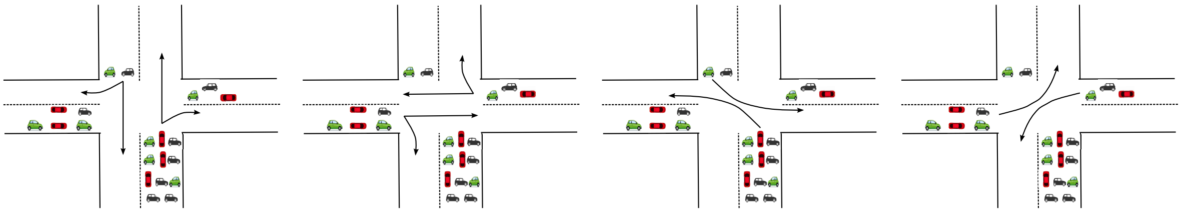
2.1 Connectivity-Aware Traffic Phase Scheduling Algorithm

2.1.1 Background

The increasing population and growing cities introduce several challenges in metropolitan areas, and one of the most challenging areas is transportation systems. In particular, the rapidly increasing number of vehicles in metropolitan transportation systems, has introduced several challenges including higher traffic congestion, delay, accidents, energy consumption, and air pollution.

Fortunately, advances in communication and networking theories offer vast amount of opportunities to address ever increasing challenges in transportation systems. In particular, connected vehicles, *i.e.,* vehicles that are connected to the Internet via cellular connections and to each other via device-to-device (D2D) connections such as Bluetooth or WiFi-Direct [10], are able to transmit and receive information to improve the control and management of traffic, which has potential of reducing congestion, delay, energy, and improving reliability. In this context, it is crucial to understand how heterogeneous communication affects the performance of transportation systems.

Example 1. *Let us consider Figure 1, which shows an isolated intersection, and all four possible traffic light phases. Traffic lights could be configured in four different phases: Phases I, II, III, and IV.* E.g., *Phase I corresponds to the case that only north-south and south-north bounds are allowed to pass through the intersection. The traffic light scheduling determines the phase that should be activated. Note that only one phase could be activated at a time. It is clear that scheduling decisions should be made based on the congestion levels of different directions (or traffic bounds). For example, selecting either Phase I or Phase III in the specific example of Figure 1 looks a better decision as compared to Phase*

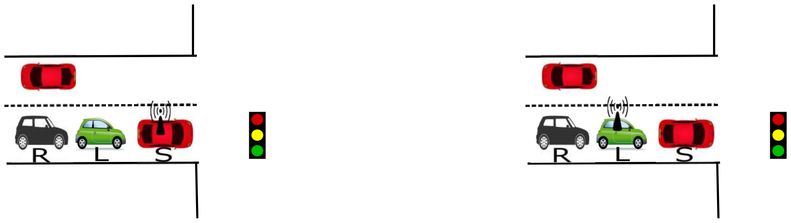


(a) Phase I ($\phi = 1$) (b) Phase II ($\phi = 2$) (c) Phase III ($\phi = 3$) (d) Phase IV ($\phi = 4$)

Figure 1. An example intersection with four possible traffic phases.

II or Phase IV, because Phase I and Phase III have a larger number of vehicles in their corresponding queues.

Example 1 is a widely known problem in network control and optimization theory, and the optimal solution to this problem is the popular max-weight algorithm [20]. The broader idea behind max-weight algorithm is to prioritize the scheduling decisions with larger weights, which corresponds to congestion level, loss probabilities, and link qualities. The max-weight idea is applied to transportation systems as well in previous work [9, 21–23] that schedules traffic phases according to congestion levels, which has potential of allowing more vehicles to pass and reduce waiting times at intersections. This approach works well in a scenario that the directions of all vehicles are known a-priori. For example, if all devices communicate with the traffic light in terms of their intentions about their directions (*e.g.,* turn right, go straight, etc.), the traffic light determines which phase to activate using the max-weight scheduling algorithm. However, due to heterogeneity of communication in connected vehicles, only a percentage of vehicles communicate their intentions. In this heterogeneous setup, new connectivity-aware traffic phase scheduling algorithms are needed as illustrated in the next example.



(a) Only the first vehicle communicates

(b) Only the second vehicle communicates

Figure 2. An example single-lane intersection, where vehicles are going straight, turning left and turning right respectively.

Example 2. *Let us consider Figure 2, which shows one of the four incoming traffic lanes in an intersection. This is a one-way single-lane road, where we call the first vehicle at the intersection as the head-of-line (HoL) vehicle. In Figure 2(a), the HoL vehicle has communication ability, and the vehicles are going straight, turning left, and turning right, respectively. In this case, the traffic light knows that the HoL vehicle is going straight (because the HoL vehicle communicates), so it arranges its phase accordingly.*

Now let us consider Figure 2(b), where the directions of vehicles are the same; i.e., *straight, left, and right. Yet, in this scenario HoL vehicle does not communicate, but only the vehicle behind HoL communicates. In this case, although the traffic light knows that the second vehicle is going to the left, it has no idea of the HoL vehicle's intention. If the traffic phase, possibly determined as a solution to the max-weight algorithm, does not match the intention of the HoL vehicle, then the HoL vehicle blocks the other vehicles at the intersection, and no vehicles can pass. Similarly, HoL blocking can be observed in more involved multiple-lane scenarios. As seen, the max-weight algorithm may not be optimal in* *some scenarios due to heterogeneous connectivity, which makes the development of new scheduling algorithms, by taking into account heterogeneity, crucial.* □

In this section , we develop a connectivity-aware traffic phase scheduling algorithm by taking into account heterogeneous communications of connected vehicles. Our approach follows a similar idea to the max-weight scheduling algorithm, which makes scheduling decisions based on congestion levels at intersections. However, our algorithm, which we name Connectivity-Aware Max-Weight (CAMW), is fundamentally different from the max-weight as we take into account heterogeneous communications while determining congestion levels. In particular, CAMW has two critical components to determine congestion: (i) Expectation: This component calculates the expected number of vehicles that can pass through the intersection at every phase based on the number of vehicles, and the percentage of communicating vehicles at the intersection. (ii) Learning: This component learns the directions of vehicles even if the vehicles do not directly communicate with the traffic light. The expectation and learning components of our algorithm operate together in harmony to make better decision on traffic phase scheduling. The simulation results demonstrate that CAMW algorithm significantly improves the intersection efficiency (in terms of the average number of vehicles that are allowed to pass the intersection) over the baseline algorithm; max-weight. The following are the key contributions of this work:

- We investigate the impact of heterogeneous communication on traffic phase scheduling problem in transportation networks. We develop a connectivity-aware traffic scheduling algorithm, which we name Connectivity-Aware Max-Weight (CAMW), by taking into account the congestion levels at intersections and the heterogeneous communications.

- The crucial parts of CAMW are expectation and learning components. In the expectation component, we characterize the expected number of vehicles that can pass through the intersections by taking into account the heterogeneous connectivity. In the learning component, we infer the directions of vehicles even if they do not directly communicate. The expectation and learning components collectively determine the number of vehicles that can pass through the intersections.
- We evaluate CAMW via simulations, which confirm our analysis, and show that our algorithm significantly improves intersection efficiency as compared to the baseline; the max-weight algorithm.

2.1.2 Related Work

This work combines ideas from traffic phase scheduling, queuing theory, and network optimization. In this section, we discuss the most relevant literature from these areas.

Traffic phase scheduling: Design and development of traffic phase scheduling algorithms have a long history; more than 50 years [24]. Thus, there is huge literature in the area, especially on the design of optimal pre-timed policies [24–26], which activate traffic phases according to a time-periodic predefined schedule. These policies do not meet expectations under changing arrival times, which require adaptive control [27]. The adaptive control mechanisms including [28], [25], [29], [30], [31] and [32], optimize control variables, such as traffic phases, based on traffic measures, and apply them on short term.

Queueing theory: Using queuing theory to analyze transportation systems has also very long history [33]. *E.g.,* [24,34,35] considered one-lane queues and calculated the expected queue length and arrivals using probability generation functions. Other modeling strategies are also studied; such as the queuing network model [36], cell transmission model [37], store-and-forward [10], and petri-nets [38].

Network optimization and its applications to transportation systems: Max-weight scheduling algorithm and backpressure routing and scheduling algorithms [20] arising from network optimization area has triggered significant research in wireless networks [39, 40]. This topic has also inspired research in transportation systems [9, 21–23]. Feedback control algorithms to ensure maximum stability are proposed both under deterministic arrivals [22] and stochastic arrivals [23,41] following backpressure idea. The infinite buffer assumption of backpressure framework is studied by capacity aware back-pressure algorithm in [42].

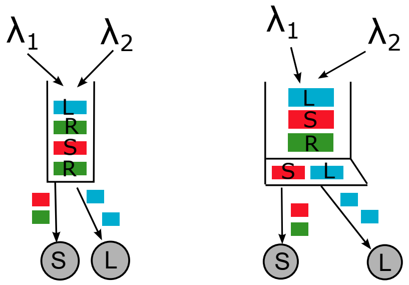
Our work in perspective: As compared to the previous work briefly summarized above, our work focuses on connected vehicles and investigates the scenario where vehicles communicate heterogeneously. In this scenario, we develop an efficient connectivity-aware traffic phase scheduling algorithm by employing expectation and learning of congestion levels at intersections.

2.1.3 System Model

In this section, we present our system model including traffic lights and phases as well as our queuing models of the traffic.

Traffic lights and phases: In our system model, we focus on an intersection controlled by a traffic light. The four traffic phases we consider in this chapter are shown in Figure 1. We define ϕ as a phase decision, *e.g.,* $\phi = 1$ corresponds to Phase I in Figure 1. The set of phases is Φ, and $\phi \in \Phi$.

We consider that time is slotted, and at each time slot t, a phase decision is made. Each traffic phase lasts for n time slots. Vehicles have a chance to pass the intersection only when the corresponding traffic



(a) `Queue I` (b) `Queue II`

Figure 3. Two queuing models considered in this chapter, where λ_1 and λ_2 are the arrival rates of straight-going and left-turning traffic, respectively.(a) Single-lane traffic model. (b) One+two lane model.

phase is active, *i.e.,* ON. For instance, vehicles in the south-north bound lanes may pass the intersection only when phase $\phi = 1$ is ON in Figure 1.

Modeling intersections with queues: We model the isolated intersection as a set of queues following [2]. Typically, there are four queues for each direction (for south-north, north-south, west-east, and east-west bound) at an intersection. We specifically focus on one direction and model it using two models: `Queue I`, which is one-lane model shown in Figure 3(a) and `Queue II`; which is a one+two lane model shown in Figure 3(b).

Note that for both of `Queue I` and `Queue II`, we can consider straight-continuing and right-turning traffic as the same traffic, since they share the similar right of way. Thus, to demonstrate the analysis in a simple way, we simply consider that the right-turning and straight-continuing traffics are combined together, and we call both right-turning and straight-continuing vehicles as straight-going vehicles.

At each slot, vehicles arrive into intersections, where λ_1 and λ_2 are the average arrival rates of straight-going and left-turning vehicles, respectively. In our analysis, the arrivals can follow any i.i.d. distribution. In this setup, when a vehicle enters the intersection, it can connect to the traffic light either using cellular or vehicle-to-vehicle communications. Thus, it can communicate its intention with the traffic light about its destination, *i.e.,* turning left, going straight, etc. The probability of communication for each vehicle is ρ.

If a vehicle does not communicate, we model their intentions probabilistically, where p_1 is the probability that a vehicle (which does not communicate its intention) will go straight, while p_2 is the probability that it will turn left.

2.1.4 Connectivity-Aware Traffic Phase Scheduling

2.1.4.1 CAMW: Connectivity-Aware Max-Weight

In this section, we develop our connectivity-aware traffic phase scheduling algorithm by taking into account heterogeneous communications. We consider the setup shown in Figure 1 for phases. Our scheduling algorithm, which we call Connectivity-Aware Max-Weight (CAMW), determines the phase ϕ by optimizing

$$\begin{aligned} &\max_{\boldsymbol{\phi}} \sum_{i \in \{1,\ldots 4\}} Q_i(t)\tilde{E}(K_i^{\phi}(t)) \\ &\text{s.t. } \phi \in \Phi. \end{aligned} \tag{2.1}$$

where $Q_i(t)$ is the number of vehicles in the ith incoming queue at time slot t, and $\tilde{E}(K_i^{\phi}(t))$ is the estimated number of vehicles that can pass the intersection from the ith incoming queue under traffic phase $\phi \in \Phi$. Note that one active phase lasts for n time slots and it takes one time slot for a vehicle to pass the intersection. In other words, at most n vehicles in a queue can pass the intersection during one green light phase. The optimization problem in Equation 2.1 applies to all queuing models (*i.e.,* includes both `Queue I` and `Queue II`).

Note that Equation 2.1 determines the phase by taking into account $Q_i(t)$ and $\tilde{E}(K_i^{\phi}(t))$. The queue size information $Q_i(t)$ can be easily determined by traffic lights using sensors that count the number of approaching vehicles. In other words, Equation 2.1 prioritizes phases with larger $Q_i(t)$ values. This is an approach followed by the classical max-weight algorithm. However, as we discussed earlier, using $Q_i(t)$ alone is not sufficient when vehicles heterogeneously communicate with traffic lights. In this case, since each device has different destinations, blocking can occur. *I.e.,* even if $Q_i(t)$ is large, the number of vehicles that can pass through the intersection could be small due to blocking. Thus, to reflect this fact, we include the term $\tilde{E}(K_i^{\phi}(t))$ in the optimization problem.

$\tilde{E}(K_i^{\phi}(t))$ is the estimated number of vehicles that can pass the intersection from the ith incoming queue under traffic phase $\phi \in \Phi$. $\tilde{E}(K_i^{\phi}(t))$ is found using two steps: expectation and learning. The key idea behind expectation part is to calculate the expected number of vehicles, which is $E(K_i^{\phi}(t))$, that can pass the intersection at phase ϕ, while the key idea of the learning part is to fine tune $E(K_i^{\phi}(t))$ and find $\tilde{E}(K_i^{\phi}(t))$ by learning the directions of vehicles that do not communicate. In the next two sections, we present the expectation and learning components of CAMW.

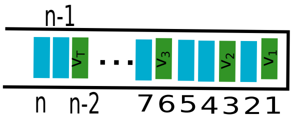


Figure 4. An illustrative example of communicating vehicles in a queue at a time slot. Communicating vehicles are at labeled locations; $v_1, v_2, \cdots, v_T$.

2.1.4.2 Expectation

2.1.4.2.1 Calculation of $E(K_i^{\phi}(t))$ for `Queue I`

Let us focus on phase $\phi \in \Phi$ and the ith queue, where $i \in \{1,2,3,4\}$. In this setup, $T(t)$ ($T(t) \leq n$) denotes the number of vehicles that have communication abilities at time slot t, and $v_l(t)$ ($l = 1, 2, \cdots, T$) denotes the location of the lth communicating vehicle in the queue. For example, $v_2(t) = 3$ means that the second communicating vehicle in the queue is actually the third vehicle in the queue. Figure 4 illustrates an example locations of communicating vehicles. Note that the vehicles that do not communicate are not assigned any location labels.

Now, let us define two conditions; C_1 and C_2. The first condition C_1 requires that all communicating vehicles would like to go to the same direction and aligned with the traffic phase, while the second condition C_2 corresponds to the case that the first communicating vehicle that is not aligned with the traffic phase is in the location of $v_L(t)$ ($L = 1, 2, \cdots, T$). Note that the conditions C_1 and C_2 are complementary. The next theorem characterizes the expected number of vehicles that would leave queue i at phase $\phi = 1$. Note that $E(K_i^{\phi=1}(t))$ calculation can be directly generalized to $E(K_i^{\phi}(t))$, $\forall \phi \in \Phi$.

Theorem 1. *Assume that all the queues in an intersection follow* `Queue I`. *The expected number of vehicles that would leave the* i*th queue and pass the intersection at traffic phase* $\phi = 1 \in \Phi$ *is characterized by*

$$E(K_i^{\phi=1}(t)) = \begin{cases} \sum_{l=0}^{T(t)} \frac{p_1^{1-l}}{p_2}((p_1 + p_2 v_l(t))p_1^{v_l(t)-1} \\ +(1 - 2p_1 - p_2 v_{l+1}(t))p_1^{v_{l+1}(t)-2}) \\ +np_1^{n-T(t)}, & \text{if } C_1 \text{ holds} \\ \\ \sum_{l=0}^{L-1} \frac{p_1^{1-l}}{p_2}((p_1 + p_2 v_l(t))p_1^{v_l(t)-1} \\ +(1 - 2p_1 - p_2 v_{l+1}(t))p_1^{v_{l+1}(t)-2}) \\ +(v_L(t) - 1)p_1^{v_L(t)-L}, & \text{if } C_2 \text{ holds.} \end{cases} \tag{2.2}$$

Proof. Here, we specifically focus on the calculation of $E(K_i^{\phi=1}(t))$, where $\phi = 1$ corresponds to the phase in Figure 1(a) to explain our the proof in an easier way.

We first derive the calculation of $E(K_i^{\phi=1}(t))$ when all communicating vehicles are going straight. The calculation of $E(K_i^{\phi=1}(t))$ for other cases will be obtained based on this derivation. If all communicating vehicles are going straight at time slot t, we can consider the queue as divided into $(T + 1)$ *blocks* by the T communicating vehicles. (Note that T is the number of communicating vehicles in a queue).

Let a random variable J denote the number of vehicles that can pass the intersection. The probability that j vehicles pass the intersection is $P[J = j]$, and it behaves similarly to the geometric distribution. However, the probability distribution is different when j falls into different *blocks* due to the communicating vehicles that go straight. To be more precise, we have

$$P[J = j] = \begin{cases} p_1^j p_2, & 1 \le j \le v_1 - 2 \\ p_1^{j-1} p_2, & v_1 \le j \le v_2 - 2 \\ \vdots & \\ p_1^{j-T} p_2, & v_T \le j \le n - 1 \\ p_1^{n-T}, & j = n \end{cases} \tag{2.3}$$

Note that $P[J = v_1 - 1]$, $P[J = v_2 - 1]$, $\cdots$, $P[J = v_T - 1]$ are all 0. The reason is that the communicating vehicles at locations $v_1, v_2, \cdots, v_T$ are all going straight, and if $v_l - 1$ vehicles can pass the intersection. Then, v_l vehicles can pass the intersection for sure ($l = 1, 2, \cdots, T$).

Using Equation 2.3, we can obtain the expected number of vehicles that can pass the intersection as $E(K_i^{\phi=1}(t))$ when all communicating vehicles are going straight. *I.e.,*

$$\begin{aligned} E(K_i^{\phi=1}(t)) = \sum_{j=1}^{v_1-2} j p_1^j p_2 + \sum_{j=v_1}^{v_2-2} j p_1^{j-1} p_2 + \cdots \\ + \sum_{j=v_T}^{n-1} j p_1^{j-T} p_2 + n p_1^{n-T} \end{aligned} \tag{2.4}$$

In Equation 2.4, $\sum_{j=v_l}^{v_{l+1}-2} jp_1^{j-l}p_2$ can be expressed as $p_1^{1-l}\, p_2 \sum_{j=v_l}^{v_{l+1}-2} jp_1^{j-1} = p_1^{1-l} p_2 \frac{\partial(\sum_{j=v_l}^{v_{l+1}-2} p_1^j)}{\partial p_1} = \frac{p_1^{1-l}}{p_2}((p_1 + p_2 v_l)p_1^{v_l-1} + (1 - 2p_1 - p_2 v_{l+1})p_1^{v_{l+1}-2})$. Thus, we can obtain $E(K_i^{\phi=1}(t))$ when all communicating vehicles are going straight as

$$\begin{aligned} E(K_i^{\phi=1}(t)) = \sum_{l=0}^{T} \frac{p_1^{1-l}}{p_2}&((p_1 + p_2 v_l)p_1^{v_l-1} \\ +(1 - 2p_1 - p_2 v_{l+1})p_1^{v_{l+1}-2}) &+ np_1^{n-T} \end{aligned} \tag{2.5}$$

Note that we have $v_0 = 1, v_{T+1} = n + 1$ in Equation 2.5 to make it consistent with Equation 2.4.

When there are some communicating vehicles going left, let $v_L(t)$ be the location of the first communicating vehicle that goes left. There are $(L-1)$ communicating vehicles in front of $v_L(t)$ that going straight and $(T - L)$ communicating vehicles behind $v_L(t)$ which will be blocked for sure. Now, we only focus on the vehicles between the location 1 to $(v_L(t) - 1)$. There are $(L - 1)$ communicating vehicles among them, and all of the communicating vehicles are going straight. Thus, we can use the similar analysis as used in Equation 2.4 except that now the maximum number of vehicles that can pass the intersection is $(v_L(t) - 1)$ instead of n. Therefore, we have the expected number of vehicles that can pass the intersection $E(K_i^{\phi=1}(t))$ when the first communicating vehicle that turns left is at location $v_L(t)$. Thus,

$$\begin{aligned} E(K_i^{\phi=1}(t)) = \sum_{l=0}^{L-1} \frac{p_1^{1-l}}{p_2}&((p_1 + p_2 v_l)p_1^{v_l-1} + \\ (1 - 2p_1 - p_2 v_{l+1})p_1^{v_{l+1}-2}) &+ (v_L(t) - 1)p_1^{v_L(t)-L} \end{aligned} \tag{2.6}$$

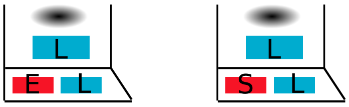
By taking into account all the $(T+1)$ situations, we conclude that

$$E(K_i^{\phi=1}(t)) = \begin{cases} \sum_{l=0}^{T(t)} \frac{p_1^{1-l}}{p_2}((p_1+p_2v_l(t))p_1^{v_l(t)-1} \\ +(1-2p_1-p_2v_{l+1}(t))p_1^{v_{l+1}(t)-2}) \\ +np_1^{n-T(t)}, & \text{if } C_1 \text{ holds} \\ \\ \sum_{l=0}^{L-1} \frac{p_1^{1-l}}{p_2}((p_1+p_2v_l(t))p_1^{v_l(t)-1} \\ +(1-2p_1-p_2v_{l+1}(t))p_1^{v_{l+1}(t)-2}) \\ +(v_L(t)-1)p_1^{v_L(t)-L}, & \text{if } C_2 \text{ holds.} \end{cases} \tag{2.7}$$

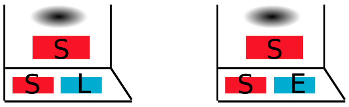
By following the same analysis, we can obtain $E(K_i^{\phi}(t))$ for $\phi = 2, 3, 4$. This concludes the proof. □

2.1.4.2.2 Calculation of $E(K_i^{\phi}(t))$ for `Queue II`

`Queue II` assumes that there are dedicated lanes for left-turning and straight-going vehicles, which makes it fundamentally different than `Queue I`. In this setup, we consider that traffic lights can sense whether the HoL location of each dedicated lane is empty or not. Thus, in `Queue II`, the first two vehicles in the queue will indirectly communicate their intentions to the traffic light. Figure 5 demonstrates four possible configurations for HoL vehicles. For example, in Figure 5(a), HoL position of the



(a) Conf. I (b) Conf. II



(c) Conf. III (d) Conf. IV

Figure 5. Four possible configurations (Conf. I to Conf IV) for the first three vehicles in `Queue II`, where L and S denote that the intention of the vehicle is to turn left or go straight, respectively, while E denotes that the location is empty (due to previous blocking).

straight going lane is empty (shown with E), the traffic light will know that two vehicles in the queue will turn left. On the other hand, in Figure 5(b), the traffic light knows that in the dedicated lanes, one vehicle will go straight, and the other will turn left, but it does not know the intentions of the other vehicles as long as they do not explicitly communicate with the traffic light.

The crucial observation with `Queue II` is that if the vehicles that indirectly communicate with the traffic light are separated from the queue, the rest of the vehicles form a *sub-queue*. For example, all the vehicles other than (i) the first two left-turning vehicles in Figure 5(a), and (ii) the two vehicles that are going straight and turning left in Figure 5(b), form a *sub-queue*. The important property of the *sub-queue* is that it follows `Queue I`, and can be modeled using the location labels as shown in Figure 4. Thus, we can calculate $E(K_i^{\phi}(t))$ of `Queue II` using the similar analysis we have in Section 2.1.4.2.1. Next, we provide the details of our $E(K_i^{\phi=1}(t))$ calculation. Again, following the same idea, one can calculate $E(K_i^{\phi}(t)), \forall \phi \in \Phi$

Let $T(t)$ denotes the number of communicating vehicles in the *sub-queue* at time t, C_3 is the condition that all communicating vehicles in the *sub-queue* go to the same direction aligned with the traffic phase, and C_4 denotes the condition that the first communicating vehicle in the *sub-queue* that goes to a different direction than what the traffic phase allows is at location $v_L(t)$ ($L = 1, 2, \cdots, T$). The next theorem characterizes the expected number of vehicles that would leave queue i at phase $\phi = 1$ for model `Queue II`.

Theorem 2. *Assume that all the queues in an intersection follow* `Queue II`*. Then, if the first three vehicles of the ith incoming queue are in the form of Figure 5(a) or Figure 5(d), the expected number of transmittable vehicles is characterized by*

$$E(K_i^{\phi=1}(t)) = \begin{cases} 2 + \sum_{l=0}^{T(t)} \frac{p_1^{1-l}}{p_2}((p_1 + p_2 v_l(t)) \\ p_1^{v_l(t)-1} + (1 - 2p_1 - p_2 v_{l+1}(t)) \\ p_1^{v_{l+1}(t)-2}) + (n-2)p_1^{n-2-T(t)}, & \text{if } C_3 \text{ holds} \\ \\ 2 + \sum_{l=0}^{L-1} \frac{p_1^{1-l}}{p_2}((p_1 + p_2 v_l(t)) \\ p_1^{v_l(t)-1} + (1 - 2p_1 - p_2 v_{l+1}(t)) \\ p_1^{v_{l+1}(t)-2}) + (v_L(t) - 1)p_1^{v_L(t)-L}, & \text{if } C_4 \text{ holds} \end{cases} \tag{2.8}$$

where $T(t) \leq n - 2$.

And if the first three vehicles of the ith incoming queue are in the form of Figure 5(b) or Figure 5(c), the expected number of transmittable vehicles is characterized by

$$E(K_i^{\phi=1}(t)) = \begin{cases} 1 + \sum_{l=0}^{T(t)} \frac{p_1^{1-l}}{p_2}((p_1 + p_2 v_l(t)) \\ p_1^{v_l(t)-1} + (1 - 2p_1 - p_2 v_{l+1}(t)) \\ p_1^{v_{l+1}(t)-2}) + (n-1)p_1^{n-1-T(t)}, & \text{if } C_3 \text{ holds} \\ \\ 1 + \sum_{l=0}^{L-1} \frac{p_1^{1-l}}{p_2}((p_1 + p_2 v_l(t)) \\ p_1^{v_l(t)-1} + (1 - 2p_1 - p_2 v_{l+1}(t)) \\ p_1^{v_{l+1}(t)-2}) + (v_L(t) - 1)p_1^{v_L(t)-L}, & \text{if } C_4 \text{ holds} \end{cases} \tag{2.9}$$

where $T(t) \leq n - 1$.

Proof. The number of vehicles that can be guaranteed to pass the intersection under certain traffic phase depends on the configuration of the first three vehicles in the queue. First, we consider the case that the first three vehicles are in the form of Figure 5(a) or Figure 5(d). In this case, at least two vehicles can pass the intersection for the corresponding traffic phase, so we need to consider the rest of the vehicles, *i.e.,* $n - 2$ vehicles assuming that n is the queue size. Noting that $n - 2$ vehicles form a *sub-queue* in this setup, and assuming that $T(t)$ ($T(t) \leq n - 2$) vehicles communicate the *sub-queue*, it is clear that the *sub-queue* is represented by `Queue I`. Thus, Equation 2.8 is obtained by adding two to Equation 2.2.

On the other hand, if the first three vehicles are in the form of Figure 5(b) or Figure 5(c), at least one vehicle can pass the intersection at any traffic phase configuration. In this scenario, one vehicle is considered as guaranteed to be transmitted, and the rest of the vehicles ($n - 1$ vehicles) form a *sub-queue*. Similar to above discussion, the *sub-queue* follows `Queue I`, so Equation 2.9 is obtained by adding one to Equation 2.2. This concludes the proof. $\square$

2.1.4.3 Learning

In the previous section, we characterized the expected number of vehicles $E(K_i^{\phi}(t))$ that can pass an intersection at phase ϕ from queue i. However, in our CAMW algorithm, which solves Equation 2.1, we do not use $E(K_i^{\phi}(t))$. The reason is that $E(K_i^{\phi}(t))$ is an expected value and its granularity is poor. In other words, if we use $E(K_i^{\phi}(t))$ in Equation 2.1, we may end up with choosing a traffic phase that allows no vehicles passing the intersection. In this case, the intersection is *blocked*. More importantly, once the intersection is *blocked*, if we keep using $E(K_i^{\phi}(t))$ in Equation 2.1, we may end up with choosing the wrong traffic phase next time with high probability, which leads to a deadlock. To address this issue, we introduce the learning mechanism, which works in the following way.

We assume that traffic lights can infer if blocking occurs at intersections, and use this information in future decisions. For example, assume that the selected traffic phase at time $t - 1$ is $\phi = 1$ (as shown in Figure 1(a)), and $\tilde{E}(K_i^{\phi=1}(t-1)) = E(K_i^{\phi=1}(t-1))$. If blocking occurs, then the traffic light can learn that both of the HoL vehicles in south-north bound queues must go left. Using this information, $\tilde{E}(K_i^{\phi=1}(t))$ is set to zero at time t so that $\phi = 1$ is not selected again. $\tilde{E}(K_i^{\phi=1}(t + \Delta))$ is set to $E(K_i^{\phi=1}(t + \Delta))$ again immediately after some vehicles are transmitted from the queues. This may take Δ time slots. This learning mechanism applies to both `Queue I` and `Queue II`, but in `Queue II`, separate lanes for each direction makes the learning process by default. *I.e.,* in `Queue II`, $\tilde{E}(K_i^\phi(t)) = E(K_i^\phi(t)), \forall t$.

2.1.5 Performance Evaluation

In this section, we consider an intersection controlled by a traffic light. Each arriving vehicle to the intersection can communicate with probability ρ. Each green phase lasts for one or more time slots. We assume that the arrival rate to each queue in the intersection is the same; *i.e.,* λ_1 and λ_2 are the same $\forall i \in \{1, 2, 3, 4\}$. We present the simulation results of our Connectivity-Aware Max-Weight (CAMW) algorithm for both of `Queue I` and `Queue II`, as compared to the baseline, the max-weight algorithm, which is briefly described next.

2.1.5.1 The baseline: max-weight algorithm

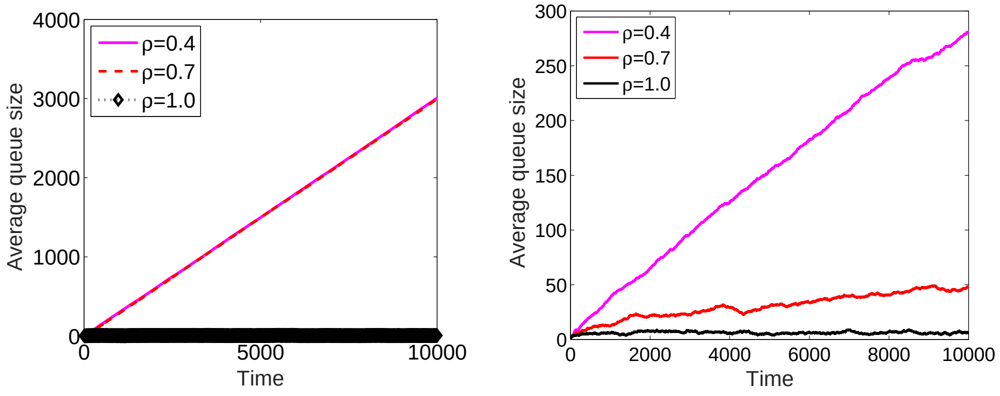
The max-weight scheduling algorithm determines a traffic phase as a solution to

$$\begin{aligned} &\max_{\boldsymbol{\rho}} \sum_{i=1}^{4} Q_i(t) K_i^\phi(t) \\ &\text{s.t. } \phi \in \Phi, \end{aligned} \tag{2.10}$$

where $K_i^\phi(t)$ is the weight of queue i for phase ϕ.[1] The value of $K_i^\phi(t)$ depends on the intersection type and the corresponding queuing models, which is explained next.

---

[1] Note that $K_i^\phi(t) = 1$ in the original max-weight algorithm, while it varies in Equation 2.10 as explained in this section. Thus, although we call this baseline the max-weight algorithm, it is actually the improved version of the classical max-weight algorithm.



(a) Max-weight (b) CAMW

Figure 6. The average queue size versus time for `Queue I`. Each green phase lasts for two time slots. The arrival rate is $\lambda_1 = 0.18$ and $\lambda_2 = 0.12$ to each of the queue in the intersection.

First, let us consider `Queue I`. If the HoL vehicle in the ith queue can communicate, then $K_i^{\phi}(t) = 1$ for the phase that is aligned with the direction of the HoL vehicle and $K_i^{\phi}(t) = 0$ for the other three phases. If the HoL vehicle cannot communicate, max-weight considers $K_i^{\phi}(t) = 1$ for the phases that control the ith queue if the queue length is larger than zero.

Second, we assume that all the queues in the intersection follow `Queue II`. In this setup, we take into account the first two vehicles in the dedicated lanes. For example, if the first two vehicles from the ith incoming queue are in the form of Figure 5(a), then $K_i^{\phi}(t) = 1$ for the left turning phase, and $K_i^{\phi}(t) = 0$ for the other phases. On the other hand, if the first two vehicles are in the form of Figure 5(b), then $K_i^{\phi}(t) = 1$ for both the left-turning and straight-going phases.

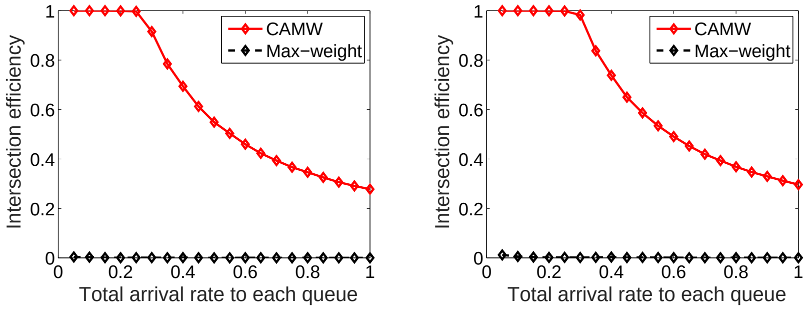
2.1.5.2 Evaluation of CAMW for `Queue I`

We first assume all the queues in the intersection follow `Queue I`, and evaluate our CAMW algorithm as compared to the baseline; max-weight. The evolution of the average queue size of the intersection for different scheduling algorithms is presented in Figure 6. Each green phase lasts for two time slots. The arrival rate is $\lambda_1 = 0.18$ and $\lambda_2 = 0.12$ to each of the queue in the intersection. It can be observed that when the communication probability is $\rho = 1.0$, both of the algorithms have the similar performance. This is because every vehicle can communicate, so the max-weight algorithm, since the traffic light can communicate with the HoL vehicle, can align the phases with the direction of HoL vehicle. However, when the communication probability reduces to $\rho = 0.7$, max-weight cannot stabilize the queues, while CAMW stabilizes. When $\rho = 0.4$, neither CAMW nor max-weight can stabilize the queues, because the arrival rates fall out of the stability region. As can be seen CAMW supports higher traffic rates than the max-weight algorithm thanks to exploiting connectivity of vehicles.

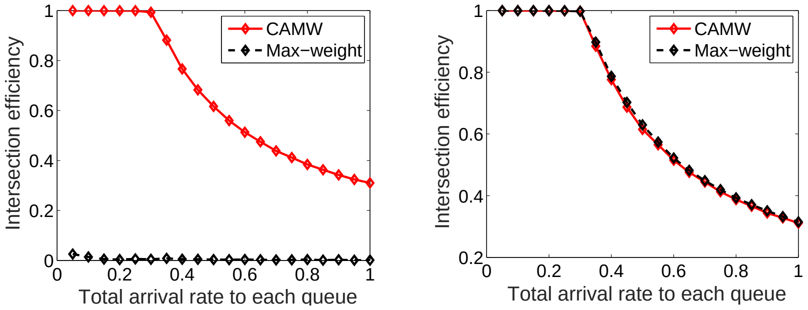
Figure 7 presents the intersection efficiency versus total arrival rate to each queue for different communication probability ρ. The intersection efficiency is defined as the ratio of departing traffic to arrival traffic. In this setup, each green phase lasts for two time slots. Each queue has the same arrival rate, and $\lambda_1 = 1.5\lambda_2$. It can be observed that when $\rho = 1.0$, both of the algorithms can achieve very similar intersection efficiency. However, if $\rho \neq 1$, the intersection efficiency of max-weight scheduling algorithm drops almost to zero, while CAMW can still achieve satisfying intersection efficiency thanks to taking into account heterogeneous communication probabilities.

2.1.5.3 Evaluation of CAMW for `Queue II`

In this section, we assume all the queues in the intersection follow `Queue II`. The evolution of the average queue size of the intersection using CAMW and max-weight algorithm for different communication probability ρ is presented in Figure 8. The arrival rate to each queue is $\lambda_1 = \lambda_2 = 0.2$ and each green phase lasts for two time slots. It can be observed from Figure 8(a) that when



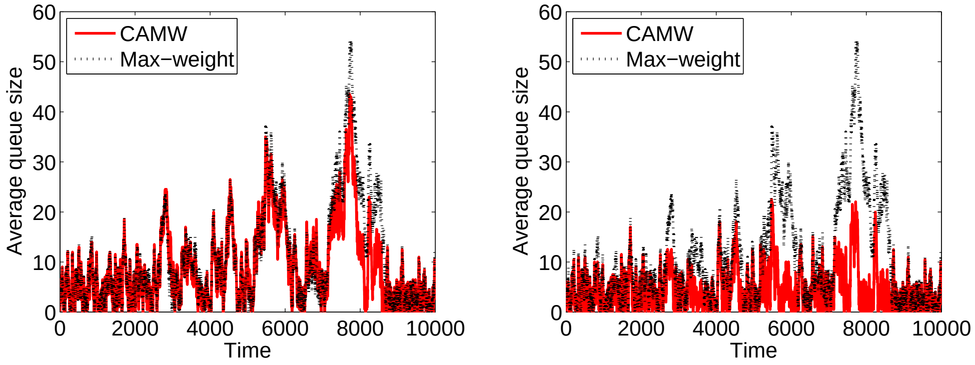
(a) $\rho = 0.1$ (b) $\rho = 0.4$



(c) $\rho = 0.7$ (d) $\rho = 1.0$

Figure 7. Intersection efficiency versus total arrival rate to each queue with different communication probability ρ for `Queue I`. Each green phase lasts for two time slots and each queue has the same arrival rate and $\lambda_1 = 1.5\lambda_2$.

communicating probability ρ is small, CAMW is slightly better than the max-weight algorithm, which is because both of the two algorithm select traffic phases in a similar way when ρ is small. The average queue sizes over 10,000 time slots when $\rho = 0.1$ using max-weight and CAMW are 10.6601 and 9.0236, respectively. It can be observed from Figure 8(b) that when communicating probability ρ is large, our algorithm improves much over max-weight algorithm. This is because the estimation accuracy in our algorithm improves as ρ increases, which allows more vehicles to pass at each green phase. When

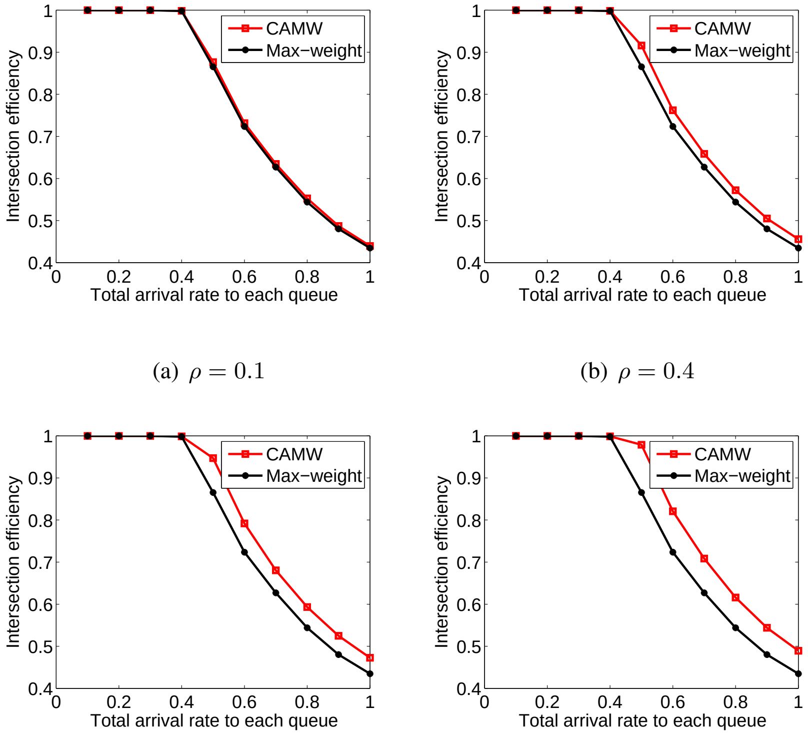


(a) $\rho = 0.1$ (b) $\rho = 0.9$

Figure 8. The evolution of the average queue size of the intersection using our algorithm and max-weight algorithm for different communication probability ρ for `Queue II`. The arrival rate to each queue is $\lambda_1 = \lambda_2 = 0.2$ and each green phase lasts for two time slots.

$\rho = 0.9$, the average queue size over 10,000 time slots using max-weight and CAMW is 10.6601 and 4.3873, respectively. Note that CAMW performs better than max-weight when ρ increases in `Queue II`, which is against the observation we had in `Queue I`. The reason is that while ρ affects max-weight's decision about HoL vehicles as explained in Equation 2.10 in `Queue I`, it does not have any effect in `Queue II`.

Figure 9 presents the intersection efficiency versus total arrival rate to each queue for different communication probabilities ρ. Each queue has the same total arrival rate and $\lambda_1 = \lambda_2$. Each green phase lasts for two time slots. It can be observed that the performance of our algorithm improves as the communicating probability ρ increases, while max-weight has the same performance as ρ changes. The reason is that the estimation accuracy in our algorithm improves as ρ increases, so CAMW performs



(a) $\rho = 0.1$ (b) $\rho = 0.4$

(c) $\rho = 0.7$ (d) $\rho = 1.0$

Figure 9. Intersection efficiency versus total arrival rate to each queue with different communication probability ρ for `Queue II`. Each queue has the same total arrival rate and $\lambda_1 = \lambda_2$, and each green phase lasts for two time slots.

better than the max-weight algorithm as ρ increases. Note that CAMW improves over max-weight by 14%, which is significant.

2.1.6 Conclusion

In this chapter, we considered a transportation system of heterogeneously connected vehicles, where not all vehicles are able to communicate. For this setup, we developed a connectivity-aware max-weight scheduling (CAMW) algorithm by taking into account the connectivity of vehicles. The crucial components of CAMW are expectation and learning components, which determine the estimated number of vehicles that can pass through the intersections by taking into account the heterogeneous communications. The simulations results show that CAMW algorithm significantly improves the intersection efficiency over max-weight.

2.2 Blocking Avoidance in Transportation Systems

The blocking problem naturally arises in transportation systems as multiple vehicles with different itineraries share available resources. In this section, we investigate the impact of the blocking problem to the waiting time at the intersections of transportation systems. Again, we assume that different vehicles, depending on their Internet connection capabilities, may communicate their intentions (*e.g.,* whether they will turn left or right or continue straight) to intersections (specifically to devices attached to traffic lights). We consider that information collected by these devices are transmitted to and processed in a cloud-based traffic control system. Thus, a cloud-based system, based on the intention information, can calculate average waiting times at intersections. We consider this problem as a queuing model, and we characterize average waiting times by taking into account (i) blocking probability, and (ii) vehicles' ability to communicate their intentions. Then, by using average waiting times at intersection, we develop a shortest delay algorithm that calculates the routes with shortest delays between two points in a transportation network. Our simulation results confirm our analysis, and demonstrate that our shortest delay algorithm significantly improves over baselines that are unaware of the blocking problem.

2.2.1 Background

In today's metropolitan transportation systems, congestion is one of the major problems. Traffic congestion causes delayed travel times as well as more energy consumption. In this section, we inves-

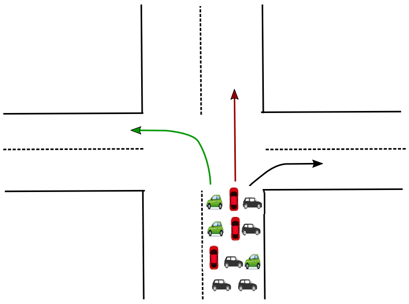
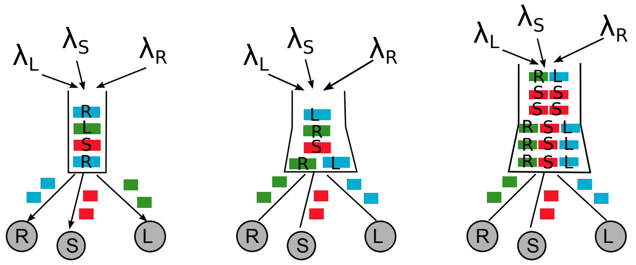


Figure 10. An example intersection with multiple vehicles with different routes.

tigate the impact of the blocking problem to transportation systems. The blocking problem naturally arises in transportation systems as multiple vehicles with different itineraries share available resources. For example, there may be multiple vehicles that would like to continue straight at an intersection, and they can block the other vehicles that would like to turn left. The next example illustrates the blocking problem using a canonical example.

Example 3. *Let us consider Figure 10, which is an example intersection, where vehicles may turn left, go straight, or turn right. Figure 11(a) is a simple queue representation of Figure 10, where one-way single-lane traffic is represented as a first-in-first-out (FIFO) queuing mechanism, and head-of-line (HoL) vehicle corresponds to the first vehicle in the queue. Let us assume that the HoL vehicle would like to turn right, but left-turn phase is on. In this case, the HoL vehicle blocks the other vehicles that are waiting in the queue,* i.e., *at the intersection. Similar blocking behavior could also be observed in more realistic scenarios such as in Figure 11(b), where lanes are dedicated to turn right (and go straight) or turn left. For example, in Figure 11(b), let us assume that two vehicles are allowed to pass through an intersection when light is on. However, HoL vehicles would like to turn left and right, and the vehicle just behind the HoL vehicles would like to go straight. In this case, if left or right light is on, one vehicle*



(a) Single-lane queue (b) One+Two-lane queue (c) Two+Three-lane queue

Figure 11. Representation of the south-north bound queue in the intersection demonstrated in Figure 10 using (a) single-lane, (b) one+two-lane, and (c) two+three-lane queuing models. λ_L, λ_S, and λ_R are the arrival rates of vehicles to the queue with destinations on the left, straight, and right, respectively.

can pass, if go-straight light is on, no vehicle can pass. Thus, even though two vehicles are allowed to pass, only one vehicle can pass in the best-case scenario. A similar argument holds for the example in Figure 11(c). As can be seen, blocking occurs in transportation systems, and it increases waiting time and wastes energy. □

In this section, we analyze the effect of the blocking problem to waiting time (which is related to energy consumption) at the intersections of transportation systems. We assume connected vehicles heterogeneously communicate their intentions to the traffic lights which further send this information to the cloud. Thus, a cloud-based system, based on the intention information, can calculate waiting times at intersections. This calculation would be deterministic if all vehicles could communicate their intentions. However, this is not possible today as only a percentage of devices may have such communication infrastructure. Even in a futuristic setup, where all devices have ability of communicating their intentions, some people may prefer not to share this information due to privacy concerns. Thus, we consider a hybrid system model, where arbitrary number of devices can communicate their intentions. In this setup, our goal is to (i) calculate average waiting time by taking into account the blocking problem, and (ii) find shortest routes/paths in terms of waiting times. The following are the key contributions of this work:

- We investigate the impact of blocking problem in transportation systems by modeling arriving and departing vehicles at an intersection as a queuing model. In particular, we investigate two queuing models; single-lane model in Figure 11(a) and one+two lane model in Figure 11(b). For each model, we characterize average waiting times by taking into account the vehicles that can communicate their intentions (to turn left, right, or go straight) and blocking probability.
- We design an algorithm that finds the routes (or set of intersections) between a starting and ending points with shortest delay. The shortest delay algorithm that we design takes into account the average waiting times at intersections, hence blocking probabilities.
- We evaluate our algorithm via simulations for a multiple-intersection transportation network. The simulation results confirm our analysis, and show that our shortest delay algorithm significantly improves over blocking-unaware schemes.

2.2.2 Related Work

Analyzing waiting times and modeling transportation systems using queueing theory have a long history (more than 50 years) [33]. *E.g.,* [24], [34], [35] considered one-lane queues and calculated the expected queue length and arrivals using probability generation functions. These models focus on fixed-cycle traffic signals, and they calculate the steady-state delays and queue lengths under the assumption that the arriving process does not change over time [27]. Time-dependent arrivals have also been considered in [43], [44], [45], [46], [30], [31], [29], [32], [25], [28]. Different modeling strategies are also studied; such as the queuing network model [36], cell transmission model [37], store-and-forward [10], and petri-nets [38]. As compared to this line of work, our work considers (i) that some vehicles can communicate their intentions, which affects the delay analysis, and (ii) blocking problem.

Recently, with the development of sensor technology, there is an increasing interest in terms of estimating the average delay and queue length at isolated intersections using the information collected by probing vehicles. The queue length estimation and estimation error are characterized in [47], [48], where the probing vehicles can provide their location and entering and departure time in the road intersection to a central controller. Isolated intersection is considered in [27], and optimal traffic cycle to adaptively serve two directions of vehicles are calculated. An optimization scheme is proposed in [49] by taking into account stochastic features of traffic flows in isolated intersections. As compared to this line of work, we focus on the impact of blocking problem to waiting times in transportation systems.

There is also an increasing interest in terms of controlling transportation networks by analyzing the network as a whole without focusing on specific intersections. For example, the control of a network of signalized intersections is discussed in [9], with the goal of stabilizing waiting times at intersections by considering the network as a whole and directing vehicles to appropriate intersections. However, this line of work does not take into account the blocking probability and its impact to the waiting times and queue sizes.

2.2.3 System Model

We consider a transportation network consisting of multiple intersections. First, we will focus on an intersection as exemplified in Figure 10 as isolated from the rest of the network, and characterize waiting times by taking into account blocking problem. Then, we will consider multiple intersections in a network, and determine the shortest delay routes/paths.

In our setup, we model each intersection as a set of queues. For example, in Figure 10, there are four queues for each direction (for south-north, north-south, west-east, and east-west bounds). We specifically focus one direction in our model; *e.g.,* south-north bound in Figure 10, and model it using two models: `Queue I`, which is one-lane model shown in Figure 11(a) and `Queue II`; which is a one+two lane model shown in Figure 11(b). Thus, we are using the same queuing models as illustrated in Section 2.1.3.

We consider a time-slotted system, where one vehicle is allowed to leave the queue during one time slot. At each slot, the arrivals of vehicles are distributed according to Poisson distribution. In this setup, when a vehicle enters a queue, it can communicate with the intersection and inform its intention in terms of turning left or right or continuing straight. We call this “communication ability” of the vehicle and, we call the probability that a vehicle has communication ability is “communication probability”. *I.e.,* each vehicle can communicate with the intersection (actually with the sensor possibly attached to traffic lights) with some “communication probability”. In this setup, if vehicles at an intersection have communication abilities, then the traffic light can arrange its phases accordingly, so *blocking is avoided.* Otherwise, the traffic light selects a phase randomly, and *blocking may occur.*

Now, let us explain our queuing models; `Queue I` and `Queue II` in detail.

2.2.3.1 `Queue I`

`Queue I` is a queuing model illustrated in Figure 11(a). In this setup, λ_L, λ_S, and λ_R correspond to arrival rates of vehicles that would like to turn left, continue straight, and turn right. However, for simplicity, we will assume in the rest of the chapter that $\lambda_1 = \lambda_L$, $\lambda_2 = \lambda_S$, and $\lambda_3 = \lambda_R$.

In this setup, if the head-of-line (HoL) vehicle at time slot t has communication ability, then there is no blocking in slot t. However, if the HoL vehicle does not have communication ability, traffic phases for left, straight, and right are "ON" with probability p_1, p_2, and p_3, respectively. Thus, if HoL vehicle would like to turn left, but traffic phase turn right is "ON", then blocking occurs. We consider in our model that p_i $(i = 1, 2, 3)$ is pre-determined and independent and identically distributed (i.i.d) over time slots.

In this setup, arrival traffic is Poisson and departure is any general traffic. Thus, we can model this queue according to M/G/1 queue. However, the analysis is not straightforward due to blocking effect. We present the details in Section 2.2.4.

2.2.3.2 `Queue II`

Our second queuing model is `Queue II`, which is shown in Figure 11(b). Similar to our analysis in Section 2.1.3, to demonstrate the analysis in a simple way, we consider that the right-turning and continuing-straight traffics are combined together. Thus, we consider the arrival rates as $\lambda_1 = \lambda_R + \lambda_S$ and $\lambda_2 = \lambda_L$. And we call both right-turning and straight-continuing vehicles as going-straight vehicles. The simplified model is the same as Figure 3(b) in Section 2.1.3, which is replotted here as shown in Figure 12.

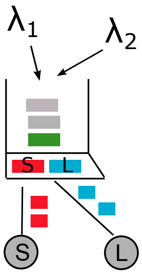


Figure 12. The queuing model with two vehicles in the head-of-line (HoL). The arrival rates of right-turning and straight-continuing vehicles are merged as λ_1 and the arrival rate of left-turning vehicle is λ_2.

In this model, there are two dedicated lanes for vehicles at the head-of-line (HoL), where left-turning vehicles can only join the left lane, while the right-turning or straight-continuing vehicles can only join the going straight lane.

In this setup, at any phase, at least one vehicle is transmitted. However, when we consider traffic phases lasting for two slots, either one (if blocking occurs) or two (if there is no blocking) vehicles are transmitted. In this model, we consider the case that traffic phases change at every two slots, and provide analysis for this setup. Similar to `Queue I`, our goal is to characterize the average waiting time of this model using M/G/1 queues. The details are provided in the next section.

2.2.4 Average Waiting Time Analysis

In this section, we characterize the average waiting time for `Queue I` and `Queue II`. Both queuing models can be modeled as M/G/1 queues. Thus, the average waiting times should follow the Pollaczek-Khinchine (PK) formula:

$$W = \frac{\sum_{i=1}^{3} \lambda_i E(x^2)}{2(1-\rho)} \tag{2.11}$$

where W is the average waiting time, x is the service time and ρ is the line utilization, $\rho = \sum\limits_{i=1}^{3} \lambda_i E(x)$. However, in this formula, it is not straightforward to characterize the expected service time $E(x)$ and the second moment of the service time $E(x^2)$ due to different communication abilities of vehicles and blocking. Thus, our main contribution in this section is the characterization of these parameters ($E(x)$ and $E(x^2)$), which we discuss next.

2.2.4.1 Average Waiting Time for `Queue I`

In `Queue I`, if there is a mismatch between the traffic phases and HoL vehicle, then blocking occurs. For example, if HoL vehicle would like to turn left, but traffic phase turn right is "ON", then blocking occurs. This kind of blocking occurs only if HoL vehicle does not have communication ability. Thus, we discuss three scenarios with different levels of communication abilities; (i) all vehicles have communication abilities, (ii) none of the vehicles have communication abilities, (iii) a percentage of vehicles have communication abilities. Note that the first two scenarios are actually the special cases of the third scenario. However, we present the average waiting time analysis for these scenarios as well in the following to better explain our approach.

2.2.4.1.1 All Vehicles Have Communication Abilities

Since every vehicle has communication ability, the traffic light always knows where the HoL vehicle would like to go. Thus, it can arrange the traffic phase accordingly, so the service time x in Equation 2.11 will be 1. Thus, $E(x) = 1$ and $E(x^2) = 1$, and Equation 2.11 is expressed as

$$W = \frac{\sum\limits_{i=1}^{3} \lambda_i E(x^2)}{2(1-\rho)} = \frac{\sum\limits_{i=1}^{3} \lambda_i}{2(1 - \sum_{i=1}^{3} \lambda_i)} \tag{2.12}$$

where the total arrival rate to the queue is $\sum_{i=1}^{3} \lambda_i$. Note that this is a trivial scenario without any blocking. Next, we consider the case that none of the vehicles have communication abilities.

2.2.4.1.2 None of the Vehicles Have Communication Abilities

If none of the vehicles have communication abilities, then the traffic lights randomly choose phases for left, straight or right with probabilities; p_1, p_2, p_3, respectively.

In this setup, the HoL vehicle can take more than one slots to pass the intersection. Therefore, the service time x can be any positive integer. If the service time $x = n, \forall n \in \mathbb{R}^+$, this means that there has been a mismatch between the HoL vehicle and the traffic phases in the last $n-1$ slots. Therefore, the probability that the vehicle pass the intersection at the nth slot is $P[x = n]$, and it follows the geometric distribution

$$P[x = n] = \sum_{i=1}^{3} \alpha_i (1 - p_i)^{n-1} p_i \tag{2.13}$$

where α_i, $(i = 1, 2, 3)$ is the probability that the HoL vehicle is going to left, straight or right. It is straightforward to see that $\alpha_i = \lambda_i / \sum\limits_{i=1}^{3} \lambda_i$. Thus, Equation 2.13 is expressed as

$$P[x = n] = \frac{\sum_{i=1}^{3} \lambda_i (1 - p_i)^{n-1} p_i}{\sum_{i=1}^{3} \lambda_i} \tag{2.14}$$

Using Equation 2.14, we can calculate $E(x)$ and $E(x^2)$, the expected service time; $E(x)$ is obtained as

$$\begin{aligned} E(x) &= \sum_{n=1}^{\infty} n P[x = n] && (2.15) \\ &= \frac{\sum_{i=1}^{3} \lambda_i p_i \sum_{n=1}^{\infty} n (1 - p_i)^{n-1}}{\sum_{i=1}^{3} \lambda_i} && (2.16) \end{aligned}$$

where $\sum_{n=1}^{\infty} n(1-p_i)^{n-1}$ is expressed as $\sum_{n=1}^{\infty} \frac{\partial(-(1-p_i)^n)}{\partial p_i} = \frac{\partial(\sum_{n=1}^{\infty}(-(1-p_i)^n))}{\partial p_i} = (-\frac{\partial(1-p_i)/p_i}{\partial p_i}) = \frac{1}{p_i^2}$. Thus, Equation 2.16 is expressed as

$$E(x) = \frac{\sum_{i=1}^{3} \lambda_i p_i \frac{1}{p_i^2}}{\sum_{i=1}^{3} \lambda_i} = \frac{\sum_{i=1}^{3} \frac{\lambda_i}{p_i}}{\sum_{i=1}^{3} \lambda_i} \tag{2.17}$$

The line utilization is calculated as

$$\rho = \sum_{i=1}^{3} \lambda_i E(x) = \sum_{i=1}^{3} \frac{\lambda_i}{p_i} \tag{2.18}$$

The second moment of the service time; $E(x^2)$ is calculated as

$$\begin{aligned} E(x^2) &= \sum_{n=1}^{\infty} \frac{n^2 \sum_{i=1}^{3} \lambda_i (1-p_i)^{n-1} p_i}{\sum_{i=1}^{3} \lambda_i} && (2.19) \\ &= \frac{\sum_{i=1}^{3} \lambda_i p_i \sum_{n=1}^{\infty} n^2 (1-p_i)^{n-1}}{\sum_{i=1}^{3} \lambda_i} && (2.20) \end{aligned}$$

where $\sum_{n=1}^{\infty} n^2(1-p_i)^{n-1}$ is expressed as $(2-p_i)/p_i^3$. Thus, Equation 2.20 is expressed as

$$E(x^2) = \frac{\sum_{i=1}^{3} \lambda_i p_i \frac{2-p_i}{p_i^3}}{\sum_{i=1}^{3} \lambda_i} = \frac{\sum_{i=1}^{3} \lambda_i \frac{2-p_i}{p_i^2}}{\sum_{i=1}^{3} \lambda_i} \tag{2.21}$$

Finally, by using PK formula in Equation 2.11, we can obtain the average waiting time as

$$W = \frac{\sum\limits_{i=1}^{3} \lambda_i E(x^2)}{2(1-\rho)} = \frac{\sum\limits_{i=1}^{3} \lambda_i \frac{2-p_i}{p_i^2}}{2(1-\sum_{i=1}^{3} \frac{\lambda_i}{p_i})}, \tag{2.22}$$

Note that the average waiting time for this scenario directly depends on the traffic phase probabilities. This is intuitive as the mismatch between vehicles and traffic lights increases the waiting time, which directly affects the average waiting time.

2.2.4.1.3 A Percentage of Vehicles Have Communication Abilities

Now, we assume that a percentage of vehicles has communication abilities. Let p_t denotes the communication probability of a vehicle. In this scenario, when the HoL vehicle has communication ability, then the traffic phases could be arranged accordingly, and the vehicle immediately passes the intersection (*i.e.,* it can take 1 slot to pass the intersection). On the other hand, when the vehicle does not have communication ability, then the traffic phases will be left, straight or right randomly with probabilities p_1, p_2 and p_3, respectively.

In this scenario, the service time $x = 1$ occurs in two cases. The first case is that the HoL vehicle has communication ability. The second case is that the HoL vehicle does not have communication ability, but the traffic phase is aligned with the direction of the vehicle in the first slot (note that it may take longer). The probability of the second case is $(1 - p_t) \sum\limits_{i=1}^{3} \alpha_i p_i = (1 - p_t) \frac{\sum_{i=1}^{3} \lambda_i p_i}{\sum_{i=1}^{3} \lambda_i}$. Thus, we can calculate the probability that the service time is equal to 1 as

$$P[x = 1] = p_t + (1 - p_t) \frac{\sum_{i=1}^{3} \lambda_i p_i}{\sum_{i=1}^{3} \lambda_i} \tag{2.23}$$

If the service time is larger than 1, we know that this only happens when the HoL vehicle does not have communication abilities. Thus, the calculation for $x > 1$ case is similar to the scenario that none of the vehicles have communication abilities. Thus, the probability of service time to be $n(n \geq 2)$ is

$$P[x = n] = (1 - p_t)\frac{\sum_{i=1}^{3} \lambda_i (1 - p_i)^{n-1} p_i}{\sum_{i=1}^{3} \lambda_i} \tag{2.24}$$

Using $P[x = n]$ in Equation 2.24, we can calculate $E(x)$ and $E(x^2)$. The expected service time can be obtained by

$$E(x) = p_t + (1 - p_t)\sum_{n=1}^{\infty} \frac{n \sum_{i=1}^{3} \lambda_i (1 - p_i)^{n-1} p_i}{\sum_{i=1}^{3} \lambda_i} \tag{2.25}$$

Using the result in calculation for $E(x)$ in section 2.2.4.1.2, we can obtain

$$E(x) = p_t + (1 - p_t)\frac{\sum_{i=1}^{3} \frac{\lambda_i}{p_i}}{\sum_{i=1}^{3} \lambda_i} \tag{2.26}$$

Thus, the line utilization is

$$\rho = \sum_{i=1}^{3} (\lambda_i p_t + (1 - p_t)\frac{\lambda_i}{p_i}) \tag{2.27}$$

In addition, the calculation of the second moment of the service time is similar to that in section 2.2.4.1.2.

$$E(x^2) = p_t + (1 - p_t)\sum_{n=1}^{\infty} \frac{n^2 \sum_{i=1}^{3} \lambda_i (1 - p_i)^{n-1} p_i}{\sum_{i=1}^{3} \lambda_i} \tag{2.28}$$

Using the result in calculation for $E(x^2)$ in section 2.2.4.1.2, we can obtain

$$E(x^2) = p_t + \frac{1-p_t}{\sum_{i=1}^{3}\lambda_i}\sum_{i=1}^{3}\lambda_i\frac{2-p_i}{p_i^2} \tag{2.29}$$

Finally, by using PK formula in Equation 2.11, we can obtain the average waiting time as

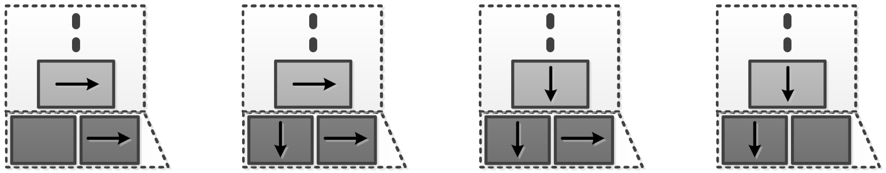
$$W = \frac{\sum_{i=1}^{3}\left[\lambda_i p_t + (1-p_t)\frac{\lambda_i(2-p_i)}{p_i^2}\right]}{2\left\{1-\sum_{i=1}^{3}\left[\lambda_i p_t + (1-p_t)\frac{\lambda_i}{p_i}\right]\right\}} \tag{2.30}$$

2.2.4.2 Average Waiting Time for `Queue II`

Now, we consider the characterization of the average waiting time for `Queue II` shown in Figure 12. In this model, we consider that traffic phases change at every two slots. Thus, in this two-slot duration, one vehicle passes if there is blocking, and two vehicles can pass if there is no blocking. Similar to `Queue I`, we consider different levels of communication abilities for vehicles; (i) all vehicles have communication abilities, (ii) none of the vehicles have communication abilities, and (iii) a percentage of vehicles has communication abilities. Next, we present our analysis for each scenario.

2.2.4.2.1 All Vehicles Have Communication Abilities

If every vehicle has communication abilities, two vehicles can pass at every two-slot duration. Thus, the service time for vehicles is always 1. Therefore, the average waiting time will be the same as in Equation 2.12.



(a) Configuration I (b) Configuration II (c) Configuration III (d) Configuration IV

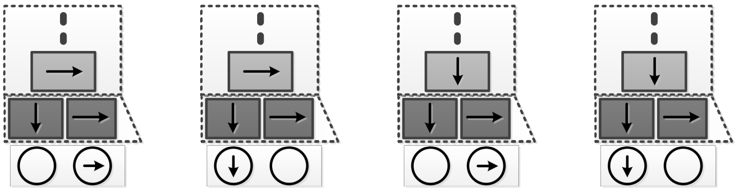
Figure 13. Four possible configurations for the first three vehicles in `Queue II`.

2.2.4.2.2 None of the Vehicles Have Communication Abilities

When none of the vehicles have communication abilities, traffic phases can be arranged depending on the first three vehicle configurations in the queue. Thus, we need to consider the four possible configurations of the first three vehicles as shown in Figure 5 in Section 2.1.3, which is replotted here in Figure 13.

When the first three vehicles in the HoL are in the form of Figure 13(a) or Figure 13(d), *i.e.,* where one of the HoL positions is empty, we assume that the traffic light can sense whether this location is empty or not. As long as it senses an empty location, *e.g.,* an empty left-turning HoL location, it can be estimated that the only possible configuration is Configuration IV in Figure 13(d). Then, the traffic light can arrange its phase to align with the configuration. On the other hand, if the configurations are Configuration II (Figure 13(b)) or Configuration III (Figure 13(c)), the traffic light will randomly arrange its phases.

Based on the discussion above, we know that the service time can be 1 when there is an empty HoL location or the traffic phase allows two vehicles go within two slots when there are no empty HoL locations. Since the service time can be 1 or 2 when there are no empty HoL locations, to calculate the



(a) Conf. II: Left-turn phase ON

(b) Conf. II: Go-straight phase ON

(c) Conf. III: Left-turn phase ON

(d) Conf. III: Go-straight phase ON

Figure 14. Vehicle configurations expansion; Configuration II and Configuration III in Figure 13 for different traffic phases. (a) Configuration II: Left turn phase is ON. (b) Configuration II: Go straight phase is ON. (c) Configuration III: Left turn phase is ON. (d) Configuration III: Go straight phase is ON.

expected service time in this case, we can expand Figure 13(b) and Figure 13(c) with traffic light status to make the calculation more efficient. The expansion is illustrated in Figure 14.

Now, it is clear that the service time will be 1 when the HoL vehicles and traffic light are in the form of Figure 13(a), Figure 13(d), Figure 14(a), or Figure 14(d), and will be 2 when they are in the form Figure 14(b) or Figure 14(c).

By taking into account all possible configurations in Figure 13 and Figure 14, we can obtain six states of the system. Let S_1, S_2, S_3, S_4, S_5 and S_6 denote these six states. Thus, we can use a Markov chain to calculate the stationary probability of the six states and accordingly, the probability distribution of the service time. The Markov chain is shown in Figure 15.

Now, let x be the random variable representing service time. At stationary state, it is clear that $P[x = 2] = P[S_3] + P[S_4]$, where $P[S_i](i = 1, \ldots, 6)$ is the stationary probability that the system is at state S_i. Since the service time can only be 1 or 2, then $P[x = 1] = 1 - P[x = 2]$. Solving the balance equations of the Markov chain in Figure 15, we can obtain

$$P[x = 2] = \frac{2\lambda_1\lambda_2(\lambda_1 p_2 + \lambda_2 p_1)}{(\lambda_1 + \lambda_2)(\lambda_1^2 p_2 + \lambda_2^2 p_1 + 4\lambda_1\lambda_2)} \tag{2.31}$$

Hence

$$E(x) = 1 + \frac{2\lambda_1\lambda_2(\lambda_1 p_2 + \lambda_2 p_1)}{(\lambda_1 + \lambda_2)(\lambda_1^2 p_2 + \lambda_2^2 p_1 + 4\lambda_1\lambda_2)} \tag{2.32}$$

$$E(x^2) = 1 + \frac{6\lambda_1\lambda_2(\lambda_1 p_2 + \lambda_2 p_1)}{(\lambda_1 + \lambda_2)(\lambda_1^2 p_2 + \lambda_2^2 p_1 + 4\lambda_1\lambda_2)} \tag{2.33}$$

Therefore, by using PK formula in Equation 2.11, we can obtain the average waiting time as

$$W = \frac{(\lambda_1 + \lambda_2) + \frac{6\lambda_1\lambda_2(\lambda_1 p_2 + \lambda_2 p_1)}{\lambda_1^2 p_2 + \lambda_2^2 p_1 + 4\lambda_1\lambda_2}}{2\left[1 - (\lambda_1 + \lambda_2) - \frac{2\lambda_1\lambda_2(\lambda_1 p_2 + \lambda_2 p_1)}{\lambda_1^2 p_2 + \lambda_2^2 p_1 + 4\lambda_1\lambda_2}\right]} \tag{2.34}$$

2.2.4.2.3 A Percentage of Vehicles Have Communication Abilities

Now, we assume that the probability of a vehicle has a communication ability is p_t. A major difference in `Queue II` as compared to `Queue I` is that we have dedicated lanes for left-turning and straight-

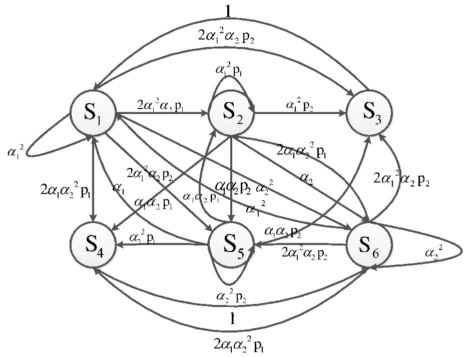


Figure 15. The Markov chain for states S_i ($i = 1, \ldots, 6$), where α_i ($i = 1, 2$) denotes the probability that the HoL vehicle is going straight or left and p_i ($i = 1, 2$) denotes the probability that the traffic light choose go-straight or turn-left phases.

going vehicles in `Queue II`. Since we assume that traffic lights can sense whether a dedicated lane (HoL location) is empty or not, then the first two vehicles in HoL locations will indirectly communicate their intentions. *i.e.,* even if a vehicle does not communicate their intention, when it goes to turn-left lane, then it can be sensed by the traffic light, and its intention can be inferred.

Thus, it only matters whether the third vehicle (the vehicle right behind the HoL locations) has communication ability or not. Note that if the third vehicle has communication ability, then the service time will always be 1 (*i.e.,* 2 vehicles can pass in two slots) since traffic phases can be arranged to match the configurations of the first three vehicles. If the third vehicle does not have communication ability, then the probability distribution of service times will be exactly the same as that in section 2.2.4.2.2.

Therefore, the service time becomes 2 when the third vehicle does not have communication ability and the traffic phase does not match its intention. Similar to the analysis in Section 2.2.4.2.2, we have

$$W = \frac{(\lambda_1 + \lambda_2) + \frac{6(1-p_t)\lambda_1\lambda_2(\lambda_1 p_2+\lambda_2 p_1)}{\lambda_1^2 p_2+\lambda_2^2 p_1+4\lambda_1\lambda_2}}{2\left[1 - (\lambda_1 + \lambda_2) - \frac{2(1-p_t)\lambda_1\lambda_2(\lambda_1 p_2+\lambda_2 p_1)}{\lambda_1^2 p_2+\lambda_2^2 p_1+4\lambda_1\lambda_2}\right]} \tag{2.35}$$

2.2.5 Shortest Delay Algorithm

In this section, we consider a transportation system, which consists of multiple intersections and roads. In this setup, by using the average waiting time analysis in Section 2.2.4, we develop a shortest delay algorithm.

In particular, we consider the transportation network as a weighted directed graph G with N nodes and M edges, each node represents an intersection and each edge represents a road that connects two intersections. Let $V(G)$, $E(G)$ denotes the set of nodes and edges respectively. Then $|V(G)| = N$, $|E(G)| = M$. The weight of each edge is the sum of the average waiting time in the arrival node (intersection) and the traveling time between the two nodes (intersections) on that edge (road). Given the starting and destination nodes in the graph, the goal of our algorithm is to find a path that returns the shortest waiting time (*i.e.,* shortest delay path). The shortest delay algorithm is expressed more specifically in the following.

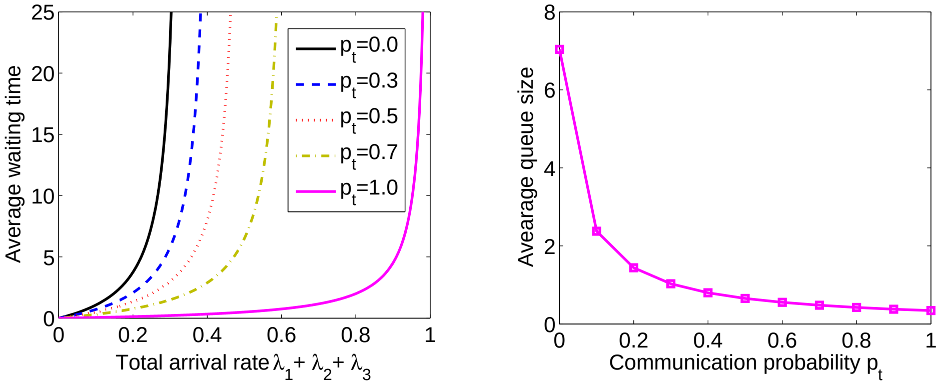
- Given the arrival rate of traffic flows into each node $n \in V(G)$ and communication probability of vehicles arriving into that node, we can calculate the average waiting time W_n using the waiting time analysis in Section 2.2.4. For `Queue I`, W_n is Equation 2.30 and for `Queue II`, W_n is Equation 2.35.

- We assume that vehicles travel at a steady speed of s over an edge between two nodes before arriving into the queue/intersection. Thus, the traveling time between two nodes t_m ($m \in E(G)$) can be obtained by dividing the length of the edge L_m by the speed s, *i.e.,* $t_m = L_m/s$. Then, the total traveling time T_m is the sum of W_n and t_m, *i.e.,* $T_m = W_n + t_m$, and T_m is assigned as the weight of edge m.
- Finally, given the starting and ending nodes, we run Dijkstra's algorithm and return the shortest path it finds. Since the weight of each edge is the total traveling time on that edge, the shortest path we obtain in this way becomes the shortest delay algorithm.

Note that our shortest delay algorithm finds the routes with the shortest average waiting times and less blocking probabilities, so it provides *blocking avoidance* over transportation networks. We note that in multiple-intersection transportation networks (with vehicles making decisions based on estimated waiting times), arrival rates of vehicles would be more generic than Poisson arrivals. In this sense, our algorithm in this section is an approximation. In the next section, we show the effectiveness of our algorithm.

2.2.6 Performance Evaluation

In this section, we first consider isolated intersections, and evaluate our waiting time analysis provided in Section 2.2.4. Then, we consider a larger transportation network with multiple intersections, and evaluate our shortest delay algorithm developed in Section 2.2.5.



(a) Waiting time vs arrival rate

(b) Queue Size vs p_t

Figure 16. `Queue I`. (a) Average waiting time for different total arrival rates with different communication probabilities p_t. (b) Average queue size versus communication probability p_t. The arrival rates are $\lambda_1 = \lambda_2 = \lambda_3 = 0.1$.

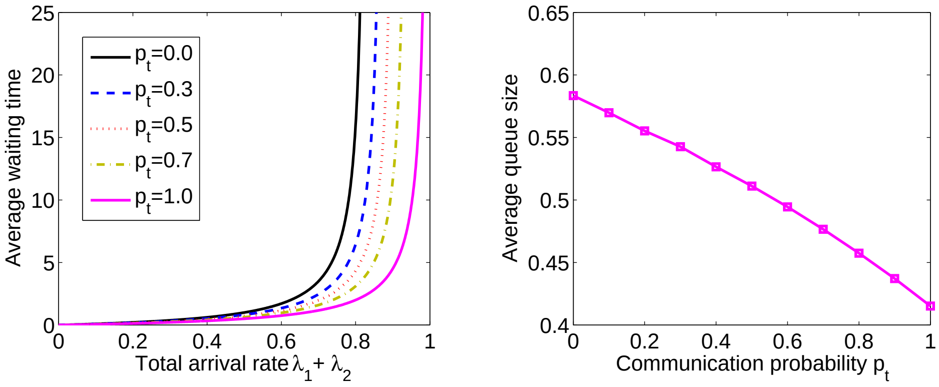
2.2.6.1 Evaluation of Average Waiting Time at Isolated Intersections

2.2.6.1.1 `Queue I`

The simulated average waiting time for `Queue I` versus total arrival rate is shown in Figure 16(a). We assume that $\lambda_1 = \lambda_2 = \lambda_3$.

It can be observed from Figure 16(a) that the average waiting time increases as the total arrival rate increases. This is because the congestion becomes worse as more vehicles enter the queue. Meanwhile, for a fixed total arrival rate, the average waiting time increases as the communication probability p_t decreases, due to the increasing randomness of the traffic signal.

Figure 16(b) shows the average queue size versus communication probability p_t over 10,000 time slots. It can be observed that the average queue size decreases as the communication probability p_t



(a) Waiting time vs arrival rate

(b) Queue Size vs p_t

Figure 17. `Queue II`. (a) Average waiting time for different total arrival rates with different communication probabilities p_t. (b) Average queue size versus communication probability p_t. The arrival rates are $\lambda_1 = \lambda_2 = 0.15$.

increases. It is expected as when p_t increases blocking probability reduces. This shows it is very important that vehicles communicate their intentions to the traffic light to avoid blocking.

2.2.6.1.2 `Queue II`

The simulated average waiting time for `Queue II` versus total arrival rate is shown in Figure 17(a), where we assume that $\lambda_1 = \lambda_2$.

Figure 17(a) shows the similar relationship between total arrival rate and average waiting time as shown in Figure 16(a). However, it can be observed in Figure 17(a) that in most cases the average waiting time is much less than that in Figure 16(a) for the same total arrival rate and communication probability p_t ($p_t \neq 1$). Only when $p_t = 1$, the average waiting times are the same in Figure 17(a) and

Figure 16(a) for the same total arrival rate. This is because when $p_t \neq 1$, at least one vehicle passes during two time slots in `Queue II`. However, it is possible that no vehicles can pass during several time slots in `Queue I`. Only when $p_t = 1$, the service rate will always be 1 in both `Queue I` and `Queue II`, and their average waiting times are closer to each other.

Figure 17(b) shows the average queue size versus communication probability p_t over 10,000 time slots. In general, when communication probability p_t is fixed, the average queue size in `Queue II` is smaller than that in `Queue I`. In addition, the average queue size in `Queue II` is almost linearly decreases as the communication probability p_t increases.

2.2.6.2 Evaluation of the Shortest Delay Algorithm

In this section, we evaluate our algorithm using an illustrative transportation network as shown in Figure 18. In this network, there are four nodes, where node 1 and 4 are the starting and ending nodes, while node 2 and 3 are two intermediate intersections/nodes . The total arrival rate to node 2 and node 3 is λ_{n2} and λ_{n3} respectively. Assuming that a percentage of vehicles has communication abilities at node 2, and none of the vehicles have communication abilities at node 3, we evaluate the estimated end-to-end traveling time (or delay). Next, we briefly describe our baselines.

2.2.6.2.1 Baselines

We consider a baseline algorithm that uses the same method as our algorithm except that it does not take into account the communication probability of vehicles. In other words, none of the vehicles can communicate their intentions in the baseline algorithm. Thus, the evaluation of our shortest delay

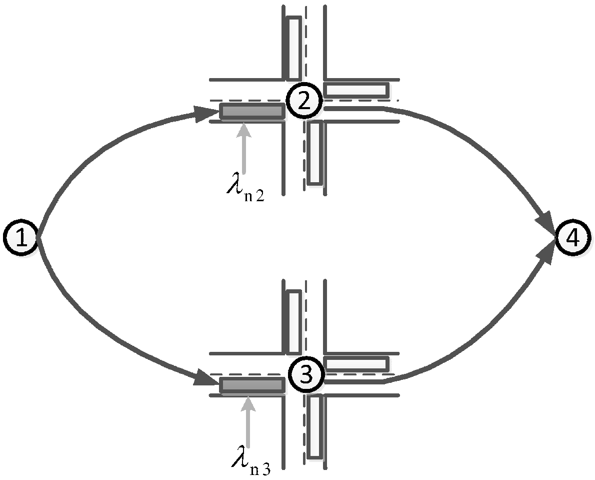
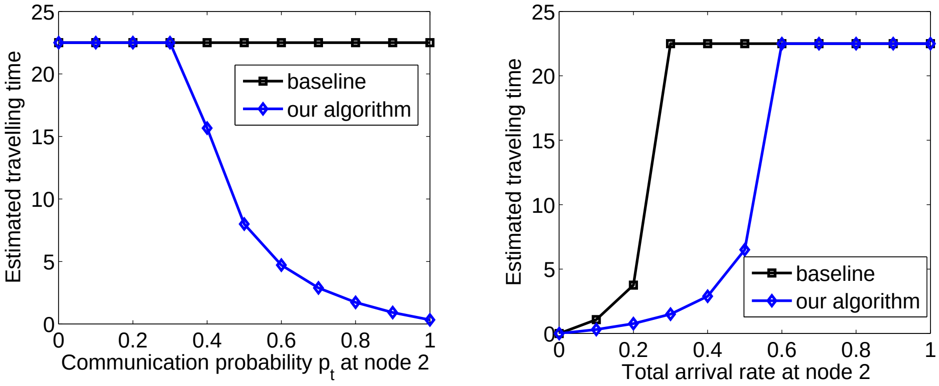


Figure 18. An illustrative transportation network with four nodes. The total arrival rate to node 2 and node 3 is λ_{n2} and λ_{n3}, respectively.

algorithm as compared to the baseline will show the benefit communication ability to overall delay, and avoiding blocking. The baseline algorithm is summarized briefly in the following.

- Given the arrival rate into each node, we calculate the average waiting time W_n in each node without considering the communication ability of vehicles. For `Queue I`, W_n is Equation 2.22 and for `Queue II`, W_n is Equation 2.34.
- Similar to our shortest delay algorithm, the total traveling time T_m on road/edge m is the sum of W_n and t_m, *i.e.*, $T_m = W + t_m$, and $T_m$is assigned as the weight of edge m.
- Given the starting and ending nodes, we run Dijkstra's algorithm and return the shortest path it finds.



(a) Traveling time vs p_t at node 2

(b) Traveling time vs λ_{n2}

Figure 19. `Queue I`. (a) Estimated traveling time vs communication probability p_t at node 2. (b) Estimated traveling time vs arrival rate at node 2.

Next, we present our simulation results for `Queue I` and `Queue II`. In the transportation network in Figure 18, we know that there are only two paths from node 1 to 4; they are $1 \to 2 \to 4$ and $1 \to 3 \to 4$ as shown in bold directed lines in Figure 18. To clearly see the effect of blocking and waiting times at intersections/queues, we assume that the four road segments in Figure 18 have the same length. Thus, without violating generality, we assume that the traveling times over each road/link is 0, *i.e.*, both in the baseline and our algorithm $t_m = 0$ $(m = 1, 2, 3, 4)$.

2.2.6.2.2 `Queue I`

In this section, we consider that all the queuing model in the intersections of the road network follow `Queue I`, which is shown in Figure 11(a).

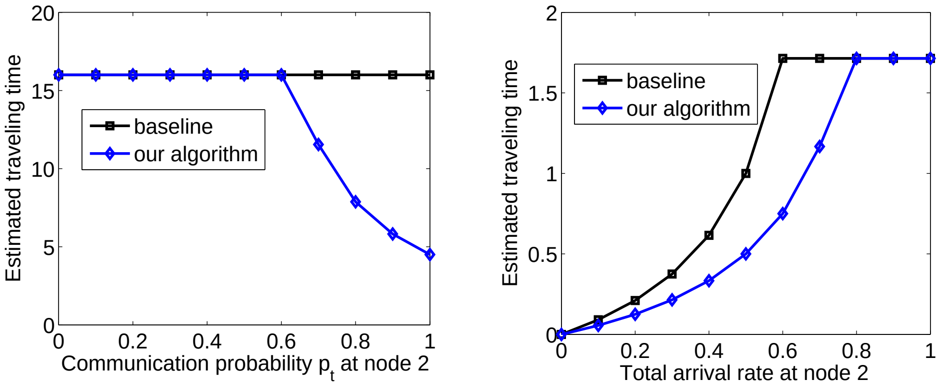
We assume that the arrival rates to node 2 are $\lambda_1 = \lambda_2 = \lambda_3 = 0.4/3$, and the arrival rates to node 3 are $\lambda_1 = \lambda_2 = \lambda_3 = 0.1$. Figure 19(a) presents the traveling time versus communication probability p_t at node 2. Since the baseline algorithm does not take into account the communication ability of vehicles, it will choose the path $1 \rightarrow 3 \rightarrow 4$ as the shortest path since the total arrival rate at node 3 is smaller than node 2. However, if we take into account the communication ability of vehicles (*i.e.,* if we use our algorithm), Figure 19(a) shows that the traveling time will decrease as our algorithm chooses the path $1 \rightarrow 2 \rightarrow 4$ instead of $1 \rightarrow 3 \rightarrow 4$ when the communication probability p_t at node 2 is larger than 0.3.

Let us assume that the arrival rates to node 3 are $\lambda_1 = \lambda_2 = \lambda_3 = 0.1$, and the communication probability at node 2 is $p_t = 0.7$. Figure 19(b) presents the traveling time versus total arrival rates at node 2 (we assume $\lambda_1 = \lambda_2 = \lambda_3$ at node 2). Again, since the baseline algorithm does not take into account the communication ability of vehicles, it will choose the path $1 \rightarrow 2 \rightarrow 4$ when $\lambda_{n2} \leq \lambda_{n3}$ and switches to path $1 \rightarrow 3 \rightarrow 4$ when $\lambda_{n2} > \lambda_{n3}$. However, our algorithm chooses path $1 \rightarrow 2 \rightarrow 4$ even for $\lambda_{n3} < \lambda_{n2} < 0.6$ because of taking into account the communication ability of vehicles in node 2. Thus, the estimated traveling time using our algorithm is smaller than the baseline for $\lambda_{n2} < 0.6$.

2.2.6.2.3 `Queue II`

In this section, we consider that all the queuing model in the intersections of the transportation network follow `Queue II`, which is shown in Figure 12.

Let us assume that the arrival rates to node 2 is $\lambda_1 = \lambda_2 = 0.45$, and the arrival rates to node 3 is $\lambda_1 = \lambda_2 = 0.4$. Figure 20(a) presents the traveling time versus communication probability p_t at node 2. Similar to `Queue I`, our algorithm improves over the baseline.



(a) Traveling time vs p_t at node 2 (b) Traveling time vs λ_{n2}

Figure 20. `Queue II`. (a) Estimated traveling time vs communication probability p_t at node 2. (b) Estimated traveling time vs arrival rate at node 2.

Let us assume that the arrival rates to node 3 are $\lambda_1 = \lambda_2 = 0.3$, and the communication probability at node 2 is $p_t = 1.0$. Figure 20(b) presents the traveling time versus total arrival rates at node 2 (we assume that $\lambda_1 = \lambda_2$ at node 2). Similar to `Queue I`, our algorithm improves over the baseline.

2.2.7 Conclusion

In this chapter, we investigated the blocking problem which naturally arises in transportation networks, where multiple vehicles with different itineraries share available resources. We characterized waiting times at intersections of transportation systems by taking into account blocking probability as well as the communication probability of vehicles. Then, by using average waiting times at intersection, we developed a shortest delay algorithm that calculates the routes with shortest delays between two points in a transportation network. Our simulation results show that our shortest delay algorithm significantly improves over baselines that are unaware of the blocking problem.

CHAPTER 3

FLOW CONTROL AND SCHEDULING FOR HETEROGENEOUS (PER-FLOW AND FIFO) QUEUES OVER WIRELESS NETWORKS

The contents of this chapters are based on our work that is published in the proceedings of 2016 ITA workshop [3]. ©2016 IEEE. Reprinted, with permission, from [3].

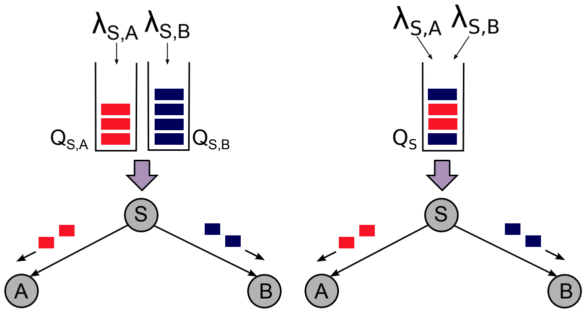
We investigate the performance of wireless networks of devices with heterogeneous (per-flow and FIFO) queues. We consider a general scenario where an arbitrary number of per-flow and FIFO queues, which are served by a wireless medium, are shared by an arbitrary number of flows. In this setup, we formulate the support region, which is characterized by the set of arrival rates that can be stably supported in the network. In general, the support region of this system is non-convex. Thus, we develop a convex inner-bound on the support region, which is provably tight in certain cases. The convexity of the inner bound allows us to develop a resource allocation scheme; dFC. Based on the structure of dFC, we develop a stochastic flow control and scheduling algorithm; qFC. We show that qFC achieves optimal operating point in the convex inner bound. Simulation results show that our algorithms significantly improve the throughput of wireless networks with FIFO queues, as compared to the well-known queue-based flow control and max-weight scheduling.

3.1 Background

The recent growth in mobile and media-rich applications continuously increases the demand for wireless bandwidth, and puts a strain on wireless networks [11], [12]. This dramatic increase in demand poses a challenge for current wireless networks, and calls for new network control mechanisms that make better use of scarce wireless resources. Furthermore, most existing, especially low-cost, wireless devices have a relatively rigid architecture with limited processing power and energy storage capacities that are not compatible with the needs of existing theoretical network control algorithms. One important problem, and the focus of this chapter, is that low-cost wireless interface cards are built using First-In, First-Out (FIFO) queueing structure, which is not compatible with the per-flow queueing requirements of the optimal network control schemes such as backpressure routing and sheduling [13].

The backpressure routing and scheduling paradigm has emerged from the pioneering work [13], [14], which showed that, in wireless networks where nodes route and schedule packets based on queue backlogs, one can stabilize the queues for any feasible traffic. It has also been shown that backpressure can be combined with flow control to provide utility-optimal operation [15]. Yet, backpressure routing and scheduling require each node in the network to construct per-flow queues. The following example demonstrates the operation of backpressure.

Example 4. *Let us consider a canonical example in Figure 21(a), where a transmitter node* S*, and two receiver nodes* A*,* B *form a one-hop downlink topology. There are two flows with arrival rates* $\lambda_{S,A}$ *and* $\lambda_{S,B}$ *destined to nodes* A *and* B*, respectively. The throughput optimal backpressure scheduling scheme, also known as max-weight scheduling, assumes the availability of per-flow queues* $Q_{S,A}$ *and* $Q_{S,B}$ *as seen in Figure 21(a), and makes a transmission decision at each transmission opportunity based on queue backlogs,* i.e., $Q_{S,A}$ *and* $Q_{S,B}$*. In particular, the max-weight scheduling algorithm determines* $F^* = \arg\max_{F \in \{A,B\}} Q_{S,F}$*, and transmits from queue* Q_{S,F^*}*. It was shown in [13], [14] that if the arrival rates* $\lambda_{S,A}$ *and* $\lambda_{S,B}$ *are inside the stability region of the wireless network, the max-weight*



(a) Per-flow queues (b) FIFO queue

Figure 21. Queueing structure of one-hop downlink topology with (a) per-flow queues, and (b) a FIFO queue.

scheduling algorithm stabilizes the queues. On the other hand, in some devices, per-flow queues cannot be constructed. In such a scenario, a FIFO queue, say Q_S *is shared by flows* A *and* B *as shown in Figure 21(b), and the packets are served from* Q_S *in a FIFO manner.* □

FIFO queues are widely used in plenty of wireless communication scenarios due to hardware constraints. For example, constructing per-flow queues may not be feasible in some devices especially at the link layer due to rigid architecture, and one FIFO queue is usually shared by multiple flows [50, 51]. Although current WiFi-based devices have more than one hardware queue [52], their numbers are restricted (up to 12 queues according to the list in [52]), while the number of flows passing through a wireless device could be significantly higher. Also, multiple queues in wireless devices are mainly constructed for prioritized traffic such as voice, video, etc., which further limits their usage as per-flow queues [52]. On the other hand, constructing per-flow queues may not be preferable in some other devices such as sensors or home appliances for which maintaining and handling per-flow queues could introduce too much processing and energy overhead [53]. Thus, some devices, either due to rigid architecture or limited processing power and energy capacities, inevitably use shared FIFO queues, which makes the understanding of the behavior of FIFO queues over wireless networks very crucial.

Example 1 - continued: Let us consider Figure 21 again. When a FIFO queue is used instead of per-flow queues, the well-known head-of-line (HoL) blocking phenomenon occurs. As an example, suppose that at transmission instant t, the links $S - A$ and $S - B$ are at “ON” and “OFF” states, respectively. In this case, a packet from $Q_{S,A}$ can be transmitted if per-flow queues are constructed. Yet, in FIFO case, if HoL packet in Q_S belongs to flow B, no packets can be transmitted and wireless resources are wasted. □

Although HoL blocking in FIFO queues is a well-known problem, achievable throughput with FIFO queues in a wireless network is generally not known. In particular, the network support region, which is characterized by a set of feasible arrival rates that can be stably supported (*i.e.,* not overflowing buffers), as well as the resource allocation schemes to achieve optimal operating point in the support region are still open problems.

In this work, we investigate a general wireless network where per-flow and FIFO queues coexist. We consider a wireless network model presented in Figure 22 with multiple heterogeneous queues that are in the same transmission and interference range.

Our first step towards understanding the performance of per-flow and FIFO queues in such a setup is to characterize the support region of the network. Then, based on the structure of the support region, we develop efficient resource allocation algorithms; *Deterministic FIFO-Control* (dFC) and *Queue-Based FIFO-Control* (qFC). The following are the key contributions of this work:

- We characterize the support region of a general scenario where an arbitrary number of per-flow and FIFO queues are shared by an arbitrary number of flows.
- The support region of the heterogeneous queuing system under investigation is non-convex. Thus, we develop a convex inner-bound on the support region, which is provably tight for certain operating points.
- We develop a resource allocation scheme; dFC, and a queue-based stochastic flow control and scheduling algorithm; qFC. We show that qFC achieves optimal operating point in the convex inner bound.
- We evaluate our schemes via simulations for multiple per-flow and FIFO queues and flows. The simulation results show that our algorithms significantly improve the throughput as compared to the well-known queue-based flow control and max-weight scheduling schemes.

3.2 Related Work

In this work, our goal is to understand FIFO queues in wireless networks and develop efficient flow control and scheduling policies for such a setup. In the seminal paper [54], the authors analyze FIFO queues in an input queued switch. They show that the use of FIFO queues in that context limits the throughput to approximately 58% of the maximum achievable throughput. However, in the context of wireless networks, similar results are in general not known.

Backpressure routing and scheduling framework has emerged from the pioneering work [13, 14], which has generated a lot of research interest [55]; especially for wireless ad-hoc networks [56–61]. Furthermore, it has been shown that backpressure can be combined with flow control to provide utility-optimal operation guarantee [15,60]. Such previous work mainly considered per-flow queues. However, FIFO queueing structure, which is the focus of this thesis, is not compatible with the per-flow queueing requirements of these routing and scheduling schemes.

The strengths of backpressure-based network control have recently received increasing interest in terms of practical implementation. Multi-path TCP scheme is implemented over wireless mesh networks in [62] for routing and scheduling packets using a backpressure based heuristic. At the link layer, [63–65] propose, analyze, and evaluate link layer backpressure-based implementations with queue prioritization and congestion window size adjustment. Backpressure is implemented over sensor networks [66] and wireless multi-hop networks [67]. In these schemes, either last-in, first-out queueing is employed [66] or link layer FIFO queues are strictly controlled [67] to reduce the number of packets in the FIFO queues, hence HoL blocking.

The backpressure routing and scheduling algorithm requires per-flow (or per-destination) information, which may be difficult to obtain and maintain, especially in large multihop networks. There is some work in the literature to stretch this necessity. For example, [68], [69] propose using real per-link and virtual per-flow queues. Such a method reduces the number of queues required in each node, and reduces the delay, but it still needs to construct per-link queues. Similarly, Queue Proportional Rate Allocation (QPRA) scheduling algorithm in [70] is based on per-link queues and allocates resources proportional to queue lengths while achieving maximum throughput. In this line, per link queues operating according to FIFO rule are considered in [71–74]. As compared to this line of work, we consider shared FIFO queues over multiple links, where HoL blocking arises, which does not occur on per-link FIFO queues.

The main differences in our work are: (i) we consider heterogeneous (per-flow and FIFO) queues shared by multiple flows where HoL blocking occurs as each flow is transmitted over a possibly different wireless link, (ii) we characterize the support region of a general scenario where an arbitrary number of per-flow and FIFO queues, which are served by a wireless medium, are shared by an arbitrary number of flows, and (iii) we develop efficient resource allocation schemes to exploit achievable rate in such a setup.

3.3 System Model

Wireless Network Setup: We consider a wireless network model presented in Figure 22 with N FIFO queues. Let $\mathcal{N}$ be the set of FIFO queues, $\mathcal{Q}_n$ be the nth FIFO queue, and $\mathcal{K}_n$ be the set of flows passing through $\mathcal{Q}_n$. Also, let Q_n and K_n denote the cardinalities of sets $\mathcal{Q}_n$ and $\mathcal{K}_n$, respectively. We assume in our analysis that time is slotted, and t refers to the beginning of slot t. Note that this model also covers per-flow queues as a FIFO queue could be considered as a per-flow queue if the number of arrival flows to that queue is one. Therefore, for simplicity of terminology, we call both of per-flow and FIFO queues as FIFO queues in the rest of the thesis.

Flow Rates: Each flow passing through $\mathcal{Q}_n$ and destined for node k is generated according to an arrival process $\lambda_{n,k}(t)$ at time slot t. The arrivals are i.i.d. over the time slots such that for every $n \in \mathcal{N}$ and $k \in \mathcal{K}_n$, we have $\lambda_{n,k} = E[\lambda_{n,k}(t)]$ and $E[\lambda_{n,k}(t)^2] < \infty$, where $E[\cdot]$ denotes the expected value.

Channel Model: In our setup in Figure 22, as we mentioned earlier, we assume that all FIFO queues are in the same transmission and interference range, *i.e.,* only one FIFO queue could be served by a shared wireless medium at time t. On the other hand, a channel state from a FIFO queue to a receiver node may vary. In particular, at slot t, $\boldsymbol{C}(t) = \{C_{n,k}(t)\}_{\forall n \in \mathcal{N}, k \in \mathcal{K}_n}$ is the channel state vector, where

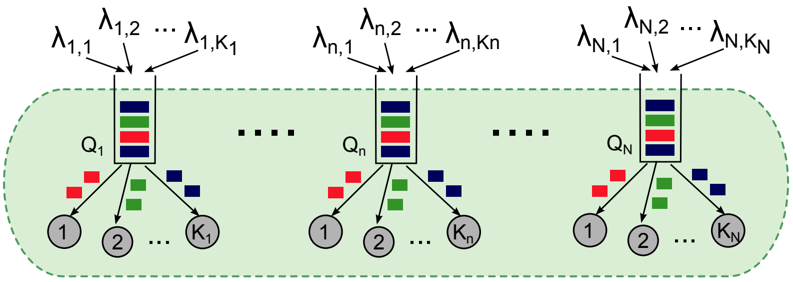

$\lambda_{1,1}$
$\lambda_{1,2}$ ... λ_{1,K_1}
$\lambda_{n,1}$
$\lambda_{n,2}$ ... $\lambda_{n,Kn}$
$\lambda_{N,1}$
$\lambda_{N,2}$ ... λ_{N,K_N}
....
....
Q_1
Q_n
Q_N
1
2
...
K_1
1
2
...
K_n
1
2
...
K_N

Figure 22. The wireless network model that we consider in this thesis. N FIFO queues share a wireless medium, where the nth FIFO queue, $\mathcal{Q}_n$ carries $\mathcal{K}_n$ flows towards their respective receiver nodes. The arrival rate of the kth flow passing through the nth queue is $\lambda_{n,k}$.

$C_{n,k}(t)$ is the state of the link at time t from the nth queue $\mathcal{Q}_n$ to receiver node k such that $k \in \mathcal{K}_n$. The link state $C_{n,k}(t)$ takes values from the set $\{ON, OFF\}$ according to a probability distribution which is i.i.d. over time slots. If $C_{n,k}(t) = ON$, packets can be transmitted to receiver node k with rate $R_{n,k}$. We assume, for the sake of simplicity in this thesis, that $R_{n,k} = 1$, and 1 packet can be transmitted at time slot t if $C_{n,k}(t) = ON$. If $C_{n,k}(t) = OFF$, no packets are transmitted. The ON and OFF probabilities of $C_{n,k}(t)$ are $\bar{p}_{n,k}$ and $p_{n,k}$, respectively. Note that $C_{n,k}(t)$ only determines the channel state; *i.e.,* the actual transmission opportunity from $\mathcal{Q}_n$ depends on the HoL packet as explained next.

Queue Structure and Evolution: Suppose that the Head-of-Line (HoL) packet of $\mathcal{Q}_n$ at time t is $H_n(t)$ and it belongs to one of the flows in $\mathcal{K}_n$. The HoL packet together with the channel state defines the state of $\mathcal{Q}_n$. In particular, let $S_n(t)$ be the state of $\mathcal{Q}_n$ at time t such that $S_n(t) \in \{ON, OFF\}$. The state of $\mathcal{Q}_n$ is ON, *i.e.,* $S_n(t) = ON$ if $C_{n,H_n(t)} = ON$ at time t. Otherwise, $S_n(t) = OFF$. We define $\mathbf{S} = \{S_1, S_2, \cdots, S_N\}$ as the vector of states of all the FIFO queues and $\mathcal{S} = \{\mathbf{S} | S_1, \ldots, S_N \in \{ON, OFF\}\}$ as the set of all the vectors.

Let us now consider the evolution of the HoL packet. If the state of queue $\mathcal{Q}_n$ is ON at time t, *i.e.,* $S_n(t) = ON$, the HoL packet can be transmitted (depending on the scheduling policy). If we assume there are always packets in the queue, after the HoL packet being transmitted, a new packet is placed in the HoL position in $\mathcal{Q}_n$. Let us denote the probability that this new HoL packet belongs to the kth flow as $\alpha_{n,k}$. Since the packets follow Fist-in First-out pattern, $\alpha_{n,k}$ depends on the arrival rates of all the flows into $\mathcal{Q}_n$, that is, $\alpha_{n,k} = \frac{\lambda_{n,k}}{\sum_{k \in \mathcal{K}_n} \lambda_{n,k}}$.

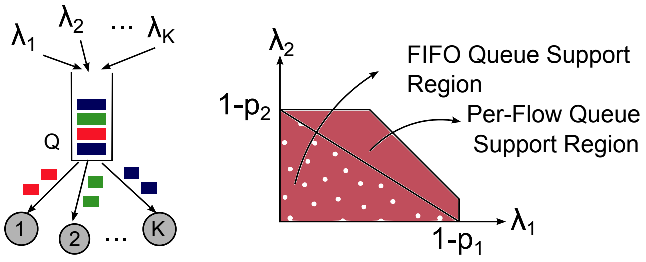
Now, we can consider the evolution of $\mathcal{Q}_n$. At time t, $\sum_{k \in \mathcal{K}_n} \lambda_{n,k}(t)$ packets arrive to $\mathcal{Q}_n$, and $g_n(t)$ packets are served according to the FIFO manner. Thus, queue size $Q_n(t)$ evolves according to the following dynamics.

$$Q_n(t+1) \leq \max[Q_n(t) - g_n(t), 0] + \sum_{k \in \mathcal{K}_n} \lambda_{n,k}(t). \tag{3.1}$$

Note that $g_n(t)$ depends on the states of the queues; $\mathbf{S}(t)$ at time t, which characterize the support region of the wireless network. Note that $\mathbf{S}(t)$ depends on arrival rates of flows to each FIFO queue; *i.e.,* $\lambda_{n,k}$ as well as the ON-OFF probability of each link, *i.e.,* $p_{n,k}$. In the next section, by taking into account $\lambda_{n,k}$ and $p_{n,k}$, we characterize the support region of the wireless network.

3.4 Support Region

In this section, our goal is to characterize the support region of a wireless network where an arbitrary number of FIFO queues are served by a shared wireless medium. As we mentioned earlier, the support region is characterized by the set of arrival rate vectors that can be served by the shared medium without overflowing the queues in the network. We first begin with the single-queue case shown in Figure 23 to



(a) FIFO queue (b) Support region

Figure 23. (a) Single-FIFO queue; $\mathcal{Q}$ is shared by K flows. (b) Support region of a single-FIFO queue as well as per-flow queues with two flows.

convey our approach for a canonical scenario, then we extend our support region analysis for arbitrary number of FIFO queues and flows.

3.4.1 Single-FIFO Queue

We study the special case of a single FIFO queue $\mathcal{Q}_1$ where $\mathcal{N} = \{1\}$ with $N = 1$. For this special case, we thus drop the queue index n from the notation in Section 3.3 for brevity. In other words, we write $\mathcal{Q}$ instead of $\mathcal{Q}_n$, $C_k(t)$ instead of $C_{n,k}(t)$, and so on. Our main result in this context is then the following theorem.

Theorem 3. *For a FIFO queue $\mathcal{Q}$ shared by $\mathcal{K} = \{1, \ldots, K\}$ flows, if the channel states $C_k(t)$ and arrival rates $\lambda_k(t)$ are i.i.d. over time slots, the support region Λ includes all arrival rates satisfying*

$$\sum_{k \in \mathcal{K}} \frac{\lambda_k}{\bar{p}_k} \leq 1. \tag{3.2}$$

In other words, the support region of the single-FIFO queue system is $\Lambda = \{\{\lambda_k\}_{k \in \mathcal{K}} |$ Equation 3.2, $\lambda_k \geq 0, \forall k \in \mathcal{K}\}$.

Proof: Suppose we are given the set of arrival rates that are supported by the network, that is, the total arrival rates are no more than the total service rate of the system, our goal is to prove that these arrival rates satisfy $\sum_{k\in\mathcal{K}} \frac{\lambda_k}{\bar{p}_k} \leq 1$.

The state of the FIFO queue $\mathcal{Q}$ takes values from $\{ON, OFF\}$ depending the HoL packet and the states of the wireless links. Now, let us take a closer look at the FIFO states. The OFF state occurs if for some $k \in \mathcal{K} = \{1, \ldots, K\}$ we have $H = k$ and $C_k = OFF$. Let z_k be the state that $H = k$ and $C_k = OFF$. We denote the probability of z_k as $P[z_k] = P[H = k, C_k = OFF]$. Also, let z_0 be the state that FIFO queue is at ON state for some HoL packet. The state z_0 happens precisely when the channel corresponding to the HoL packet is in the ON state. Therefore, the probability of z_0 is $P[z_0] = P[C_H = ON]$.

Having defined the queue state probabilities, we can observe that the packets from the FIFO queue could be served only at state z_0. It is also clear that the necessary condition that the arrival rates are in the support region is that the sum of the arrival rates to the queue $\mathcal{Q}$ should be less than the service rate, which is $P[z_0]$. Noting that we assumed $R_k = 1$, we conclude that $\sum_{k\in\mathcal{K}} \lambda_k \leq P[z_0]$.

Let us now calculate $P[z_0]$ and $P[z_k]$, $k \in \mathcal{K}$ using a Markov chain with states; z_0 and z_k, $k \in \mathcal{K}$. We first show that the state transition probability from z_0 to z_k is $P_{0,k} \triangleq \alpha_k p_k$, where $\alpha_k = \frac{\lambda_k}{\sum_{k\in\mathcal{K}} \lambda_k}$. Since we consider only one FIFO queue, when the queue is at state z_0, the HoL packet is always transmitted. Suppose there are always packets in the queue, the new HoL packet in the next state will belong to the kth flow with probability α_k, and $C_k = OFF$ with probability p_k. Therefore, the state transition probability from z_0 to z_k is $P_{0,k} = \alpha_k p_k$, as claimed.

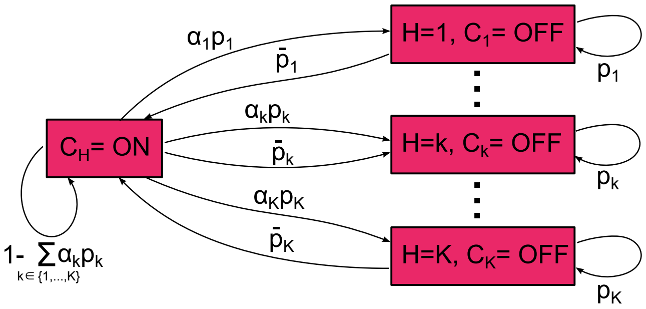


Figure 24. Markov chain for the single-FIFO queue system shown in Figure 23(a).

The probability of moving from state z_k to z_0 is $P_{k,0} \triangleq \bar{p}_k$ as we can move to the unblocking state z_0 from the blocking state z_k if the channel is ON (with probability $\bar{p}_k$.). On the other hand, staying in the blocking state z_k is the OFF probability of the channel C_k. Thus, $P_{k,k} \triangleq p_k$. Note that the expressions for $P_{k,0}$ and $P_{k,k}$ do not involve the quantity α_k. The reason is that z_k is the blocking state, so when we move from z_k to another state (or staying at state z_k), the HoL packet is not transmitted and does not change (because $C_k = OFF$ at state z_k).

For any given k, $l \in \mathcal{K}$ with $k \neq l$, the state transition probability from z_k to z_l is $P_{k,l} \triangleq 0$. This follows since it is not possible to move from a blocking state to another (the HoL packet cannot be transmitted.). Finally, the probability of staying at state z_0 is $P_{0,0} \triangleq 1 - \sum_{k \in \mathcal{K}} \alpha_k p_k$ as the condition $\sum_{k=0}^{K} P_{0,k} = 1$ should be satisfied thanks to the law of total probability. The state transition probabilities are as shown in Figure 24.

Now that we know the state transition probabilities of our Markov chain, we can calculate the balance equations, and these yield $P[z_0] = \frac{\sum_{k \in \mathcal{K}} \lambda_k}{\sum_{k \in \mathcal{K}} \lambda_k / \bar{p}_k}$. The calculations are provided in the following.

Let $P[\boldsymbol{z}] = \begin{bmatrix} P[z_0] & P[z_1] & \dots & P[z_K] \end{bmatrix}^T$. In the steady the state, the following set of equations are satisfied for the Markov Chain shown in Figure 24.

$$P[\boldsymbol{z}]^T \begin{bmatrix} 1-\sum_{k\in\mathcal{K}}\alpha_k p_k & \alpha_1 p_1 & \dots & \alpha_K p_K \\ \bar{p}_1 & p_1 & \dots & 0 \\ \bar{p}_2 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \bar{p}_k & 0 & \dots & p_K \end{bmatrix} = P[\boldsymbol{z}] \tag{3.3}$$

If we combine the $(k+1)$th equation in Equation 3.3, which is $P[z_k] = P[z_0]\frac{\alpha_k p_k}{\bar{p}_k}$, and the fact that $P[z_0] + \sum_{k\in\mathcal{K}} P[z_k] = 1$, we have

$$P[z_0] = \frac{\sum_{k\in\mathcal{K}} \lambda_k}{\sum_{k\in\mathcal{K}} \lambda_k/\bar{p}_k} \tag{3.4}$$

We can then obtain $\sum_{k\in\mathcal{K}} \lambda_k \leq P[z_0] = \frac{\sum_{k\in\mathcal{K}} \lambda_k}{\sum_{k\in\mathcal{K}} \lambda_k/\bar{p}_k}$ which is equivalent to Equation 3.2. This concludes the proof. ■

Example 5. *Now suppose that single-FIFO queue $\mathcal{Q}$ is shared by two flows with rates λ_1 and λ_2. According to Theorem 3, the arrival rates should satisfy $\lambda_1/\bar{p}_1 + \lambda_2/\bar{p}_2 \leq 1$ for stability. This support region is shown in Figure 23(b). In the same figure, we also show the support region of per-flow queues, [55]. As seen, the FIFO support region is smaller as compared to per-flow capacity region. Yet, we still need flow control and scheduling algorithms to achieve the optimal operating point in this support region. This issue will be discussed later in Section 3.5.* □

3.4.2 Arbitrary Number of Queues and Flows

We now consider a wireless network with arbitrary number of FIFO queues and flows as shown in Figure 22. The main challenge in this setup is that packet scheduling decisions affect the support region. For example, if both $\mathcal{Q}_1$ and $\mathcal{Q}_n$ in Figure 22 are at ON state, a decision about which queue to be served should be made. This decision affects future transmission opportunities from the queues, hence the support region.

In this thesis, we consider a scheduling policy where the packet transmission probability of each queue depends only on the queue states. In other words, if the state of the FIFO queues is $\mathbf{S} \in \mathcal{S}$, a packet from queue $\mathcal{Q}_n$ is transmitted with probability $\tau_n(\mathbf{S})$. We call this scheduling policy the *queue-state* policy. Note that as $\tau_n(\mathbf{S})$ is the transmission probability from queue $\mathcal{Q}_n$, we have the obvious constraint

$$\sum_{n \in \mathcal{N}} \tau_n(\mathbf{S}) \leq 1, \forall \mathbf{S} \in \mathcal{S}. \tag{3.5}$$

Our main result is then the following theorem.

Theorem 4. *For a wireless network with N FIFO queues, if a queue-state policy τ_n is employed, then the support region consists of the flow rates that satisfy*

$$\begin{aligned}\lambda_{n,k} \leq \sum_{\mathbf{S}\in\mathcal{S}} \Bigg\{ & \frac{\lambda_{n,k} 1_{[S_n]}}{\sum_{k\in\mathcal{K}_n} \lambda_{n,k}/\bar{p}_{n,k}} \\ & \prod_{m\in\mathcal{N}-\{n\}} \left(\frac{\sum_{k\in\mathcal{K}_m} \lambda_{m,k}\rho_{m,k}(S_m)}{\sum_{k\in\mathcal{K}_m} \lambda_{m,k}/\bar{p}_{m,k}} \right) \tau_n(\mathbf{S}) \Bigg\}, \\ & \forall n \in \mathcal{N}, k \in \mathcal{K}_n, \end{aligned} \tag{3.6}$$

where

$$1_{[S_n]} = \begin{cases} 1, & S_n = ON \\ 0, & S_n = OFF \end{cases},$$

$$\rho_{m,k}(S_m) = \begin{cases} 1, & S_m = ON \\ p_{m,k}/\bar{p}_{m,k}, & S_m = OFF \end{cases}.$$

Proof: In order for the arrival rates $\lambda_{n,k}$ to be supported by the system, we have

$$\lambda_{n,k} \leq \sum_{\mathbf{S}\in\mathcal{S}} P[\mathbf{S}, H_n = k] 1_{[S_n]} \tau_n(\mathbf{S}), \forall n \in \mathcal{N}, k \in \mathcal{K}_n. \tag{3.7}$$

where $\mathbf{S} = \{S_1 \cdots S_N\}$ is the vector of the states of all the FIFO queues, and $P[\mathbf{S}, H_n = k]$ is the probability that the states of the queues are $\mathbf{S}$ and $H_n = k$, which is required as we can transmit a packet from the kth flow only when the HoL packet belongs to the kth flow.

For a given index set $\mathcal{I} \subset \{1, \ldots, N\}$, and a sequence of variables $x_1, \ldots, x_N$, we use the notation $x_{\mathcal{A}}$ to represent the $|\mathcal{A}|$-tuple of x_is whose indices satisfy $i \in \mathcal{A}$. Let $\mathcal{I}_n \triangleq \{1, \ldots, N\} - \{n\}$, and $\mathbf{K}_n \triangleq \mathcal{K}_1 \times \cdots \times \mathcal{K}_{n-1} \times \mathcal{K}_{n+1} \times \cdots \times \mathcal{K}_N$, where × denotes the Cartesian product. We can then calculate the probability $P[\mathbf{S}, H_n = k]$ in Equation 3.7 as

$$\begin{aligned}
& P[\mathbf{S}, H_n = k] \\
&= \sum_{l_{\mathcal{I}_n} \in \mathbf{K}_n} P[\mathbf{S}, H_n = k, H_{\mathcal{I}_n} = l_{\mathcal{I}_n}] \\
&= \sum_{l_{\mathcal{I}_n} \in \mathbf{K}_n} P[\mathbf{S} | H_n = k, H_{\mathcal{I}_n} = l_{\mathcal{I}_n}] P[H_n = k, H_{\mathcal{I}_n} = l_{\mathcal{I}_n}] \\
&= \sum_{l_{\mathcal{I}_n} \in \mathbf{K}_n} P[S_n | H_n = k] \prod_{i \in \mathcal{I}_n} P[S_i | H_i = l_i] \\
&\qquad\qquad\qquad\qquad P[H_n = k, H_{\mathcal{I}_n} = l_{\mathcal{I}_n}]. \qquad (3.8)
\end{aligned}$$

We now calculate $P[H_n = k, H_{\mathcal{I}_n} = l_{\mathcal{I}_n}]$. We claim that

$$P[H_n = k, H_{\mathcal{I}_n} = l_{\mathcal{I}_n}] = P[H_n = k] \prod_{i \in \mathcal{I}_n} P[H_i = l_i]. \qquad (3.9)$$

To prove this claim, we show that the following conditions hold.

$$\begin{aligned}
&\text{C1: } P[H_n = k | H_{\mathcal{I}_n} = l_{\mathcal{I}_n}] = P[H_n = k] \\
&\text{C2: } P[H_n = k | H_{\mathcal{I}_n - \{N\}} = l_{\mathcal{I}_n - \{N\}}] = P[H_n = k] \\
&\quad \vdots \\
&\text{CN: } P[H_n = k | H_1 = l_1] = P[H_n = k]
\end{aligned} \tag{3.10}$$

We can calculate the conditional probabilities in the left hand side of the conditions; C1, C2, . . ., CN in Equation 3.10 by using a Markov chain. For C1, we can write a state transition probability of going from state $H_n = k$ to $H_n = m$ as $P[H_n = k \to H_n = m | H_{\mathcal{I}_n} = l_{\mathcal{I}_n}]$, which is equal to $\bar{p}_{n,k}\pi_n\alpha_{n,m}$, where π_n is the probability of $\mathcal{Q}_n$ is selected to transmit packets. Similarly, if we write the state transition probabilities for the other conditions C2, . . ., CN, we have

$$\begin{aligned}
& P[H_n = k \to H_n = m | H_{\mathcal{I}_n} = l_{\mathcal{I}_n}] \\
= \; & P[H_n = k \to H_n = m | H_{\mathcal{I}_n} = l_{\mathcal{I}_n - \{N\}}] \\
\vdots \; & \\
= \; & P[H_n = k \to H_n = m | H_l = l_1] \\
= \; & P[H_n = k \to H_n = m] \\
= \; & \bar{p}_{n,k}\pi_n\alpha_{n,m}
\end{aligned} \tag{3.11}$$

Therefore, in all Markov chains we can create for C1, C2, …, CN, we have the same transition probabilities, so we have

$$\begin{aligned}
& P[H_n = k | H_{\mathcal{I}_n} = l_{\mathcal{I}_n}] \\
= \quad & P[H_n = k | H_{\mathcal{I}_n} = l_{\mathcal{I}_n - \{N\}}] \\
\vdots \quad & \\
= \quad & P[H_n = k | H_1 = l_1] \\
= \quad & P[H_n = k] \qquad (3.12)
\end{aligned}$$

This proves our claim in Equation 3.9.

Now that we have shown that Equation 3.9 holds, Equation 3.8 is expressed as

$$\begin{aligned}
& P[\mathbf{S}, H_n = k] \\
& = \sum_{l_{\mathcal{I}_n} \in \mathbf{K}_n} P[S_n | H_n = k] P[H_n = k] \\
& \prod_{i \in \mathcal{I}_n} P[S_i | H_i = l_i] P[H_i = l_i] \qquad (3.13)
\end{aligned}$$

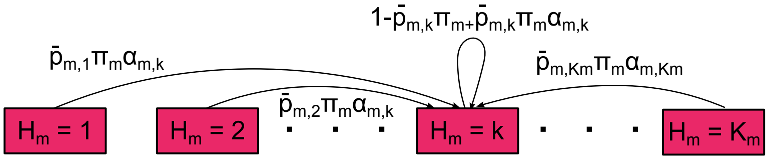


Figure 25. The state transition diagram for the states $H_m = k$, $\forall k \in \mathcal{K}_m$ and for the mth queue. Note that this state transition diagram only shows a subset of state transitions for clarity.

Let $\xi_{n,k}(S_n) \triangleq P[S_n|H_n = k]$, Equation 3.13 further yields

$$P[\mathbf{S}, H_n = k] = \xi_{n,k}(S_n)P[H_n = k] \prod_{m\in\mathcal{N}-\{n\}} \left(\sum_{k\in\mathcal{K}_m} \xi_{m,k}(S_m)P[H_m = k]\right). \tag{3.14}$$

Now, we should calculate $P[H_m = k]$ in Equation 3.14. The state transition diagram for the states $H_m = k$, $\forall k \in \mathcal{K}_m$ and for the mth queue is shown in Figure 25. We can write the global balance equations for the state $H_m = k$ as

$$P[H_m = 1]\bar{p}_{m,1}\pi_m\alpha_{m,k} + \ldots + P[H_m = k][1 - \bar{p}_{m,k}\pi_m + \bar{p}_{m,k}\pi_m\alpha_{m,k}] + \ldots + P[H_m = K_m]\bar{p}_{m,K_m}\pi_m\alpha_{m,k} = P[H_m = k], \tag{3.15}$$

which is expressed as

$$\alpha_{m,k} \sum_{i \in \mathcal{K}_m} \bar{p}_{m,i} P[H_m = i] = P[H_m = k] \bar{p}_{m,k}. \tag{3.16}$$

Similarly, the global balance equations for state $H_m = l$ leads to

$$\alpha_{m,l} \sum_{i \in \mathcal{K}_m} \bar{p}_{m,i} P[H_m = i] = P[H_m = l] \bar{p}_{m,l}. \tag{3.17}$$

From Equation 3.16 and Equation 3.17, we have

$$\frac{\alpha_{m,k}}{\alpha_{m,l}} = \frac{P[H_m = k]\bar{p}_{m,k}}{P[H_m = l]\bar{p}_{m,l}}. \tag{3.18}$$

Thus, we have

$$P[H_m = l] = \frac{\alpha_{m,l}}{\bar{p}_{m,l}} \frac{P[H_m = k]}{\alpha_{m,k}} \bar{p}_{m,k}. \tag{3.19}$$

By law of total probability, $\sum_{l \in \mathcal{K}_m} P[H_m = l] = 1$ should be satisfied, thus we have

$$P[H_m = k] = \frac{\lambda_{m,k}/\bar{p}_{m,k}}{\sum_{l \in \mathcal{K}_m} \lambda_{m,l}/\bar{p}_{m,l}}. \tag{3.20}$$

When Equation 3.20 is substituted in Equation 3.14, we have

$$P[\mathbf{S}, H_n = k] = \xi_{n,k}(S_n)\frac{\lambda_{n,k}/\bar{p}_{n,k}}{\sum_{l\in\mathcal{K}_n}\lambda_{n,l}/\bar{p}_{n,l}} \prod_{m\in\mathcal{N}-\{n\}} \frac{\sum_{k\in\mathcal{K}_m}\xi_{m,k}(S_m)\lambda_{m,k}/\bar{p}_{m,k}}{\sum_{k\in\mathcal{K}_m}\lambda_{m,k}/\bar{p}_{m,k}}. \tag{3.21}$$

Since $\rho_{m,k}(S_m) = \frac{\xi_{m,k}(S_m)}{\bar{p}_{m,k}}$, we have

$$P[\mathbf{S}, H_n = k] = \xi_{n,k}(S_n)\frac{\lambda_{n,k}/\bar{p}_{n,k}}{\sum_{l\in\mathcal{K}_m}\lambda_{n,l}/\bar{p}_{n,l}} \prod_{m\in\mathcal{N}-\{n\}} \frac{\sum_{k\in\mathcal{K}_m}\rho_{m,k}(S_m)\lambda_{m,k}}{\sum_{k\in\mathcal{K}_m}\lambda_{m,k}/\bar{p}_{m,k}}. \tag{3.22}$$

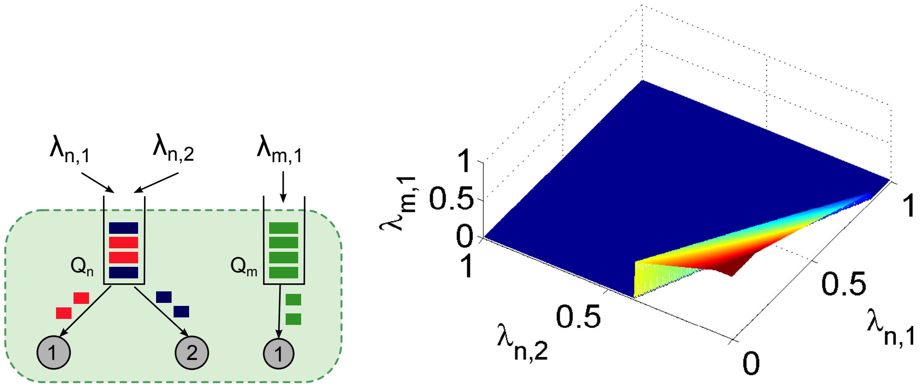
When we substitute Equation 3.22 into Equation 3.7, we have Equation 3.6. This concludes the proof.

■

The support region of a FIFO queuing system with N FIFO queues served by a wireless medium is characterized by $\Lambda = \{\{\lambda_{n,k}\}_{\forall n\in\mathcal{N},k\in\mathcal{K}_n}|$ Equation 3.6, Equation 3.5, $\lambda_{n,k} \geq 0$, $n \in \mathcal{N}$, $k \in \mathcal{K}_n$, $\tau_n(\mathbf{S}) \geq 0, \forall\, n \in \mathcal{N}, \mathbf{S} \in \mathcal{S}\}$.

Example 6. *Now let us consider two FIFO queues $\mathcal{Q}_n$ and $\mathcal{Q}_m$ which are shared by three flows with rates; $\lambda_{n,1}$, $\lambda_{n,2}$, and $\lambda_{m,1}$ (Figure 26(a)). According to Theorem 4, the support region Λ should include arrival rates satisfying inequalities in (Equation 3.6 and Equation 3.5. In this example, with two queues and three flows, these inequalities are equivalent to*

$$\lambda_{n,1} + \lambda_{n,2} + \lambda_{m,1} \leq \left(\frac{p_{m,1}(\lambda_{n,1} + \lambda_{n,2})}{\lambda_{n,1}/\bar{p}_{n,1} + \lambda_{n,2}/\bar{p}_{n,2}}\right) + \bar{p}_{m,1} \tag{3.23}$$



(a) Two FIFO Queues (b) Support Region

Figure 26. (a) Two FIFO queues; $\mathcal{Q}_n$ and $\mathcal{Q}_m$ are shared by two and one flows, respectively. (b) Three dimensional support region with $\lambda_{n,1}$, $\lambda_{n,2}$ and $\lambda_{m,1}$ for the two-FIFO queues scenario shown in (a).

with $\lambda_{n,1}/\bar{p}_{n,1} + \lambda_{n,2}/\bar{p}_{n,2} \leq 1$*, and* $\lambda_{m,1}/\bar{p}_{m,1} \leq 1$*. The support region corresponding to these inequalities is the region below the surface in Figure 26(b).* [1] $\square$

In general, we wish to find the optimal operating points on the boundary of the support region Λ. However, the support region may not be convex for arbitrary number of queues and flows. Developing a convex inner bound on the support region is crucial for developing efficient resource allocation algorithms for wireless networks with FIFO queues. We thus next propose a convex inner bound on the support region.

---

[1] *Note that the time sharing argument to convexify the support region does not apply to this scenario, because the non-convexity comes from the relationship among the arrival rates instead of the service rates from the FIFO queues. Thus, the centralized time-sharing for the arrival rates is not practical.*

3.4.3 A Convex Inner Bound on the Support Region:

Let us consider a flow with arrival rate $\lambda_{n,k}$ to the FIFO queue $\mathcal{Q}_n$. If there are no other flows and queues in the network, then the arrival rate should satisfy $\lambda_{n,k}/\bar{p}_{n,k} \leq 1$ according to Theorem 4. In this formulation, $\lambda_{n,k}/\bar{p}_{n,k}$ is the total amount of wireless resources that should be allocated to transmit the flow with rate $\lambda_{n,k}$. For multiple-flow, single-FIFO case, the support region is $\sum_{k\in\mathcal{K}_n} \lambda_{n,k}/\bar{p}_{n,k} \leq 1$. Similar to the single-flow case, $\lambda_{n,k}/\bar{p}_{n,k}$ term is the amount of wireless resources that should be allocated to the kth flow. Finally, for the general support region for arbitrary number of queues and flows, let us consider Equation 3.6 again. Assuming $\psi_m(S_m) = \frac{\sum_{k\in\mathcal{K}_m} \lambda_{m,k}\rho_{m,k}(S_m)}{\sum_{k\in\mathcal{K}_m} \lambda_{m,k}/\bar{p}_{m,k}}$, we can write $\sum_{k\in\mathcal{K}_n} \lambda_{n,k}$ from Equation 3.6 as;

$$\sum_{k\in\mathcal{K}_n} \lambda_{n,k} \leq \sum_{\mathbf{S}\in\mathcal{S}} \Bigg\{ \frac{\sum_{k\in\mathcal{K}_n} \lambda_{n,k} 1_{[S_n]}}{\sum_{k\in\mathcal{K}_n} \lambda_{n,k}/\bar{p}_{n,k}} \\ \prod_{m\in\mathcal{N}-\{n\}} \psi_m(S_m)\tau_n(\mathbf{S}) \Bigg\}, \forall n \in \mathcal{N}, k \in \mathcal{K}_n \tag{3.24}$$

which, assuming that $\sum_{k\in\mathcal{K}_n} \lambda_{n,k} > 0$, is equivalent to

$$\sum_{k\in\mathcal{K}_n} \lambda_{n,k}/\bar{p}_{n,k} \leq \sum_{\mathbf{S}\in\mathcal{S}} 1_{[S_n]} \prod_{m\in\mathcal{N}-\{n\}} \psi_m(S_m) \\ \tau_n(\mathbf{S}), \forall n \in \mathcal{N}, k \in \mathcal{K}_n \tag{3.25}$$

Intuitively speaking, the right hand side of Equation 3.25 corresponds to the amount of wireless resources that are allocated to the nth queue $\mathcal{Q}_n$. Thus, similar to the single-FIFO queue, we can consider that $\lambda_{n,k}/\bar{p}_{n,k}$ term corresponds to the amount of wireless resources that should be allocated to the kth flow.

Our key point while developing an inner bound on the support region is to provide rate fairness across competing flows in each FIFO queue. Since each flow requires $\lambda_{n,k}/\bar{p}_{n,k}$ amount of wireless resources; it is intuitive to have the following equality $\lambda_{n,k}/\bar{p}_{n,k} = \lambda_{n,l}/\bar{p}_{n,l}$, $k \neq l$ to fairly allocate wireless resources across flows. More generally, we define a function $a_n = \lambda_{n,k}/(\bar{p}_{n,k})^{\beta}$, $\forall k \in \mathcal{K}_n$ where $\beta \geq 1$, and we develop a support region for a_n instead of $\lambda_{n,k}$. The role of the exponent β is to provide flexibility to the targeted fairness. For example, if we want to allocate more resources to flows with better channels, then β should be larger.

Now, by the definition of a_n, we have the equivalent form

$$a_n \leq \sum_{\mathbf{S} \in \mathcal{S}} \frac{1_{[S_n]}}{\sum_{k \in \mathcal{K}_n} (\bar{p}_{n,k})^{\beta-1}} \prod_{m \in \mathcal{N} - \{n\}} \omega_m(S_m)$$

$$\tau_n(\mathbf{S}), \forall n \in \mathcal{N} \tag{3.26}$$

of Equation 3.6, where $\omega_m(S_m) = \frac{\sum_{k \in \mathcal{K}_m} (\bar{p}_{m,k})^{\beta} \rho_{m,k}(S_m)}{\sum_{k \in \mathcal{K}_m} (\bar{p}_{m,k})^{\beta-1}}$. As seen, Equation 3.26 is a convex function of a_n. Thus, we can define the region $\tilde{\Lambda} = \{\{a_n\}_{n \in \mathcal{N}} |$ Equation 3.26, Equation 3.5, $a_n \geq 0, \tau_n(\mathbf{S}) \geq 0, \forall n \in \mathcal{N}, \mathbf{S} \in \mathcal{S}\}$, which is clearly an inner bound on the actual support region Λ. Despite the fact that $\tilde{\Lambda}$ is only an inner bound on Λ, for some operating points, *i.e.,* at the intersection of $\lambda_{n,k}/\bar{p}_{n,k} = \lambda_{n,l}/\bar{p}_{n,l}$, $k \neq l$ lines, the two support regions ($\tilde{\Lambda}$ and Λ) coincide. Thus, for some utility functions, optimal operating points in both $\tilde{\Lambda}$ and Λ coincide. In the next section, we develop resource allocation schemes; dFC and qFC that achieve utility optimal operating points in $\tilde{\Lambda}$.

3.5 Flow Control and Scheduling

In this section, we develop resource allocation schemes; *deterministic FIFO-Control* (dFC), and a *queue-based FIFO control* (qFC).

In general, our goal is to solve the optimization problem

$$\begin{aligned} &\max_{\boldsymbol{\lambda}} \sum_{n\in\mathcal{N}} \sum_{k\in\mathcal{K}_n} U_{n,k}(\lambda_{n,k}) \\ &\text{s.t. } \lambda_{n,k} \in \Lambda, n \in \mathcal{N}, k \in \mathcal{K}_n \end{aligned} \tag{3.27}$$

and to find the corresponding optimal rates, where $U_{n,k}$ is a concave utility function assigned to flow with rate $\lambda_{n,k}$. Although the objective function $\sum_{n\in\mathcal{N}} \sum_{k\in\mathcal{K}_n} U_{n,k}(\lambda_{n,k})$ in Equation 3.27 is concave, the optimization domain Λ (*i.e.,* the support region) may not be convex. Thus, we convert this problem to a convex optimization problem based on the structure of the inner bound we have developed in Section 3.4.3. In particular, setting $a_n = \lambda_{n,k}/(\bar{p}_{n,k})^{\beta}$, the problem in Equation 3.27 reduces to $\max_{\boldsymbol{a}}$ $\sum_{n\in\mathcal{N}} \sum_{k\in\mathcal{K}_n} U_n(a_n(\bar{p}_{n,k})^{\beta})$, $a_n \in \tilde{\Lambda}, n \in \mathcal{N}$. This is our deterministic FIFO-control scheme; dFC and expressed explicitly as;

Deterministic FIFO-Control (dFC):

$$
\begin{aligned}
&\max_{\boldsymbol{a},\tau} \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}_n} U_{n,k}(a_n(\bar{p}_{n,k})^{\beta}) \\
&\text{s.t. } a_n \leq \sum_{\mathbf{S}\in\mathcal{S}} \frac{1_{[S_n]}}{\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^{\beta-1}} \\
&\prod_{m\in\mathcal{N}-\{n\}} \omega_m(S_m)\tau_n(\mathbf{S}), \forall n\in\mathcal{N} \\
&\sum_{n\in\mathcal{N}} \tau_n(\mathbf{S}) \leq 1, \forall \mathbf{S}\in\mathcal{S} \\
&a_n \geq 0, \forall n\in\mathcal{N}, \mathbf{S}\in\mathcal{S} \\
&\tau_n(\mathbf{S}) \geq 0, \forall n\in\mathcal{N}, \mathbf{S}\in\mathcal{S} \qquad (3.28)
\end{aligned}
$$

Note that dFC optimizes a_n and $\tau_n(\mathbf{S})$. After the optimal values are determined, packets are inserted into the FIFO queue $\mathcal{Q}_n$ depending on $\lambda_{n,k} = a_n(\bar{p}_{n,k})^{\beta}$ and served from the FIFO queue $\mathcal{Q}_n$ depending on $\tau_n(\mathbf{S})$.

Although dFC gives us optimal operating points in the support region; $\tilde{\Lambda}$, it is a centralized solution, and its adaptation to varying wireless channel conditions is limited. Thus, we also develop a more practical and queue-based FIFO-control scheme qFC, next.

Queue-Based FIFO-Control (qFC):

- *Flow Control:* At every slot t, the flow controller attached to the FIFO queue $\mathcal{Q}_n$ determines $a_n(t)$ according to;

$$\begin{aligned} &\max_{\boldsymbol{a}} \ M\big[\sum_{k\in\mathcal{K}_n} U_{n,k}(a_n(t)(\bar{p}_{n,k})^{\beta})\big] - Q_n(t)a_n(t) \\ &\text{s.t. } a_n(t) \leq R_n^{max}, a_n(t) \geq 0 \end{aligned} \tag{3.29}$$

where M is a large positive number, $Q_n(t)$ is the queue size of $\mathcal{Q}_n$ at time slot t and R_n^{max} is a positive value larger than the maximum outgoing rate from FIFO queue $\mathcal{Q}_n$ (which is $R_n^{max} > 1$ as we assume that the maximum outgoing rate from a queue is 1 packet per slot). After $a_n(t)$ is determined according to Equation 3.29, $\lambda_{n,k}(t)$ is set as $\lambda_{n,k}(t) = a_n(t)\ (\bar{p}_{n,k})^{\beta}$. Then, $\lambda_{n,k}(t)$ packets from the kth flow are inserted in $\mathcal{Q}_n$.

- *Scheduling:* At slot t, the scheduling algorithm determines the FIFO queue from which a packet is transmitted according to;

$$\begin{aligned} &\max_{\boldsymbol{\tau}} \ \sum_{n\in\mathcal{N}} Q_n(t)\frac{1_{[S_n(t)]}}{\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^{\beta}}\tau_n(\mathbf{S}(t)) \\ &\text{s.t. } \sum_{n\in\mathcal{N}} \tau_n(\mathbf{S}(t)) \leq 1, \\ &\qquad \tau_n(\mathbf{S}(t)) \geq 0 \end{aligned} \tag{3.30}$$

After $\tau_n(\mathbf{S}(t))$ is determined, the outgoing traffic rate from queue $\mathcal{Q}_n$ is set to $g_n(t) = \tau_n(\mathbf{S}(t))1_{[S_n(t)]}$, and $g_n(t)$ packets (which is 1 or 0 in our case) are transmitted from $\mathcal{Q}_n$.

Thus, the queue dynamics change according to Equation 3.1 and based on Equation 3.29 and Equation 3.30. Such queue dynamics lead to the following result.

Theorem 5. *If the channel states are i.i.d. over time slots, the traffic arrival rates are controlled by the rate control algorithm in Equation 3.29, and the FIFO queues are served by the scheduling algorithm in Equation 3.30, then the admitted flow rates converge to the utility optimal operating point in the support region* $\tilde{\Lambda}$ *with increasing* M.

Proof: Let us define a Lyapunov function as; $L(\boldsymbol{Q}(t)) = \sum_{n\in\mathcal{N}} Q_n(t)^2$, and the Lyapunov drift as; $\Delta(\boldsymbol{Q}(t)) = E[L(\boldsymbol{Q}(t+1)) - L(\boldsymbol{Q}(t))|\boldsymbol{Q}(t)]$, where $\boldsymbol{Q}(t) = \{Q_1(t), \ldots Q_N(t)\}$. Then, the Lyapunov drift is expressed as;

$$\Delta(\boldsymbol{Q}(t)) = E[\sum_{n\in\mathcal{N}} Q_n(t+1)^2 - \sum_{n\in\mathcal{N}} Q_n(t)^2|\boldsymbol{Q}(t)] \tag{3.31}$$

Note that we have, from Equation 3.1 and the assumption $a_n(t) = \lambda_{n,k}(t)/(\bar{p}_{n,k})^\beta$ that,

$$Q_n(t+1) \leq \max[Q_n(t) - g_n(t), 0] + a_n(t) \sum_{k\in\mathcal{K}_n} (\bar{p}_{n,k})^\beta \tag{3.32}$$

Using Equation 3.32 in Equation 3.31, and using the fact that $(\max(Q-b,0)+A)^2 \leq Q^2 + A^2 + b^2 + 2Q(A-b)$, we have

$$\begin{aligned}\Delta(\boldsymbol{Q}(t)) \leq E\big[\sum_{n\in\mathcal{N}} \{Q_n(t)^2 + (a_n(t) \sum_{k\in\mathcal{K}_n} (\bar{p}_{n,k})^\beta)^2 + (g_n(t))^2 \\ + 2Q_n(t)(a_n(t) \sum_{k\in\mathcal{K}_n} (\bar{p}_{n,k})^\beta - g_n(t))\} - \sum_{n\in\mathcal{N}} Q_n(t)^2|\boldsymbol{Q}(t)\big]\end{aligned} \tag{3.33}$$

which is expressed as

$$\frac{\Delta(\boldsymbol{Q}(t))}{2\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^\beta} \leq E\big[\sum_{n\in\mathcal{N}}\big\{\frac{(a_n(t))^2}{2}\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^\beta +$$

$$\frac{(g_n(t))^2}{2\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^\beta} + Q_n(t)a_n(t) - \frac{Q_n(t)g_n(t)}{\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^\beta}\big\}|\boldsymbol{Q}(t)\big] \tag{3.34}$$

There always exist a finite and positive B satisfying; $B \geq E\big[\sum_{n\in\mathcal{N}}\big\{\frac{(a_n(t))^2}{2}\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^\beta + \frac{(g_n(t))^2}{2\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^\beta}\big\}\big]$. Thus, Equation 3.34 is expressed as;

$$\frac{\Delta(\boldsymbol{Q}(t))}{2\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^\beta} \leq B + E\big[\sum_{n\in\mathcal{N}} Q_n(t)\big(a_n(t) -$$

$$\frac{g_n(t)}{\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^\beta}\big)|\boldsymbol{Q}(t)\big] \tag{3.35}$$

Note that if the flow arrival rates $\lambda_{n,k}(t) = a_n(t)(\bar{p}_{n,k})^\beta$ are inside the support region $\tilde{\Lambda}$, then the minimizing the right hand side of the drift inequality in Equation 3.35 corresponds to the scheduling part of qFC in Equation 3.30.

Now, let us consider again the support region constraint in Equation 3.7, which is $\lambda_{n,k} \leq \sum_{\mathbf{S}\in\mathcal{S}} P[\mathbf{S}, H_n = k]1_{[S_n]}\tau_n(\mathbf{S}), \forall n \in \mathcal{N}, k \in \mathcal{K}_n$, and expressed as;

$$\sum_{k\in\mathcal{K}_n} \lambda_{n,k} \leq \sum_{\mathbf{S}\in\mathcal{S}}\big(\sum_{k\in\mathcal{K}_n} P[\mathbf{S}, H_n = k]\big)1_{[S_n]}\tau_n(\mathbf{S}) \tag{3.36}$$

which is equal to

$$\sum_{k \in \mathcal{K}_n} \lambda_{n,k} \leq \sum_{\mathbf{S} \in \mathcal{S}} P[\mathbf{S}] 1_{[S_n]} \tau_n(\mathbf{S}) \tag{3.37}$$

Since $\lambda_{n,k} = a_n (\bar{p}_{n,k})^{\beta}$, we have

$$\sum_{k \in \mathcal{K}_n} a_n (\bar{p}_{n,k})^{\beta} \leq \sum_{(S_1 \ldots S_N) \in \mathcal{S}} P[\mathbf{S}] 1_{[S_n]} \tau_n(\mathbf{S}) \tag{3.38}$$

$$a_n \sum_{k \in \mathcal{K}_n} (\bar{p}_{n,k})^{\beta} \leq \sum_{(S_1 \ldots S_N) \in \mathcal{S}} P[\mathbf{S}] 1_{[S_n]} \tau_n(\mathbf{S}) \tag{3.39}$$

$$a_n \leq \sum_{(S_1 \ldots S_N) \in \mathcal{S}} P[\mathbf{S}] \frac{1_{[S_n]} \tau_n(\mathbf{S})}{\sum_{k \in \mathcal{K}_n} (\bar{p}_{n,k})^{\beta}} \tag{3.40}$$

Let $g_n = 1_{[S_n]} \tau_n(\mathbf{S})$. Then, Equation 3.40 is expressed as;

$$a_n \leq \sum_{\mathbf{S} \in \mathcal{S}} P[\mathbf{S}] \frac{g_n}{\sum_{k \in \mathcal{K}_n} (\bar{p}_{n,k})^{\beta}} \tag{3.41}$$

There exists a small positive value ϵ satisfying

$$a_n + \epsilon \leq \sum_{\mathbf{S} \in \mathcal{S}} P[\mathbf{S}] \frac{g_n}{\sum_{k \in \mathcal{K}_n} (\bar{p}_{n,k})^{\beta}} \tag{3.42}$$

Thus, we can find a randomized policy satisfying

$$E\big[\overset{*}{a}_n(t) - \frac{\overset{*}{g}_n(t)}{\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^{\beta}}\big] \leq -\epsilon \tag{3.43}$$

Now, let us consider Equation 3.35 again, which is expressed as;

$$\frac{\Delta(\boldsymbol{Q}(t))}{2\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^{\beta}} \leq B + \sum_{n\in\mathcal{N}} Q_n(t)E\big[a_n(t)- \\ \frac{g_n(t)}{\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^{\beta}}|\boldsymbol{Q}(t)\big] \tag{3.44}$$

We minimize the right hand side of Equation 3.35, so the following inequality satisfies;

$$E\big[a_n(t) - \frac{g_n(t)}{\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^{\beta}}|\boldsymbol{Q}(t)\big] \leq E\big[\overset{*}{a}_n(t) - \frac{\overset{*}{g}_n(t)}{\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^{\beta}} \\ |\boldsymbol{Q}(t)\big] \tag{3.45}$$

where $\overset{*}{a}_n(t)$ and $\overset{*}{g}_n(t)$ are the solutions of a randomized policy. Incorporating Equation 3.43 and Equation 3.45 into Equation 3.44, we have

$$\frac{\Delta(\boldsymbol{Q}(t))}{2\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^{\beta}} \leq B - \epsilon\sum_{n\in\mathcal{N}} Q_n(t) \tag{3.46}$$

The time average of Equation 3.46 leads to

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \frac{\Delta(\boldsymbol{Q}(\tau))}{2\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^{\beta}} \leq \limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1}\big[B- \\ \epsilon\sum_{n\in\mathcal{N}} Q_n(\tau)\big] \tag{3.47}$$

By law of telescoping sum and iterated expectations, we can obtain

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1}\big(\sum_{n\in\mathcal{N}} E[Q_n(\tau)]\big) \leq \frac{B}{\epsilon} \tag{3.48}$$

This concludes that the time average of the queues are bounded if the arrival rates are inside the capacity region $\tilde{\Lambda}$.

Now, let us focus on the original claim of Theorem 5. Let us consider a drift+penalty function as;

$$\frac{\Delta(\boldsymbol{Q}(t))}{2\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^{\beta}} - \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}_n} ME[U_{n,k}(\lambda_{n,k}(t))|\boldsymbol{Q}(t)] \leq \\ B + E\big[\sum_{n\in\mathcal{N}} Q_n(t)\big(a_n(t) - \frac{g_n(t)}{\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^{\beta}}\big)|\boldsymbol{Q}(t)\big]- \\ \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}_n} ME[U_{n,k}(\lambda_{n,k}(t))|\boldsymbol{Q}(t)] \tag{3.49}$$

Since we set $\lambda_{n,k}(t) = a_n(t)(\bar{p}_{n,k})^\beta$, we have

$$\begin{aligned}
&\frac{\Delta(\boldsymbol{Q}(t))}{2\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^\beta} - \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}_n} ME[U_{n,k}(a_n(t)(\bar{p}_{n,k})^\beta)|\boldsymbol{Q}(t)] \\
&\leq B + \sum_{n\in\mathcal{N}} E\big[Q_n(t)\big(a_n(t) - \frac{g_n(t)}{\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^\beta}\big)|\boldsymbol{Q}(t)\big] - \\
&\sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}_n} ME[U_{n,k}(a_n(t)(\bar{p}_{n,k})^\beta)|\boldsymbol{Q}(t)] \qquad (3.50)
\end{aligned}$$

Note that minimizing the right hand side of Equation 3.50 corresponds to the flow control and scheduling algorithms of qFC in Equation 3.29 and Equation 3.30, respectively. Since there exists a randomized policy satisfying Equation 3.43, Equation 3.50 is expressed as

$$\begin{aligned}
&\frac{\Delta(\boldsymbol{Q}(t))}{2\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^\beta} - \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}_n} ME[U_{n,k}(a_n(t)(\bar{p}_{n,k})^\beta)|\boldsymbol{Q}(t)] \\
&\leq B - \epsilon\sum_{n\in\mathcal{N}} Q_n(t) - \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}_n} MU_{n,k}(A_n(\bar{p}_{n,k})^\beta + \delta) \qquad (3.51)
\end{aligned}$$

where $\sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}_n} U_{n,k}(A_n(\bar{p}_{n,k})^\beta+\delta)$ is the maximum time average of the sum utility function that can be achieved by any control policy that stabilizes the system. Then, the time average of Equation 3.51 becomes

$$\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\Bigg\{\frac{\Delta(Q(\tau))}{2\sum_{k\in\mathcal{K}_n}(\bar{p}_{n,k})^\beta}-$$
$$\sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}_n} ME[U_{n,k}(a_n(\tau)(\bar{p}_{n,k})^\beta)|\boldsymbol{Q}(t)]\Bigg\}\leq$$
$$\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\Bigg\{B-\epsilon\sum_{n\in\mathcal{N}}Q_n(\tau)-$$
$$\sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}_n} MU_{n,k}(A_n(\bar{p}_{n,k})^\beta+\delta)\Bigg\} \tag{3.52}$$

Now, let us first consider the stability of the queues. If both sides of Equation 3.52 are divided by ϵ and the terms are rearranged, we have

$$\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\{\sum_{n\in\mathcal{N}}Q_n(\tau)\}\leq\frac{B}{\epsilon}+$$
$$\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\{\sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}_n}\frac{M}{\epsilon}E[U_{n,k}(a_n(\tau)(\bar{p}_{n,k})^\beta)]\}-$$
$$\sum_{k\in\mathcal{N}}\sum_{k\in\mathcal{K}_n}\frac{M}{\epsilon}U_{n,k}(A_n(\bar{p}_{n,k})^\beta+\delta) \tag{3.53}$$

Since the right hand side is a positive finite value, this concludes that the time averages of the total queue sizes are bounded.

Now, let us consider the optimality. If both sides of Equation 3.52 are divided by M, we have

$$\begin{aligned}
& -\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}_n} E[U_{n,k}(a_n(\tau)(\bar{p}_{n,k})^{\beta})] \leq \\
& \limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\{\frac{B}{M}-\frac{\epsilon}{M}\sum_{n\in\mathcal{N}}Q_n(\tau)- \\
& \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}_n}U_{n,k}(A_n(\bar{p}_{n,k})^{\beta}+\delta)\} \qquad (3.54)
\end{aligned}$$

By arranging the terms, we have

$$\begin{aligned}
& \limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}_n} E[U_{n,k}(a_n(\tau)(\bar{p}_{n,k})^{\beta})] \geq \\
& \limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\{\sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}_n}U_{n,k}(A_n(\bar{p}_{n,k})^{\beta}+\delta)-\frac{B}{M} \\
& +\frac{\epsilon}{M}\sum_{n\in\mathcal{N}}Q_n(\tau)\} \qquad (3.55)
\end{aligned}$$

Since $\frac{\epsilon}{M}\sum_{n\in\mathcal{N}}Q_n(\tau)$ is positive for any τ, we have

$$\begin{aligned}
& \limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}_n} E[U_{n,k}(a_n(\tau)(\bar{p}_{n,k})^{\beta})] \geq \\
& \limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\{\sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}_n}U_{n,k}(A_n(\bar{p}_{n,k})^{\beta}+\delta)-\frac{B}{M}\} \qquad (3.56)
\end{aligned}$$

which leads to

$$\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{n\in\mathcal{N}} \sum_{k\in\mathcal{K}_n} E[U_{n,k}(a_n(\tau)(\bar{p}_{n,k})^{\beta})] \geq$$

$$\sum_{n\in\mathcal{N}} \sum_{k\in\mathcal{K}_n} U_{n,k}(A_n(\bar{p}_{n,k})^{\beta} + \delta) - \frac{B}{M} \tag{3.57}$$

This proves that the admitted flow rates converge to the utility optimal operating point with increasing M. This concludes the proof. □

3.6 Performance Evaluation

In this section, we evaluate our dFC and qFC algorithms as compared to the baselines; (i) *optimal* solution, and (ii) *max-weight* algorithm for different number of FIFO queues and flows. Next, we briefly explain our baselines.

3.6.1 Baselines

The *optimal* solution is a solution to Equation 3.27, and we compared dFC and qFC with the *optimal* solution for some scenarios where the support region Λ is convex. On the other hand, *max-weight* algorithm is a queue-based flow control and max-weight scheduling scheme. Our baseline *max-weight* algorithm mimics the structure of the solution provided in [15], and it is summarized briefly in the following.

Max-weight for FIFO:

- *Flow Control:* At every time slot t, the flow controller attached to the FIFO queue $\mathcal{Q}_n$ determines $\lambda_{n,k}(t)$ according to;

  $$\begin{aligned} \max_{\boldsymbol{\lambda}} \; & M\big[\sum_{k \in \mathcal{K}_n} U_{n,k}(\lambda_{n,k}(t))\big] - Q_{n,k}(t)\lambda_{n,k}(t) \\ \text{s.t. } & \lambda_{n,k}(t) \leq R_{n,k}^{max}, \forall k \in \mathcal{K}_n \end{aligned} \tag{3.58}$$

  where M and $R_{n,k}^{max}$ are positive large constants similar to Equation 3.29, and $Q_{n,k}(t)$ is the number of packets that belong to the kth flow in queue $\mathcal{Q}_n$.

- *Scheduling:* At slot t, the scheduling algorithm determines the FIFO queue from which a packet is transmitted according to;

  $$\begin{aligned} \max_{\boldsymbol{\tau}} \; & \sum_{n \in \mathcal{N}} Q_n(t) 1_{[S_n(t)]} \tau_n(\mathbf{S}(t)) \\ \text{s.t. } & \sum_{n \in \mathcal{N}} \tau_n(\mathbf{S}(t)) \leq 1 \\ & \tau_n(\mathbf{S}(t)) \geq 0 \end{aligned} \tag{3.59}$$

  After τ_n $(\mathbf{S}(t))$ is determined, a packet from the queue $\mathcal{Q}_n$ is transmitted if τ_n $(\mathbf{S}(t)) = 1$; no packet is transmitted, otherwise.

Next, we present our simulation results for single and multiple FIFO queues.

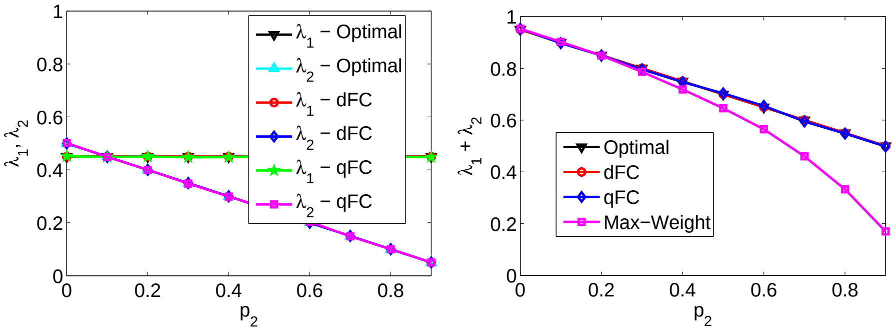
3.6.2 Single-FIFO Queue

In this section, we consider a single FIFO queue $\mathcal{Q}_1$. Similar to Section 3.4.1, we drop the queue index $n = 1$ from the notation for brevity. In other words, we write λ_k instead of $\lambda_{1,k}$, p_k instead of $p_{1,k}$, and so on.

Figure 27 presents simulation results for a single queue and two flows for $p_1 = 0.1$, $\beta = 1$, and $U_k(\lambda_k) = \log(\lambda_k)$. In this setup, the channel associated with the first flow will be OFF with probability $p_1 = 0.1$ and the utility function is a logarithm function. Figure 27(a) shows per-flow rates; λ_1 and λ_2 when the channel OFF probability for the second user p_2 is increasing. As seen, λ_1 is the same for all algorithms; optimal, dFC, and qFC. This also holds for λ_2. These results show that our algorithms dFC and qFC are as good as the optimal solution, and achieve the optimal operating points in Λ in this scenario. The simulations results also show that our algorithms reduce the second flow rate λ_2 when p_2 increases while λ_1 and p_1 do not change. This means that our algorithms do not penalize a flow (flow 1) when the channel of another competing flow (flow 2) deteriorates, which shows the effectiveness of our algorithms to provide fairness.

Figure 27(b) shows the total rate $\lambda_1 + \lambda_2$ versus p_2 for the same setup. As seen, our algorithms improves throughput over max-weight significantly. This is expected as our algorithms are designed to reduce the HoL blocking and to allocate wireless resources fairly among multiple flows.

Figure 28 shows simulation results for a single queue shared by multiple flows. In this setup, each channel's OFF probability p_k is selected randomly between $[0, 1]$, $\beta = 1$, $U_k(\lambda_k) = \log(\lambda_k)$. The simulations are repeated for 1000 different seeds, and the average values are reported. Figure 28(a) shows average flow rate versus number of flows for our algorithms as well as max-weight. As seen,



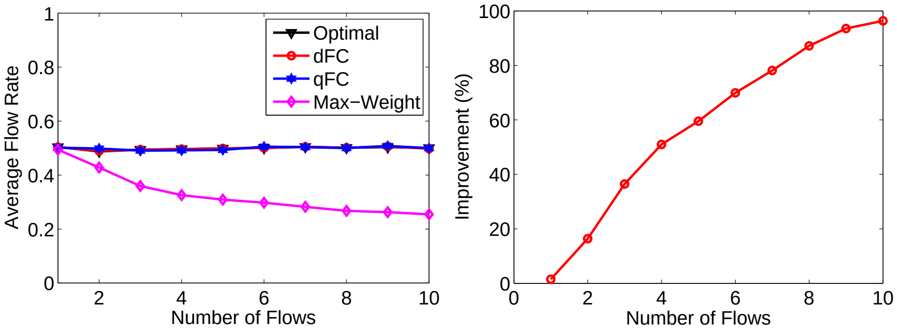
(a) Per-flow rates (b) Total rate

Figure 27. Single-FIFO queue shared by two flows when $p_1 = 0.1$, $\beta = 1$, and $U_k(\lambda_k) = \log(\lambda_k)$. (a) Per-flow rates vs. p_2. (b) Total flow rate vs. p_2.

dFC and qFC are as good as the optimal solution, and they improve over max-weight significantly. Figure 28(b) shows the same simulation results, but reports the improvement of qFC over max-weight. This figure shows that the improvement of our algorithms increases with increasing number of flows. Indeed, the improvement is up to 100% when $K = 10$, which is significant. The improvement is higher for large number of flows, because our algorithm allocates resources to the flows based on the quality of their channels and reduces the flow rate for the flows with bad channel conditions. However, max-weight does not have such a mechanism, and when there are more flows in the system, the probability of having a flow with bad channel condition increases, which reduces the overall throughput.

3.6.3 Two-FIFO Queues

In this section, we consider two FIFO queues $\mathcal{Q}_m$ and $\mathcal{Q}_n$. There are four flows in the system and each queue carries two flows, *i.e.,* $\mathcal{Q}_n$ carries flows with rates $\lambda_{n,1}$, $\lambda_{n,2}$ and $\mathcal{Q}_m$ carries flows with rates $\lambda_{m,1}$, $\lambda_{m,2}$.



(a) Flow rates (b) Throughput improvement

Figure 28. Single-FIFO queue shared by multiple flows. p_k is selected randomly between $[0, 1]$, $\beta = 1$, and $U_k(\lambda_k) = \log(\lambda_k)$. (a) Average flow rate versus number of flows. (b) Percentage of throughput improvement of qFC over max-weight.

Figure 29(a) shows the total flow rate versus β for the scenario of two-FIFO queues with four flows when channel OFF probabilities are $p_{n,1} = 0.1$, $p_{n,2} = 0.5$, $p_{m,1} = 0.1$, $p_{m,2} = 0.5$, and $\log$ utility is employed, *i.e.*, $U_{n,k}(\lambda_{n,k}) = \log(\lambda_{n,k})$. (We do not present the results of the optimal solution as the support region Λ is not convex in this scenario.) As seen, dFC and qFC have the same performance and improve over max-weight. The improvement increases with increasing β as dFC and qFC penalize flows with bad channel conditions more when β increases, which increases the total throughput.

Figure 29(b) shows the total rate versus $p_{n,2} = p_{m,2}$ for two-FIFO queues with four flows when $p_{n,1} = p_{m,1} = 0.1$ and $\beta = 2$. As seen, dFC and qFC improve significantly over max-weight. Furthermore, they achieve almost maximum achievable rate 1 all the time. The reason is that dFC and qFC penalizes the queues with bad channels. For example, when $p_{n,2} = p_{m,2} = 1$, the total rate is 1, because they allocate all the resources to $\lambda_{n,1}$ and $\lambda_{m,1}$ as there is no point to allocate those resources to $\lambda_{n,2}$ and $\lambda_{m,2}$ since their channels are always OFF. On the other hand, max-weight does not arrange

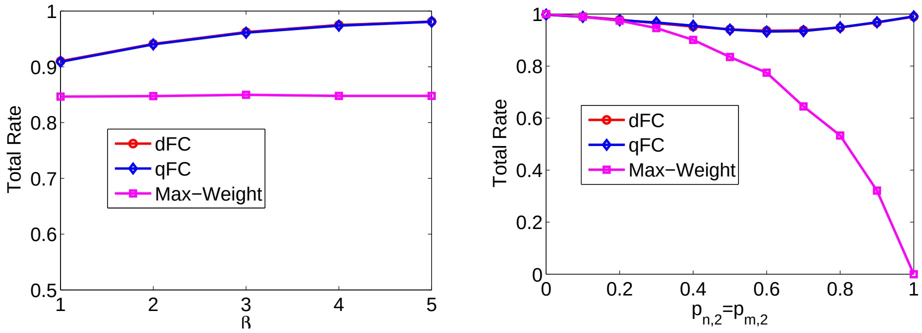
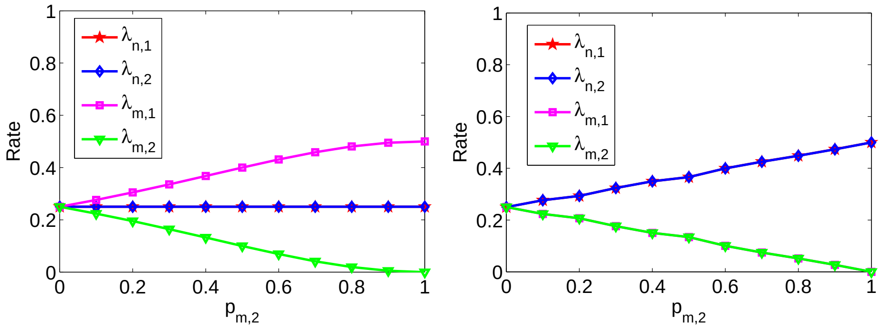


Figure 29. Two FIFO queues with four flows. (a) Total flow rate versus β when $p_{n,1} = 0.1$, $p_{n,1} = 0.5$, $p_{m,1} = 0.1$, $p_{m,2} = 0.5$, and log utility is employed, *i.e.,* $U_{n,k}(\lambda_{n,k}) = \log(\lambda_{n,k})$. (b) Total rate versus $p_{n,2} = p_{m,2}$ when $p_{n,1} = p_{m,1} = 0.1$ and $\beta = 2$.

the flow and queue service rates based on the channel conditions, so the total rate reduces to 0 when $p_{n,2} = p_{m,2} = 1$, *i.e.,* it is not possible to transmit any packets when max-weight is employed in this scenario.

Figure 30 further demonstrates how our algorithms treat flows with bad channel conditions. In particular, Figure 30 presents per-flow rate versus $p_{m,2}$ for the scenario of two-FIFO queues with four flows when $p_{n,1} = p_{n,2} = p_{m,1} = 0$ and $\beta = 2$ for (a) dFC and qFC and (b) max-weight. As seen, when $p_{m,2}$ increases, $\lambda_{m,2}$ decreases in Figure 30(a) since its channel is getting worse. Yet, this does not affect the other flows. In fact, $\lambda_{m,1}$ even increases as more resources are allocated to it when $p_{m,2}$ increases. On the other hand, both $\lambda_{m,1}$ and $\lambda_{m,2}$ decrease with increasing $p_{m,2}$ in max-weight (Figure 30(b)). This is not fair, because $\lambda_{m,1}$ decreases with increasing $p_{m,2}$ although its channel is always ON as $p_{m,1} = 0$. In the same scenario (Figure 30(b)), the rates of the nth queue ($\lambda_{n,1}$ and $\lambda_{n,2}$) increase with increasing $p_{m,2}$ as they use available resource opportunistically. This makes the total rate



(a) dFC and qFC (b) Max-weight

Figure 30. Per-flow rates versus $p_{m,2}$ for the scenario of two-FIFO queues with four flows when $p_{n,1} = p_{n,2} = p_{m,1} = 0$ and $\beta = 2$. (a) dFC and qFC. (b) Max-weight.

the same for dFC, qFC, and max-weight. Yet, as we discussed, max-weight is not fair to flow $\lambda_{m,1}$ in this scenario.

3.7 Conclusion

We investigated the performance of heterogeneous (per-flow and FIFO) queues over wireless networks and characterized the support region of this system for arbitrary number of queues and flows. We developed inner bound on the support region, and developed resource allocation schemes; dFC and qFC, which achieve optimal operating point in the convex inner bound. Simulation results show that our algorithms significantly improve throughput in a wireless network with per-flow and FIFO queues as compared to the well-known queue-based flow control and max-weight scheduling schemes.

CHAPTER 4

MANAGING HETEROGENEOUS TRAFFIC FOR INTERNET OF THINGS OVER CELLULAR NETWORKS

The contents of this chapters are based on our work that is published in the proceedings of 2019 IEEE LANMAN Conference [4]. ©2019 IEEE. Reprinted, with permission, from [4].

We consider a cellular network consisting of multiple IoT devices such as smart phones, tablets, connected vehicles, etc. A large number of applications run on those devices and have heterogeneous priority for data transmission. For example, a video streaming flow on the smart phone should have higher priority than background software updating flow on the conected vehicle because it is more sensitive to delay and throughput. In this context, we would like to design and implement a traffic management scheme for data transmission over cellular network, which should support heterogeneous traffic including high volumes of low-priority background traffic as well as high priority foreground traffic. In this part of the thesis, we develop (i) *Sneaker*, which is a in-network controller that yields to time-sensitive foreground traffic during periods of congestion and enables time insensitive background traffic to efficiently utilize any spare capacity, and (ii) *Legilimens*, which is low priority transport protocol for background traffic that only transmits data when there is sparse capacity and backs off when the network is congested by foreground traffic.

4.1 *Sneaker*: Managing Background Traffic in Cellular Networks

4.1.1 Background

In this era of ubiquitous cellular network connectivity, management of the scarce cellular bandwidth is important. A number of new applications (e.g., software updates, cloud sync) are starting to compete with existing applications (e.g., web browsing, video streaming) for cellular bandwidth. Therefore, it is critical to develop traffic management mechanisms that can handle large volumes of traffic with diverse characteristics.

One way to tackle the problem of managing such diverse traffic is to partition flows into two classes: *foreground* and *background* flows (although it is straightforward to extend the approach to more than two classes). In this chapter, we broadly consider regular mobile traffic like human communication or even some time-sensitive machine-to-machine (*M2M*) flows (*e.g.,* alarms, health monitoring) as foreground traffic, while treating large volume, time insensitive flows (*e.g.,* software downloads) as background traffic. Flow classification can be independent of applications running on- or off-screen at user devices, and it can be performed by either end hosts, network or content providers. Our goal then is to design a traffic management approach for cellular networks that prioritizes foreground traffic during periods of congestion while scheduling background traffic to effectively utilize any spare capacity. Further, because most large volume traffic flows on the downlink, our solution focuses only on the downlink.

There are existing in-network and end-to-end approaches to manage foreground and background traffic. Today's cellular networks provide support for prioritization using QoS Class Identifier (QCI). However, QCI suffers from several limitations: (1) identifies only at the granularity of devices or radio bearers, (2) offers only a handful traffic classes, many of which are reserved for operator-provided services like voice and VoLTE, and (3) uses static weights for the classes independent of network load. As an alternative, end-to-end, low-priority transport protocols (*e.g.,* TCP-LP [75]) perform well in wired and WI-FI networks but suffer in cellular networks due to per-device queues and schedulers as demonstrated in the following Section. To summarize, existing in-network approaches do not prioritize at flow granularity and existing end-to-end approaches are not effective in cellular networks.

To address the challenge of managing large-volume background traffic, we propose *Sneaker*, which achieves high network utilization and high throughput for background flows, without affecting foreground flows. The actual classification of traffic into foreground and background classes is beyond the scope of our work. *Sneaker* ensures that background flows quickly yield to foreground flows when the network is congested and quickly recapture spare capacity when the network becomes lightly loaded. Further, *Sneaker* seamlessly co-exists with existing schedulers and end-to-end protocols. Our *key* insight is that we can achieve the appropriate prioritization by randomly dropping background flow packets based on network load. Although our design is inspired by Random Early Detection (RED) [76], our goals and mechanisms are different from those of RED; *Sneaker* drops packets from queues, not based on overall congestion, but the load of foreground flows.

Our key design goals are to co-exist with end-to-end protocols (*i.e.,* TCP), without requiring changes to existing schedulers. To achieve these goals, we first study the interaction of TCP with common cellular schedulers. Then, we formulate the problem as a Network Utility Maximization (NUM) problem to determine the *optimal* transmission rate of background flows. Using this optimal transmission rate, we derive an optimal dropping rate for background flows. Because the optimal dropping rate is hard to realize in practice, we identify a close approximation to the optimal rate, which is easy to implement and works in harmony with end-to-end protocols. We show that our *practical* dropping rate avoids TCP timeouts for background flows and achieves the intended prioritization of foreground flows. Extensive ns-3 simulations confirm our analysis and show that *Sneaker* outperforms an aggressive baseline that gives strict priority to foreground traffic. Further, we also show that *Sneaker* performs better than existing low priority transport protocols. In summary, we make the following contributions:

- We analyze the complex interaction between TCP and cellular schedulers and characterize the average throughput of TCP for common cellular schedulers.
- We derive a dropping rate that satisfies our network objective of maximizing the performance of foreground traffic and **prove that the rate is optimal**.
- We develop a **practical dropping rate** that achieves prioritization and fairness among flows.
- We design *Sneaker* using the practical dropping rate and evaluate it via simulations in ns-3. We show that *Sneaker* significantly improves over in-network and end-to-end baselines in terms of prioritizing foreground flows over background flows and efficiently utilizing any spare capacity for background flows.

4.1.2 Related Work

Our work is closely related to the ideas from active queue management (AQM), interaction of TCP with cellular network and low priority data transmission.

Active queue management (AQM): AQM is a common approach to control data transmission rate in order to avoid congestion and improve network performance. One of the best known AQM schemes is Random Early Detection (RED) [77], which controls congestion by randomly dropping packets in the queue based on the average queue size information. This idea generated tremendous interest in congestion control and a lot of work has been done to improve the performance of RED based on local information such as queue dynamics and packet loss [78–81]. While our approach seems similar to RED, there are important differences. While the goal of RED is to manage congestion across all flows, ours is to achieve prioritization. We seek to design a scheduler that randomly drops packets based on the amount of foreground traffic.

Interaction of TCP with cellular network: TCP is not designed to work in cellular networks [82]. Thus the interaction of TCP with cellular network needs to be explored to better understand the performance of TCP flows. Due to highly variable delays on wireless links, spurious timeouts also occur in cellular network, which causes unnecessary re-transmissions and decreases throughput [83–86]. The scheduling algorithm in the base station also affects performance of TCP [87]. Compared to these works, our focus is to characterize the data rate of TCP by taking into account the specific factors in cellular networks and formulate the desired dropping rate at the base station based on the desired TCP data rate.

Low priority transport protocols: Researchers have studied several low priority transport protocols such as LEDBAT [88], TCP-LP [75] and TCP-Nice [89]. LEDBAT [88] and TCP-LP [75] use one-way delay as the congestion indicator and adjust congestion control accordingly. TCP-Nice uses RTT-threshold-based mechanism to indicate congestion [89]. The key idea of the current low-priority protocols is to detect congestion earlier than regular TCP. Compared to these works, the focus of our

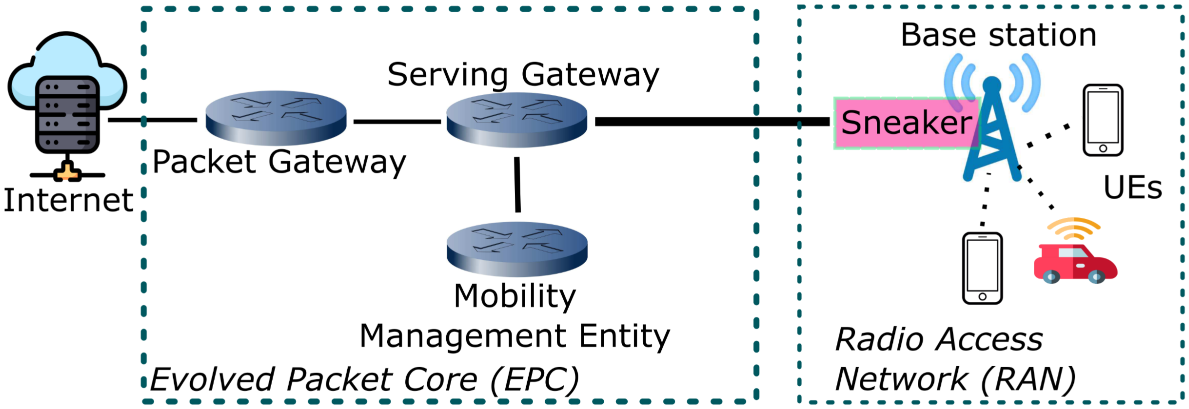


Figure 31. The service architecture in LTE network

work is not to design an end-to-end low priority transport protocol, but to design a packet dropping policy in queues to enable per-flow differentiation.

4.1.3 System Model

System Overview. We consider the general architecture of LTE cellular networks as shown in Figure 31, with two parts; evolved packet core (EPC) and radio access networks (RAN). The evolved packet core is a high-speed wired network that comprises the Mobility Management Entity (MME), the Serving Gateway (SGW), and the Packet Data Network Gateway (PGW). The LTE RAN includes the eNobeB (base station) and User Equipment (UE), which could be cellphones, tablets, connected vehicles, etc. Traffic from remote hosts in the Internet traverses through the packet core, arrives at the base station and eventually reaches the end users through wireless channels. In this setup, we develop *Sneaker* at base stations that works with existing TCP and cellular scheduling protocols and prioritizes foreground traffic over background.

Flows. We consider a system model depicted in Figure Figure 32, which represents the RAN from Figure Figure 31. There are N flows destined to K users within the same radio cell. M of these flows

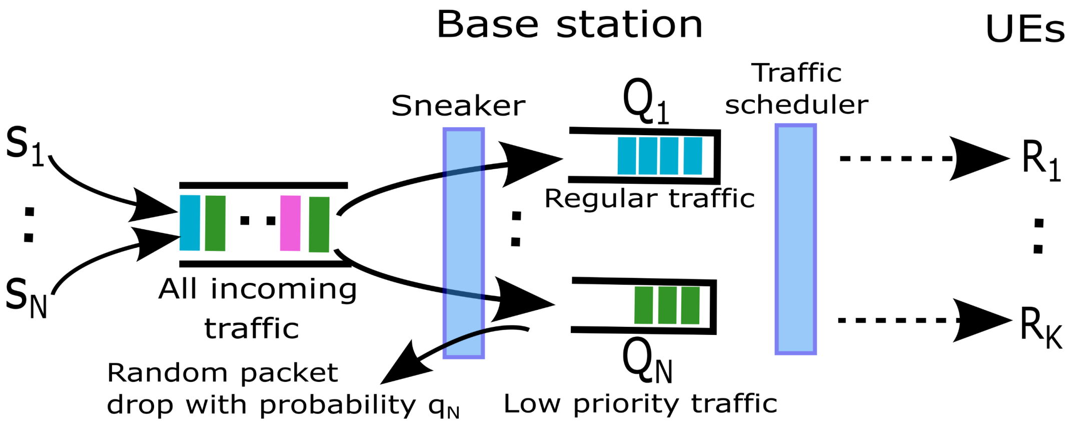


Figure 32. Example cellular system setup with *Sneaker*

are foreground, while L of them are background flows (*i.e.,* $N = M + L$). The set of all flows is $\mathcal{S} = \{S_1, \ldots, S_N\}$.

Queuing Model. At the base station, packets are queued in per-flow queues; $\{Q_1, \ldots, Q_N\}$. Packets from flow S_n are stored in queue Q_n. These queues operate according to the First-Come First-Serve (FCFS) rule.

Channel Model. We consider that base station back-haul links are high speed and lossless. The bottleneck of the system is the last hop radio channel; a radio frequency carrier shared by a set of cellular devices. We consider that $c_n(t)$ denotes the downlink channel capacity of device n at time t, which is the maximum achievable data rate based on the channel characteristics, as determined by the base station.

Scheduler. At the base station, time is divided into Transmission Time Intervals (TTIs), and each TTI is usually 1 ms in LTE [90]. The traffic scheduler determines which packets should be transmitted from the base station at a given TTI. Proportional Fair scheduler (PFS) is one of the widely used schedulers in today's cellular systems [91]. The other popular schedulers are Maximum Throughput (MT), Round Robin (RR), Blind Equal Throughput (BET), Throughput to Average (TTA), etc [92].

Problem Statement. Our goal is to develop a method that works with existing transport layer protocols and scheduling algorithms to prioritize foreground traffic over background. Our approach is to achieve the desired goal by randomly dropping packets coming into the base station, based on traffic type – foreground or background. In this context, the fundamental problem is to determine the optimal dropping rate. In this chapter, we determine the optimal dropping rate that forces background traffic to quickly yield to foreground traffic when the network is congested, but allows it to quickly recapture spare capacity when network load subsides.

4.1.4 Interaction of TCP with Cellular Networks

In this section, we characterize the average TCP sending rate in cellular networks.

Let the congestion window size of flow destined to user n at time slot t be $W_n(t)$. We assume that round-trip time (RTT) of each packet is constant, and equals to T_n. This is a common assumption in classical TCP analysis [93, 94]. Let $q_n(t)$ be the probability that packets from flow n are dropped from buffers due to overflow in the core network as well as the base station. Let $\rho_n(t)$ be the probability that packets from flow n are scheduled to be transmitted from the base station according to the underlying scheduling algorithm.

In this analysis, we ignore the slow start and time-out phases and only focus on the congestion avoidance phase of TCP, since the duration of congestion avoidance phase takes most of the TCP flow's lifetime. In congestion avoidance phase, at time $t - T_n$, $W_n(t - T_n)$ packets are transmitted from the source of TCP flow n. The ACKs corresponding to these packets, received between t and $t + T_n$, determine the window size update. In particular, for each transmitted and ACKed packet, window size is increased by $1/W_n$. For each packet dropped due to buffer overflow or delayed at the base station due to congestion, window size is reduced by $W_n(t)\beta$, where $0 < \beta < 1$. Thus, the window size evolves as follows: $W_n(t + T_n) = W_n(t) + I_n(t) - D_n(t)$, where $I_n(t) = W_n(t - T_n)\frac{1}{W_n(t)}(1 - q_n(t))\rho_n(t)$ is the increase in window size, and $D_n(t) = W_n(t - T_n)\beta W_n(t)(1 - (1 - q_n(t))\rho_n(t))$ is the decrease in window size.

The differential of the window size at time slot t is $\dot{W}_n(t) = (W_n(t + T_n) - W_n(t))/T_n$. The steady-state window size that satisfies $\dot{W}_n = 0$ becomes $W_n = \sqrt{\frac{(1-q_n)\rho_n}{\beta(1-(1-q_n)\rho_n)}}$. Thus, the steady-state TCP rate is formulated as

$$x_n^{TCP} = \frac{W_n B}{T_n} = \frac{B}{T_n}\sqrt{\frac{(1 - q_n)\rho_n}{\beta(1 - (1 - q_n)\rho_n)}}, \tag{4.1}$$

where B is the typical TCP packet size. TCP rate x_n^{TCP} depends on RTT, the scheduling probability at the base station, and the packet dropping probability in end-to-end path. As the bottleneck is usually the radio interface, we assume that packet drops (with probability q_n) only happen at the base station.

Our goal in this work is to determine the dropping rate q_n and actively drop packets from the queues (according to q_n) to prioritize foreground traffic over background. We characterize the optimal value of q_n in the next section.

4.1.4.1 Design of *Sneaker*

In this section, we derive the dropping probability for background traffic so that in the case of congestion, it does not compete with foreground traffic while keeping the TCP connection *alive* from avoiding costly timeouts, which hurt throughput.

4.1.4.2 NUM Formulation When Foreground and Background Flows Coexist

First, we formulate a network utility maximization (NUM) problem when foreground and background flows coexist. Let $\mathcal{L}$ and $\mathcal{M}$ denote the sets of background and foreground flows, respectively. We assume there are L and M background and foreground flows in the network. Let x_l and x_m denote the rate of the background and foreground flows, respectively. And let c_l, c_m denote the channel capacity of background flow user l and foreground flow user m, respectively. We can formulate the following NUM problem,

$$\begin{aligned} \max_{[x_1,\ldots,x_L]} \quad & \sum_{l=1}^{L} U_l(x_l) \\ \text{s.t.} \quad & \sum_{m\in\mathcal{M}} \frac{x_m}{c_m} + \sum_{l\in\mathcal{L}} \frac{x_l}{c_l} \leq 1 \\ & x_l \geq 0, \forall l \in \mathcal{L} \end{aligned} \tag{4.2}$$

where $U_l(\cdot)$ is the utility function associated with background flow l. The NUM formulation in Equation 4.2 optimizes the total utility of background flows assuming that there exist foreground flows. The first constraint is the time sharing constraint across foreground and background traffic flows. In this problem, we do not control the data rate x_m of regular flows. That is to say, x_m is given (*i.e.,* not an optimization parameter), which is controlled by end-to-end TCP congestion control mechanism. Note that we assume that $\sum_{m\in\mathcal{M}} \frac{x_m}{c_m} \leq 1$ since the data rate controlled by TCP will not exceed the capacity on average.

Theorem 6. *Assuming that we use* log*-based utility function (*i.e., $U_l(x_l) = \log(x_l)$*), which is widely used to provide proportional rate fairness, the optimal solution to the NUM problem Equation 4.2 is expressed as*

$$x_l^{OPT} = \frac{c_l}{L}[1 - \sum_{m\in\mathcal{M}} \frac{x_m}{c_m}], \forall l \in \mathcal{L} \tag{4.3}$$

where x_l^{OPT} depends on its channel capacity c_l, the number of background flows L, and the occupancy of the air interface by foreground traffic $\sum_{m\in\mathcal{M}}(x_m/c_m)$.

Proof: The optimization problem can be equivalently expressed as

$$\begin{aligned} \min_{[x_1,\ldots,x_L]} \quad & \sum_{l=1}^{L} -\log(x_l) \\ \text{s.t. } \tau_F + & \sum_{l=1}^{L} \frac{x_l}{c_l} \leq 1 \\ & x_l \geq 0, \forall l \in \mathcal{L} \end{aligned} \tag{4.4}$$

where $\tau_F = \sum_{m\in\mathcal{M}} \frac{x_m}{c_m}$.

It is notable that in Equation 4.4, the objective function and constraints are convex functions of x_l, thus this is a convex optimization problem. By KKT condition [95], we have

$$\begin{aligned} C1 \quad &: \nabla \left(\sum_{l=1}^{L} - \log(x_l) \right) + \lambda \nabla \left(\tau_F + \sum_{l=1}^{L} \frac{x_l}{c_l} - 1 \right) = 0 \\ C2 \quad &: \lambda \left(\tau_F + \sum_{l=1}^{L} \frac{x_l}{c_l} - 1 \right) = 0 \\ C3 \quad &: x_l \geq 0, \forall l \in \mathcal{L} \\ C4 \quad &: \lambda \geq 0 \end{aligned} \tag{4.5}$$

where λ is the multiplier variable and $C2$ is the complimentary condition.

By $C1$, we can obtain

$$-\frac{1}{x_l} + \lambda \frac{1}{c_l} = 0, \forall l \in \mathcal{L} \tag{4.6}$$

That is,

$$\lambda = \frac{c_l}{x_l}, \forall l \in \mathcal{L} \tag{4.7}$$

Substituting λ into the $C2$ of Equation 4.5 yields

$$\tau_F + \sum_{l=1}^{L} \frac{1}{\lambda} - 1 = 0 \tag{4.8}$$

That is,

$$\tau_F + \frac{L}{\lambda} - 1 = 0 \tag{4.9}$$

Thus, we have

$$\lambda = \frac{L}{1 - \tau_F} \geq 0 \tag{4.10}$$

Substituting this λ back to Equation 4.7 yields

$$x_l = \frac{c_l}{\lambda} = \frac{c_l}{L}(1 - \tau_F), \forall l \in \mathcal{L} \tag{4.11}$$

This completes the proof. $\square$

4.1.4.3 Optimal Dropping Rate

Now that we characterized the optimal and TCP rates of background flows (*i.e.,* x_l^{OPT} in Equation 4.3 and x_n^{TCP} Equation 4.1), we can design *Sneaker* by pushing the TCP rate to the optimal rate. The optimal dropping probability q_l^{OPT} that satisfies $x_l^{TCP} = x_l^{OPT}$ is expressed as

$$q_l^{OPT} = 1 - \frac{\gamma(c_l T_l(1-\tau_F))^2}{\rho_l(1 + \gamma(c_l T_l(1-\tau_F))^2)}, \forall l \in \mathcal{L}, \tag{4.12}$$

where $\tau_F = \sum_{m \in \mathcal{M}}(x_m/c_m)$, and $\gamma = \beta/(L^2 B^2)$. Therefore, q_l^{OPT} depends on channel capacity c_l, RTT T_l, the amount of foreground traffic $\tau_F = \sum_{m \in \mathcal{M}} x_m/c_m$, and the number of background flows L. Equation 4.12 also depends on ρ_l, the packet scheduling probability at the base station.

When foreground traffic congests the network, *i.e.,* τ_F approaches 1, and q_l^{OPT} approaches 1. This means that every packet from background flows would be dropped at the base station, which would cause TCP timeouts, and eventually *stop* the transmission. This is too harsh for background flows in practice. Therefore, we develop a practical dropping probability in the next section, which allows transmitting foreground traffic with higher priority yields to foreground traffic, yet allocates enough resources to keep prevents background flows from timing out.

4.1.4.4 Practical Dropping Rate

We develop a practical background flow rate based on the structure of the optimal rate in Equation 4.3 as $x_l^{PR} = \frac{c_l}{L}\max\{1-\tau_F, \epsilon\}$, where ϵ ($0 < \epsilon < 1$) is the minimum rate that should be allocated to background flows to keep them *alive*.

Similar to the optimal packet dropping rate, we set $x_l^{TCP} = x_l^{PR}$ and determine the practical drop rate q_l^{PR} as

$$q_l^{PR} = 1 - \frac{\gamma(c_l T_l \max\{1-\tau_F, \epsilon\})^2}{\rho_l(1+\gamma(c_l T_l \max\{1-\tau_F, \epsilon\})^2)}, \forall l \in \mathcal{L} \tag{4.13}$$

From Equation 4.13, the largest dropping rate for background flow l is $q_l^{PR} = 1 - \frac{\gamma(c_l T_l \epsilon)^2}{\rho_l(1+\gamma(c_l T_l \epsilon)^2)} \leq 1$, which happens when $1 - \tau_F \leq \epsilon$. Thus, even when foreground traffic is high, the background flows still get some resources for transmitting their packets. By tuning ϵ, we can adjust how much resources should be allocated to background traffic, hence aggressiveness of the background flows.

4.1.5 Implementation of *Sneaker*

4.1.5.1 Design Parameters and Signalling for *Sneaker*

4.1.5.1.1 Scheduling Probability

The packet dropping probability in Equation 4.13 is a function of ρ_l, which is the packet scheduling probability at the base station and depends on the scheduling algorithm. The packet scheduling probability can be measured by the scheduler and passed to *Sneaker* to calculate the packet dropping probability. In this section, we characterize the long term scheduling probability of Maximum Throughput (MT) scheduler and Proportional Fair (PF) scheduler. In this setup, we consider there are N users in the system. Let R_j be the instantaneous data rate of user j and μ_j be its throughput. Let $f_{R_j}(.)$ and $F_{R_j}(.)$ be the PDF and CDF of random variable R_j, respectively. For each user j, we assume its instantaneous data rate is an independently distributed and stationary random variable with mean $E[R_j]$, and its throughput μ_j is first-order wide sense stationary (WSS) with mean $E[\mu_j]$. We consider time is slotted, at each time slot t, user j's instantaneous data rate is denoted as $R_j(t)$ and its throughput up to time t is denoted as $\mu_j(t)$. We denote $S_j(t)$ as the random variable indicating whether user j is scheduled or not at time slot t. $S_j(t) = 1$ if user j is scheduled and $S_j(t) = 0$ otherwise. And we denote PS_j as the steady state probability of user j being scheduled to transmit data.

Theorem 7. *The steady-state packet scheduling probability of PFS for each flow approaches to $1/N$, where N is the number of users served by PFS.*

*Proof:*PF scheduler selects the user j^* to transmit data at each time slot t according to

$$j^* = \arg\max_j \frac{R_j(t)}{\mu_j(t)} \tag{4.14}$$

AT time slot t, given user j's instantaneous rate x, the probability that user j being scheduled is

$$Pr[S_j(t) = 1|R_j(t) = x] = Pr[\frac{x}{\mu_j(t)} > \frac{R_i(t)}{\mu_i(t)}], \forall i \neq j \tag{4.15}$$

It is shown in [96] that

$$\lim_{t\to\infty} Pr[S_j(t) = 1|R_j(t) = x] = \prod_{i\neq j} F_{R_i}(\frac{xE(\mu_i)}{E(\mu_j)}) \tag{4.16}$$

Thus, the steady state scheduling of user j is

$$\begin{aligned} PS_j &= \lim_{t\to\infty}\int_0^{+\infty} Pr[S_j(t)|R_j(t)=x]f_{R_j}(x)dx \\ &= \int_0^{\infty}\prod_{i=1,i\neq j}^{N} F_{R_i}(\frac{xE(\mu_i)}{E(\mu_j)})f_{R_j}(x)dx \end{aligned} \tag{4.17}$$

Furthermore, it is shown in [97] that in steady state, PF scheduler achieves

$$\frac{R_i}{E[\mu_i]} = \frac{R_j}{E[\mu_j]}\forall i \in \{1,2,\cdots,N\} \tag{4.18}$$

Therefore

$$\begin{aligned} F_{R_i}(\frac{xE(\mu_i)}{E(\mu_j)}) &= Pr[R_i \leq \frac{xE(\mu_i)}{E(\mu_j)}] \\ &= Pr[\frac{R_i}{E[\mu_i]} \leq \frac{x}{E[\mu_j]}] \\ &= Pr[\frac{R_j}{E[\mu_j]} \leq \frac{x}{E[\mu_j]}]\forall i \in \{1,2,\cdots,N\} \\ &= Pr[R_j \leq x]\forall i \in \{1,2,\cdots,N\} \\ &= F_{R_j}(x), \forall i \in \{1,2,\cdots,N\} \end{aligned} \tag{4.19}$$

Thus, Equation 4.17 can be further expressed as

$$\begin{aligned}
PS_j &= \int_0^{+\infty} \prod_{i=1,i\neq j}^{N} F_{R_j}(x) f_{R_j}(x)dx \\
&= \int_0^{+\infty} F_{R_j}^{N-1}(x) f_{R_j}(x)dx \\
&= \int_0^{+\infty} F_{R_j}^{N-1}(x) dF_{R_j}(x) \\
&= \frac{F_{R_j}^{N}(x)}{N} |_0^{+\infty} \\
&= \frac{1}{N}
\end{aligned} \tag{4.20}$$

This concludes the proof. □

Theorem 8. *The steady-state packet scheduling probability of MT scheduler for each flow j is*

$$PS_j = \int_0^{+\infty} \prod_{i=1,i\neq j}^{N} F_{R_i}(x) f_{R_j}(x)dx \tag{4.21}$$

Proof: Maximum throughput (MT) scheduler selects the user with the largest instantaneous data rate in current TTI to transmit data. Given the data rate of user j at time slot t, the probability of user j being scheduled is

$$Pr[S_j(t) = 1 | R_j(t) = x] = Pr[x > \max_{i\neq j, i\in\{1,2,\cdots,N\}} \{R_i(t)\}] \tag{4.22}$$

Therefore, the long term scheduling probability of user j is

$$\begin{aligned} PS_j &= \lim_{t\to\infty}\int_0^{+\infty} Pr[S_j(t)=1|R_j(t)=x]f_{R_j}(x)dx \\ &= \lim_{t\to\infty}\int_0^{+\infty} Pr[x > \max_{i\neq j, i\in\{1,2,\cdots,N\}}\{R_i(t)\}]f_{R_j}(x)dx \\ &= \int_0^{+\infty}\prod_{i=1,i\neq j}^{N} Pr[x>R_i]f_{R_j}(x)dx \\ &= \int_0^{+\infty}\prod_{i=1,i\neq j}^{N} F_{R_i}(x)f_{R_j}(x)dx \qquad (4.23) \end{aligned}$$

This concludes the proof. □

Corollary 9. *For a two-user system with R_j ($j=1,2$) following Rayleigh distribution, the scheduling probability of user j is*

$$PS_j = \frac{\sigma_j^2}{\sigma_1^2+\sigma_2^2}, j=1,2 \qquad (4.24)$$

Proof: Let σ_1 and σ_2 denote the mode of R_1 and R_2, respectively. Thus, we have

$$f_{R_j}(x) = \frac{x}{\sigma_j^2}e^{-\frac{x^2}{2\sigma_j^2}}, j=1,2 \qquad (4.25)$$

and

$$F_{R_j}(x) = 1-e^{-\frac{x^2}{2\sigma_j^2}}, j=1,2 \qquad (4.26)$$

According to Equation 4.23, the steady state scheduling probability of user 1 is

$$\begin{aligned}
PS_1 &= \int_0^{+\infty} F_{R_2}(x) f_{R_1}(x) dx \\
&= \int_0^{+\infty} (1 - e^{-\frac{x^2}{2\sigma_2^2}}) \frac{x}{\sigma_1^2} e^{-\frac{x^2}{2\sigma_1^2}} dx \\
&= \int_0^{+\infty} \frac{x}{\sigma_1^2} e^{-\frac{x^2}{2\sigma_1^2}} dx - \int_0^{+\infty} \frac{x}{\sigma_1^2} e^{-x^2(\frac{1}{2\sigma_1^2}+\frac{1}{2\sigma_2^2})} dx \\
&= 1 - \frac{1}{\frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}} \frac{1}{\sigma_1^2} \int_0^{+\infty} (\frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}) x e^{-x^2(\frac{1}{2\sigma_1^2}+\frac{1}{2\sigma_2^2})} dx \\
&= 1 - \frac{1}{\frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}} \frac{1}{\sigma_1^2} \\
&= \frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2} \qquad (4.27)
\end{aligned}$$

Similarly, we can obtain that

$$PS_2 = \frac{\sigma_2^2}{\sigma_1^2+\sigma_2^2} \qquad (4.28)$$

This concludes the proof. □

4.1.5.1.2 Local Signalling

The packet dropping rate in Equation 4.13 depends on scheduling parameters such as the time share of foreground traffic τ_F, capacity of background user c_l, and the number of background flows L.

We approximate the time share of foreground traffic as $\tau_F \approx \sum_{m \in \mathcal{M}} (Q_m(t)/R_m(t))$, where $Q_m(t)$ is the queue size of foreground user m in the base station at TTI t, and $R_m(t)$ is the maximum number of packets that can be transmitted from foreground flow user m at TTI t. The main idea behind this approximation is that both x_m/c_m and $Q_m(t)/R_m(t)$ are time shares, and we conjecture that the time average of $Q_m(t)/R_m(t)$ approaches x_m/c_m. $Q_m(t)$ and $R_m(t)$ are collected by existing scheduling algorithms (such as PFS) and passed to *Sneaker*. Similarly, the scheduler passes $R_l(t)$ information to *Sneaker*, and we make $c_l \approx R_l(t)$ approximation.

4.1.5.1.3 End-to-end Signalling

As for the access of other parameters, in our implementation, sender adds one bit as a tag in its TCP header to mark its classification (foreground or background). Sender also piggybacks the RTT information into its TCP header. At the base station, *Sneaker* extracts TCP header to obtain RTT and classification of the flow. In this way, *Sneaker* obtains T_l and learns if a flow is a background flow (hence the total number of background flows L).

4.1.5.2 Implementation of *Sneaker* on ns-3

We implemented and evaluated *Sneaker* using the ns-3 simulator [98]. We build on top of existing LTE protocol stack shown in Figure 33, where data packets are buffered in Radio Link Control (RLC) layer after passing through Packet Data Convergence Protocol (PDCP) layer. MAC layer reads packets in RLC buffers depending on the scheduling algorithm used. *Sneaker*, implemented at eNodeB as a slim layer on top of PDCP, inspects every incoming packet, and extracts end-to-end information. It also gets local signalling data from RLC and MAC layers to calculate the packet dropping probability using Equation 4.13. Packets are dropped by *Sneaker* according to the calculated packet drop probability before arriving to PDCP. Note that we also make minimal updates to TCP so that packets carry end-to-end information.

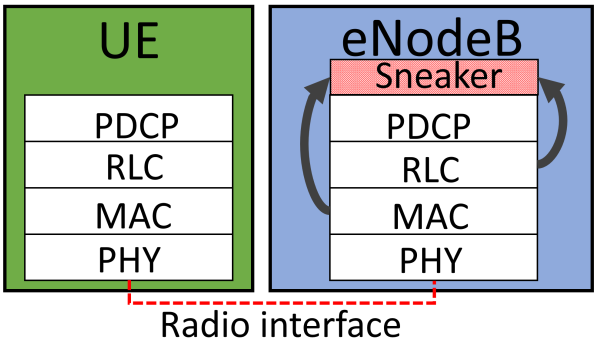


Figure 33. LTE protocol stack

4.1.6 Evaluation of *Sneaker*

We evaluate the performance of *Sneaker* through ns-3 simulations. Our simulation topology is shown in Figure 31. It consists of multi-hop wired links that connect remote servers to Packet Gateway using 1 Gbps, 10 ms delay links. The link speed between the base station and packet and service gateways is 300 Mbps. The base station is configured with 751 MHz downlink band with 10 MHz bandwidth and 50 resource blocks, MIMO transmission, transmission power of 47.78 dBm, and RLC buffer size is 512 KB. The path loss model is log distance propagation model with loss exponential parameter of 3.52.

Prioritization. In this experiment, we verify if *Sneaker* achieves the correct prioritization between background and foreground flows. We use two remote senders to send traffic to two different end users. While the first sender continuously sends background traffic to a user, the second sender sends foreground traffic in an on-off pattern every 10 seconds. All senders share a cellular link with 70 Mbps and use TCP Reno. We compare *Sneaker* with the following baselines; `PFS only`, `RED`, `Strict Priority`. All these mechanisms (including *Sneaker*) use TCP-Reno and Proportional Fair Scheduler (PFS). *Sneaker* drops packets according to Equation 4.13, `PFS only` does not drop packets actively,

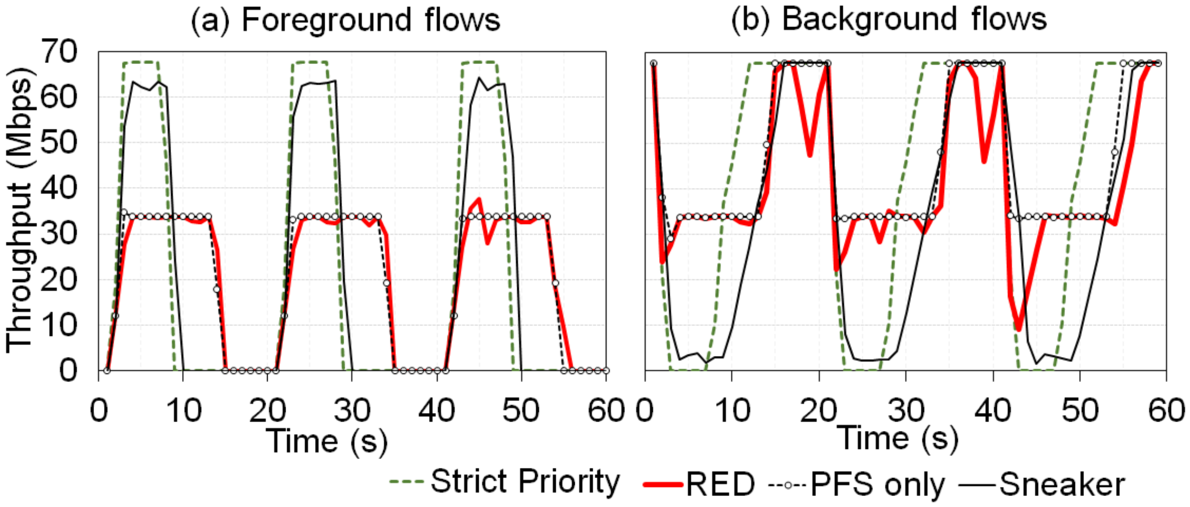


Figure 34. Comparison of different approaches to prioritization

`RED` drops packets according to the policy in [99], and `Strict Priority` drops every packet from background flow if there exists foreground traffic.

Figure 34 shows the throughput achieved by both foreground and background flows. *Sneaker* and `Strict Priority` work as expected, *i.e.,* yield to foreground traffic, and serve background traffic if there is no foreground traffic. On the other hand, `PFS only` and `RED` do not provide such ability. We prefer *Sneaker* over `Strict Priority` as `Strict Priority` completely blocks background traffic until foreground traffic finishes and causes timeouts and connection disconnects, which is not ideal.

We also study the impact of increasing number of background flows on foreground traffic rate. In this experiment, we send a foreground flow to one user. Simultaneously, we have a different number of background flows to another user connected to the same base station. We show the average throughput achieved by the foreground flow with `PFS only` and *Sneaker* in Figure 35. As shown, the throughput

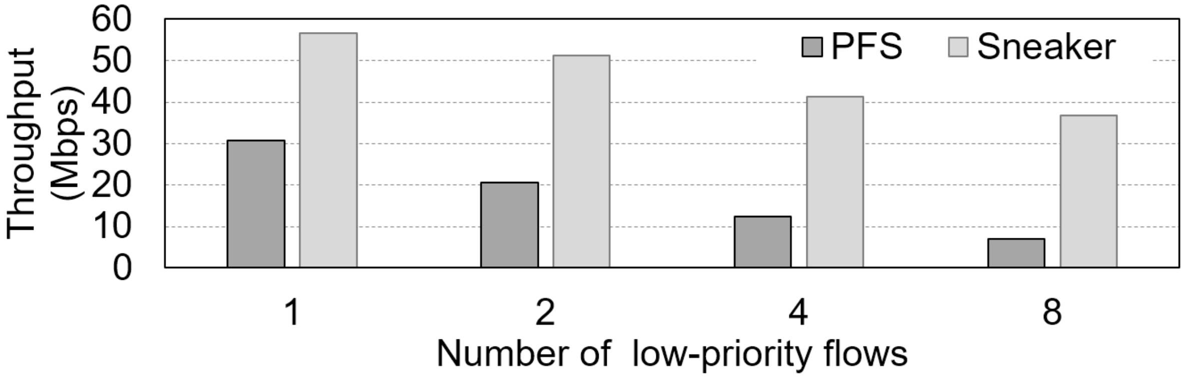
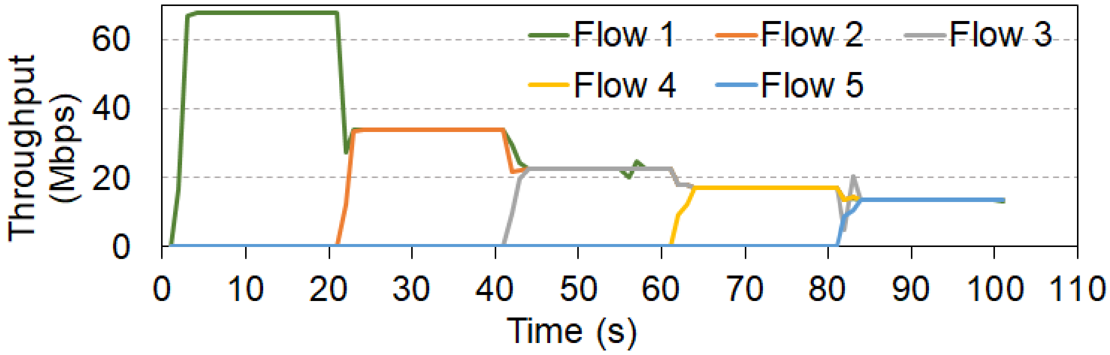


Figure 35. Foreground flows benefit from *Sneaker*

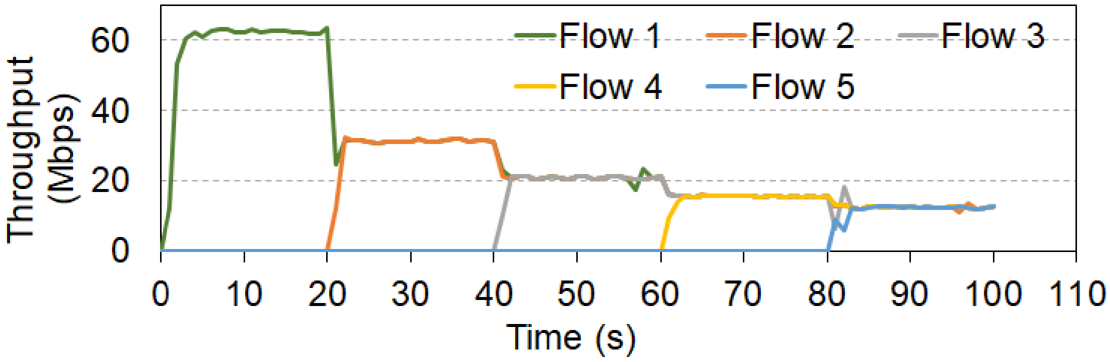
of foreground flows degrades as the number of background flows increases, as expected. However, the degradation is significantly smaller with *Sneaker* than it is with `PFS only`, which shows that *Sneaker* is able to better isolate foreground flows. With *Sneaker*, the foreground flow achieves 1.8 times higher throughput than with `PFS only`; with *five-fold* throughput gain as the number of background flows increases to 8.

Fairness. We want to verify: *fair sharing of spare bandwidth among background flows*, and *fair sharing of available bandwidth among foreground flows*. We consider two scenarios: (i) five background flows without any foreground flow, and (ii) five foreground flows with one background flow. In both scenarios, the new flows join every 20 seconds. Figure Figure 36(a) shows the throughput achieved by background flows using *Sneaker* in the first scenario. We see that the background flows quickly *converge* to their fair-share throughput as new flows are added. Figure Figure 36(b) shows that rate is fairly shared by foreground flows in the second scenario using *Sneaker*.

*Large-scale simulations.*We consider realistic workloads and create a large topology with 100 users that connect to a base station. We generate a mixed traffic with short (64 KB), medium (1 MB), and long flows (32 MB). Short flows generate 10% of the overall foreground traffic, whereas medium flows gen-



(a) Fairness among background flows



(b) Fairness among foreground flows

Figure 36. Fairness among background and foreground flows

erate 40% of the load; the rest (*i.e.,* 50% of load) comes from long flows. While we generate foreground traffic mix, we send a continuous background (long) flow. We compare *Sneaker* (with background flow using TCP-Reno) with `TCP-LP` and `LEDBAT`, the known end-to-end congestion control mechanisms designed to yield to foreground traffic. We simulate traffic for 120 seconds and compare the throughput achieved by foreground and background flows in all the schemes, with `PFS` as the default scheduler.

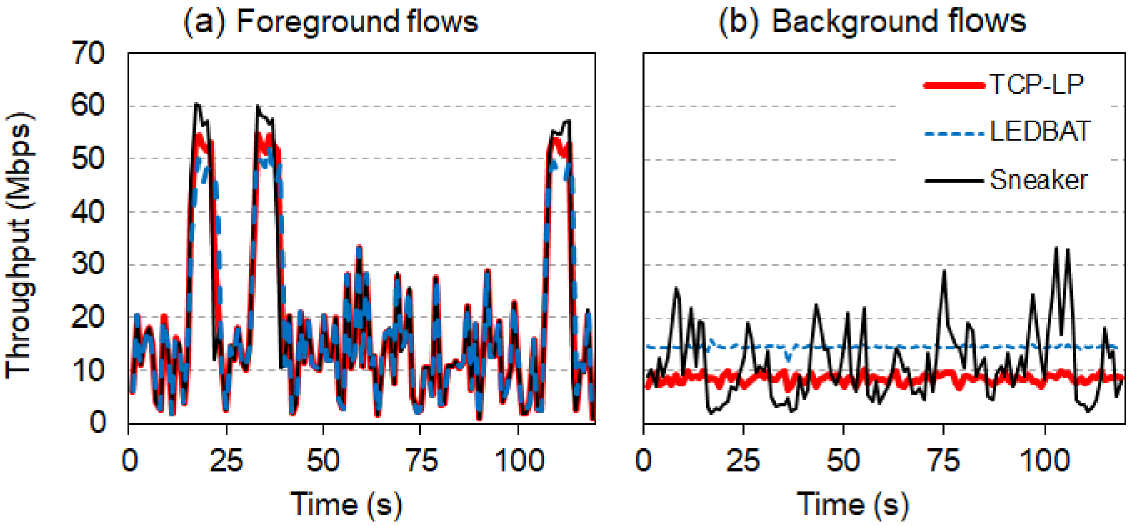


Figure 37. Large-scale results at moderate load (50%)

We first evaluate the performance of the three schemes under moderate load of 50%. For this study, we set the overall network load due to foreground flows to 50%. Figure 37 shows the throughput of foreground and background flows. We observe the bursty nature of foreground traffic under all schemes, as expected. For foreground flows, `TCP-LP` achieves better throughput than `LEDBAT`, and *Sneaker* achieves the best throughput. On the other hand, the throughputs achieved by background flows drastically differ between the three schemes. `LEDBAT` and `TCP-LP` does not yield to foreground flows, which is not desirable. In contrast, *Sneaker* modulates its throughput to allow foreground traffic to use most of the available capacity while effectively utilizing the spare capacity.

We consider the same experiment under low load (20%). Figure 38 shows that all the schemes achieve similar throughput for foreground flows, as expected. However, their background flow rates differ significantly. While `TCP-LP` and `LEDBAT` are unable to fully utilize the spare capacity at low

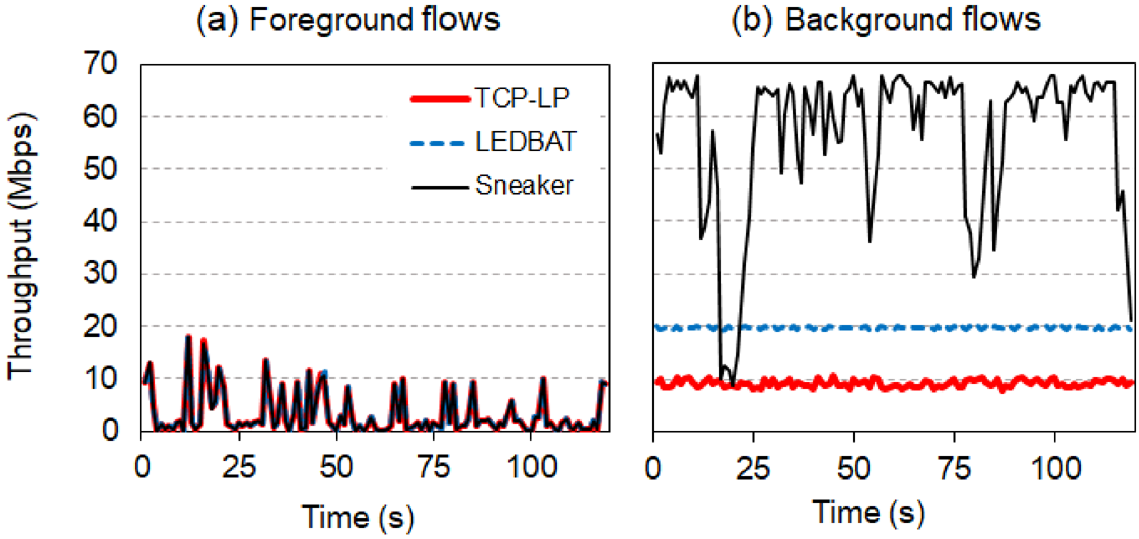


Figure 38. Large-scale results at low load (20%)

loads (*i.e.,* they achieve lower throughput), *Sneaker* efficiently utilizes the spare capacity and achieves much higher throughput.

4.1.7 Conclusion

Existing in-network and end-to-end mechanisms for per-flow prioritization do not work well over scheduled links in cellular networks. We presented *Sneaker*, an in-network arbiter that provides both high-performance and fairness among (within) foreground and background flows.differentiation between foreground and background flows, while enabling high performance and maintaining fairness within each traffic class. We have formulated the problem of per-flow prioritization using NUM framework and shown that *Sneaker* achieves the desired optimality. Further, we have extensively evaluated our design using both targeted small-scale simulations and realistic large-scale simulations.

4.2 *Legilimens*: An Agile Transport for Background Traffic in Cellular Networks

Large data transfers can result in significant congestion and performance degradation for users of typical applications such as web browsing, streaming, and other interactive applications. While there are existing TCP congestion control algorithms for delivery of large volume data (*e.g.,* LEDBAT, TCP-LP), our results show that these protocols are not effective in cellular networks due to variability in radio channel conditions and the use of proportional fair (PF) schedulers in cellular base stations. We propose *Legilimens*, an agile TCP variant for cellular downlink transfers, which has the properties of existing approaches, but addresses the challenges of cellular networks by identifying novel mechanisms that exploit the properties of the PF scheduler to estimate load and capacity. As a result, *Legilimens* is able to deliver traffic using only the *spare* capacity on the downlink. We conduct extensive evaluations of *Legilimens* in multiple settings — in a large cellular network for real-world performance, on the PhantomNet emulator for controlled experiments, and `ns-3` simulator for scaled experiments — all of which demonstrate that *Legilimens* is superior to other protocols in transferring large volumes of data without interfering with regular user traffic. Compared to CUBIC, *Legilimens* improves the throughput of competing long flows by up to 117%, and flow completion time of short flows by up to 48%.

4.2.1 Background

Cellular networks are increasingly used not just for popular end-user applications like web browsing, chat, social networks and video streaming, but also to transfer large volumes of data, *e.g.,* through cloud data sync and software updates [100] (*e.g.,* cloud data sync and software updates [100]). Large data transfers can negatively impact interactive user applications by saturating downlink capacity, in both the congested and neighboring cells. Cellular links from base stations to client devices are usually the bottleneck [101], making techniques that allow for efficient delivery of large data transfers without impacting users' experience a necessity.

There exist several possible approaches to alleviate the issue. Some services, applications and mobile operating systems allow users to restrict or schedule transfers. Unfortunately, these approaches have limited impact because the approaches are agnostic of prevailing network conditions and users may prefer the convenience of up-to-date data. Rate-limiting specific flows is also possible. But naive rate-limiting can cause under-utilization at low loads and overwhelm the network at high loads, while more sophisticated versions tend to be complex and expensive to realize. Existing traffic management for cellular networks, in a form of QoS class identifier (QCI), is not sufficient due to the need for tight integration between network and each application, must be provisioned a priori by operators, cannot dynamically change weights for flows based on load, and has minimal flexibility to move flows between classes.

Transport-layer approaches, exemplified by protocols like LEDBAT and TCP-LP, introduce the concept of *two-class* service prioritization. The key idea is to have a "low-priority" mechanism for delivering large volume or time-insensitive data. This allows for typical user interactive applications (considered foreground flows) to have fast response times using a *fair-share* TCP variant like RENO or CUBIC, while simultaneously making progress on large volume transfers (considered background flows) using the lower priority TCP variant. Our experiments with such Low Priority Transport (LPT) protocols show that they are not effective in cellular networks. We conjecture that this is due to the nature of cellular links, in particular highly variable delays, and the use of *proportional fair* (PF) schedulers and per-device queues for downlink transmissions.

Drawing inspiration from LPT protocols, we propose *Legilimens*, an agile LPT variant for cellular networks that delivers background traffic without adversely affecting foreground traffic. *Legilimens* has the main properties of other LPT protocols — to use all available bandwidth when no other traffic is present, and to yield quickly to standard TCP flows that share the same bottleneck link — but includes novel features that make it effective in cellular networks. Specifically, a well-designed LPT protocol for cellular networks must operate with minimal queuing, so that its packets do not compete with those of foreground flows at the scheduler.

We focus on optimizing *Legilimens* for cellular downlink traffic in this chapter. To that end, we design a novel algorithm that quickly estimates capacity and load based on packet inter-arrival times, not RTT or OWD. Our central *idea* lies in leveraging the downlink PF scheduler's unique *strength* of providing fairness at short time scales to counteract its *weakness* of interfering with the operation of traditional LPT protocols. Based on the estimated capacity and load, *Legilimens* strives to deliver background traffic using *only* spare capacity. *Legilimens* operates in one of two modes: normal mode and probing mode. If the load is low and spare capacity is available, *Legilimens* operates in normal mode and quickly captures available bandwidth. If the load is substantial, *Legilimens* enters the probing mode, in which it yields all scheduling opportunities to other traffic most of the time while periodically sensing the network until spare capacity becomes available.

We implement *Legilimens* in Linux as a sender-only modification to the network stack, enabling simpler and incremental deployment, without changes to the cellular infrastructure. We first sketch a practical deployment model and then we extensively evaluate *Legilimens* in realistic (uncontrolled) and controlled settings, including a large U.S. cellular network, using a PhantomNet emulator and `ns-3` simulator.

In summary, we make the following contributions:

- We design a novel capacity and load inference algorithm for senders in cellular networks, which overcomes and leverages PF scheduling properties that impede traditional LPT protocols and we show that our algorithm is robust to sender- and receiver-side optimizations (*e.g.,* batching, delayed `ACKs`).
- We design and implement *Legilimens*, a sender-side LPT protocol for cellular networks, which allows the network to balance the conflicting goals of foreground and background traffic while achieving high utilization and low congestion.
- We demonstrate that *Legilimens* achieves better overall performance than existing fair-share and LPT protocols. For example, in a real network, compared to CUBIC, *Legilimens* improves the throughput of foreground long flows by up to 117%, and flow completion time of short flows by up to 48%. We summarize our quantitative results in Table II.

4.2.2 Related Work

Legilimens is broadly related to fair-share protocols and low priority protocols. Our capacity estimation algorithm is somewhat related to existing work on bandwidth estimation but our algorithm estimates busyness as well.

Fair-share protocols fall into two categories: loss-based and delay-based. RENO [102], NEWRENO [103], TAHOE [104]), and CUBIC [105] are some of the well-known loss-based TCP variants. Most of the conventional, loss-based TCP variants, incur high queuing delays due to “buffer bloat” in cellular networks [106]. Being less aggressive than loss-based protocols, delay-based protocols (*e.g.,* VEGAS [107]) do not suffer as much due to buffer bloat in cellular networks. Nevertheless, fair-share protocols do not distinguish foreground flows from background flows, and therefore, foreground performance degrades at high loads.

We have extensively discussed the shortcomings of low priority TCP variants, including TCP-LP [108], LEDBAT [109], and NICE [110]. Although the low priority protocols perform well in wired and Wi-Fi networks, they do not perform well in cellular networks due to the nature of cellular links [82–86] and due to the presence of the cellular scheduler.

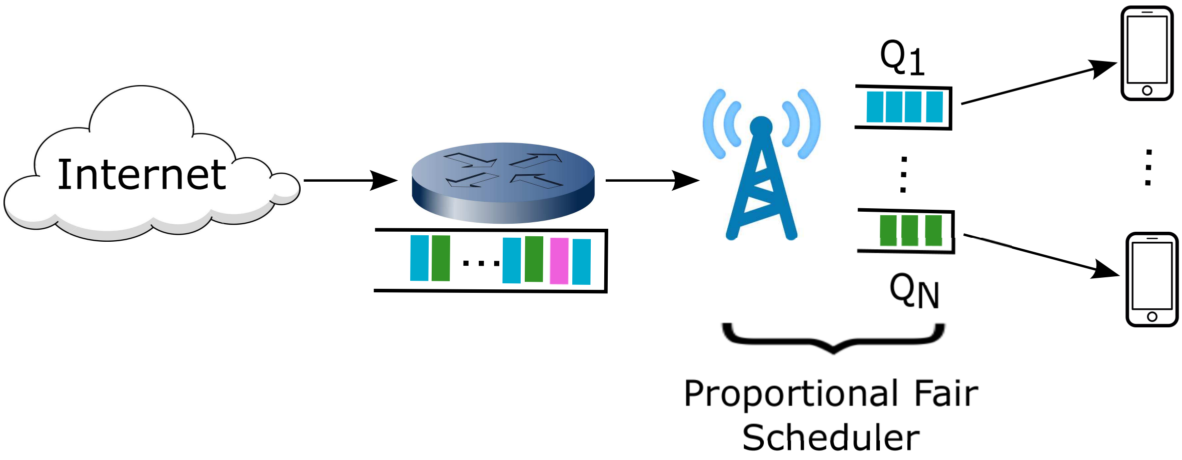
Recent proposals address the time-varying capacity of cellular links. Sprout [111] and Verus [112] build a model for accurately predicting the “best” congestion window by observing packet arrivals. BBR [113] uses *both* bandwidth estimation and round trip times to find an operating point that maximizes throughput. PropRate [114] uses one-way delay to achieve a desired trade-off between latency and throughput. While these newer protocols improve performance in cellular networks, their underlying goal is to achieve fair share of bottleneck capacity. In contrast, our work focuses on efficiently utilizing spare capacity without affecting foreground flows. Our evaluations show that BBR [113], when used for background flows, suffers in foreground performance, when compared to *Legilimens*. Loadsense [115] schedules background traffic based on *passive* estimation of cellular load by observing the power of channel and pilot signal. However, passive estimation requires support at clients. In contrast, *Legilimens* performs active measurements and does not require support at clients.

There is a large body of work on capacity estimation. While several of these ideas [116–120] are applicable to a broader class of networks, our algorithm optimizes for the specific case of PF schedulers in cellular networks, which enables us to be simple and efficient (e.g., we use data packets for probing, so there is no overhead). QProbe [121] leverages the scheduler to estimate congestion at the base station, somewhat similar to *Legilimens*. However, QProbe uses the estimation to identify bottleneck links, whereas *Legilimens* uses the estimation to schedule background traffic. ExLL's [122] capacity estimation bears some similarity to our estimation. But, ExLL is a fair-share protocol. To the best of our knowledge, none of the existing papers estimate *busyness*, which is central to *Legilimens*'s goal of prioritizing foreground flows over background flows.

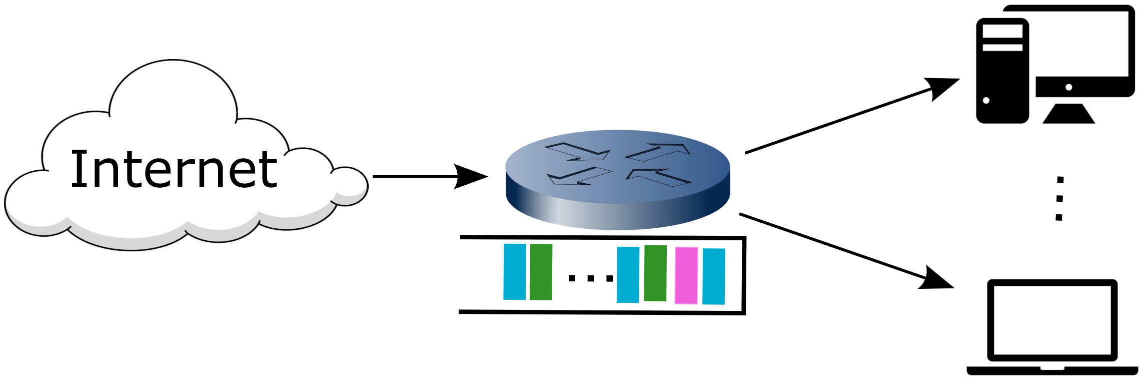
4.2.3 Challenges

4.2.3.1 The Proportional Fair (PF) Scheduler

Figure 39 depicts the key infrastructural differences between wired/Wi-Fi networks and cellular networks. While the actual scheduler implementations in today's cellular networks are vendor specific and some parameters might be different, we present a generalized version of the PF scheduler [111,123]. Consider a particular radio cell in a typical cellular network as shown in Figure 39(a). There are N clients connected to this cell, with a queue constructed for each client. Time is divided into 1 ms intervals, referred to as Transmission Time Intervals (TTI), each consisting of 2 slots. The system bandwidth is split into many sub-carriers in the frequency domain. A bundle of sub-carriers plus a TTI is referred to as a Physical Resource Block (PRB). At each TTI, the base station assigns all or part of resource blocks to a client.



(a) Cellular network with per-device queues



(b) Wired or Wi-Fi network with shared queues

Figure 39. Cellular vs. wired or Wi-Fi networks

The base station selects the client n^* to transmit data according to

$$n^* = \arg \max_{n \in \{1, \cdots, N\}} \frac{c_n(t)}{\mu_n(t)} \tag{4.29}$$

where $c_n(t)$ is the maximum transmission rate to client n at time slot t, a function of Signal to Interference+Noise Ratio (SINR), and $\mu_n(t)$ is the average throughput between the base station and client n up to TTI t [123]. The average throughput of user n is updated according to

$$\mu_n(t+1) = (1 - \frac{1}{\alpha})\mu_n(t) + \frac{1}{\alpha} r_n(t) \tag{4.30}$$

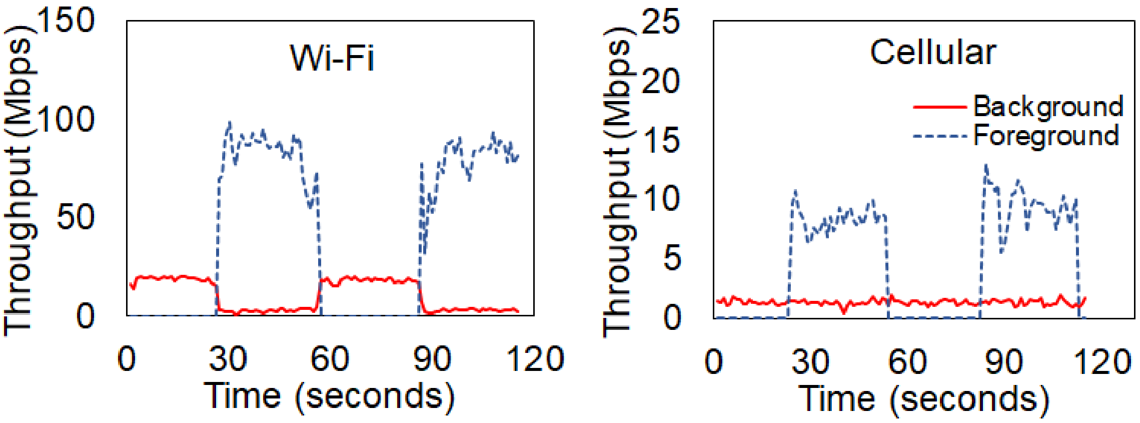
where α is a small constant and $r_n(t)$ is the data transmission rate of user n at time slot t.

The base station allocates all resource blocks for the transmission of client n^* packets. However, if there are not enough packets in the base station for client n^*, the resource blocks are shared, *i.e.,* other clients are scheduled using the same rule by giving preference to clients with larger $\frac{c_n(t)}{\mu_n(t)}$. Finally, the base station updates the average throughput $\mu_n(t)$ using exponential moving average.

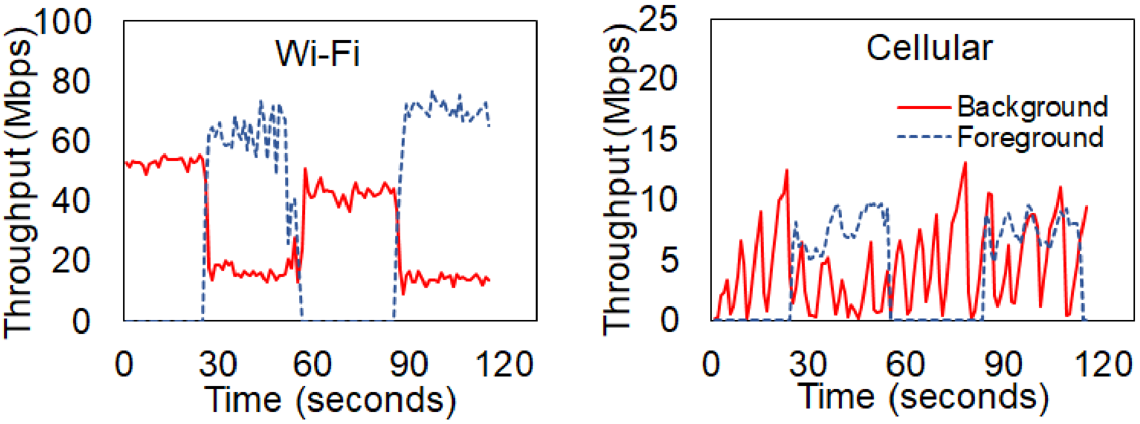
4.2.3.2 LEDBAT and TCP-LP in Cellular

LEDBAT [88] and TCP-LP [75] are two popular LPT protocols. The key idea of these background transport protocols is to detect congestion earlier than regular TCP using use one-way packet delays [88, 124]. We performed a simple experiment to study the performance of LEDBAT and TCP-LP in Wi-Fi and cellular networks.

We use a simple Wi-Fi test-bed, where two clients connect to an access point. The access point is connected to a server using a high speed wired connection. There are two flows in the network; one for each client. The first flow is a persistent long flow that uses LEDBAT, representing the background flow; the second flow uses CUBIC, representing the foreground flow. The foreground traffic is an on-off traffic; the flow joins the system at every 1 minute mark, transmits for 30 seconds and sleeps for 30



(a) LEDBAT



(b) TCP-LP

Figure 40. Behavior of existing protocols

seconds. We repeat the experiment for a cellular network, in which the two clients connect to a base station, instead of a Wi-Fi access point.

Figure 40(a) shows the throughput of foreground flow (using CUBIC) and background flow (using LEDBAT) for Wi-Fi (Figure 40(a)(left)) and cellular (Figure 40(a)(right)) test-beds. While LEDBAT shows expected performance in Wi-Fi, it performs poorly in cellular (i.e., the background flow does not use spare capacity when foreground flow is not present). Figure 40(a) shows the results when we use TCP-LP for the background flow. While TCP-LP also performs well in Wi-Fi, it competes with and take away significant capacity from the foreground flow in the cellular network. Therefore, we conclude that existing LPT protocols achieve sub-optimal performance in cellular networks.

In wired and Wi-Fi networks, delay-based background transport protocols perform well as all traffic passes through the same bottleneck queues and the delay includes the aggregate congestion (queuing) on the path. However, cellular networks use per-device queues and the scheduler provides opportunity for all flows during each scheduling interval. Therefore, the delay would not increase as steeply as it would in wired and Wi-Fi networks during congestion. Further, because the PF scheduler determines which queue to serve based on signal quality and recent throughput history, the delay includes the effect of additional factors. Lastly, the propagation delay in cellular networks is highly variable, which further perturbs the delay and compromises the accuracy of delay-based congestion estimation in these protocols [111, 112]. Therefore, the measured delay in LEDBAT and TCP-LP does not fully capture the overall congestion of the radio cell, and offers limited insight into the cell capacity.

4.2.4 *Legilimens*

4.2.4.1 High-level Overview

We propose *Legilimens* for background applications to utilize spare capacity without significantly affecting other foreground traffic. We provide a high-level overview of *Legilimens* in Figure 41. *Legilimens* estimates capacity and cell load and sends the background traffic to fill the spare channel capacity. This is the *normal* operation mode. *Legilimens* uses AIMD-style congestion control during normal mode.

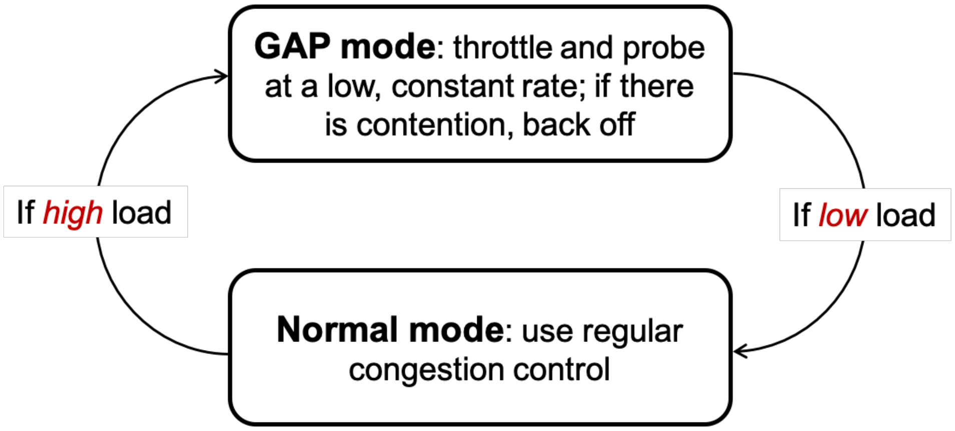

GAP mode: throttle and probe at a low, constant rate; if there is contention, back off
If *high* load
If *low* load
Normal mode: use regular congestion control

Figure 41. High-level overview

On the other hand, if the load estimate indicates that there is increased competition, *Legilimens* enters a dormant mode (i.e., GAP mode in Figure 41) during which the flow waits and yields to the foreground traffic. While waiting, *Legilimens* intermittently sends short *bursts* of packets to probe for spare capacity. Because we use regular data packets during probing, there is no bandwidth overhead. Choosing the correct burst size for probing is non-trivial – while large bursts provide more accurate estimates, they degrade the performance of foreground flows at high loads. Therefore, instead of employing a single burst size, we start with the small burst and gradually increase it upon detecting lower load, as opposed to sending one large burst. We call this the *gradually aggressive probing* (GAP) mode. The GAP mode prevents spurious oscillations to normal mode and back, and serves as hysteresis between the modes.

We require three key mechanisms to realize the design from Figure 41. First, we need to estimate capacity, so we can provide room for transient bursts of foreground traffic and to establish a bound on sending rate. Second, we need an agile mechanism to detect the load level (i.e., there is other traffic).

Finally, we need a robust and efficient congestion control scheme to achieve both high performance for all applications and fairness among background applications.

The cellular network uses different scheduling mechanisms for downlink vs. uplink. As a result, we need different techniques to estimate capacity and busyness in the two scenarios. In this chapter, we focus only on techniques for downlink and leave the identification of techniques for uplink as part of our future work. Also, we constrain our design to not require network- or receiver-side changes to be deployment friendly.

4.2.4.2 Estimating Capacity and Busyness

4.2.4.2.1 Intuition: Optimal Low Priority Data Rate

Let us consider that every low priority user regards itself as the only low priority user in the system and considers all other users as regular users. To obtain the optimal data rate for low priority user l, we can formulate a network utility maximization problem as following,

$$\begin{aligned} &\max_{x_l} \; U_l(x_l) \\ &\text{s.t.} \sum_{n\in\mathcal{N}} \frac{x_n}{c_n} + \sum_{j\in\mathcal{L}, j\neq l} \frac{x_j}{c_j} + \frac{x_l}{c_l} \leq 1 \\ &\quad\; x_l \geq 0 \end{aligned} \tag{4.31}$$

where the objective function is the utility of low priority flow l, $\mathcal{N}$ and $\mathcal{L}$ are the set of regular users and low priority users, respectively, and x and c is the user rate and channel capacity. The first constraint is the time sharing constraint across regular and low priority users. We note that only the data rate x_l is a variable here.

Assuming that we use $\log$ utility function (i.e., $U_l(x_l) = \log(x_l)$), which is widely used to provide proportional rate fairness, by KKT condition, we can obtain the compact solution to the NUM problem as

$$x_l = c_l(1 - \sum_{n \in \mathcal{N}} \frac{x_n}{c_n} - \sum_{j \in \mathcal{L}, j \neq l} \frac{x_j}{c_j}) \tag{4.32}$$

Not that the term $\sum_{n \in \mathcal{N}} \frac{x_n}{c_n} + \sum_{j \in \mathcal{L}, j \neq l} \frac{x_j}{c_j}$ in Equation 4.32 represents the time fraction taken by all other flows except flow l in one TTI on the average sense. That is, $(\sum_{n \in \mathcal{N}} \frac{x_n}{c_n} + \sum_{j \in \mathcal{L}, j \neq l} \frac{x_j}{c_j}) * TTI$ represents the total time base station has to spend to transmit other flows in one TTI on average. Thus this value actually corresponds to the received packets interarrival time of user l on the average sense denoted as Δt_l. Therefore,

$$(\sum_{n \in \mathcal{N}} \frac{x_n}{c_n} + \sum_{j \in \mathcal{L}, j \neq l} \frac{x_j}{c_j}) = \frac{\Delta t_l}{TTI} \tag{4.33}$$

Combining Equation 4.32 and Equation 4.33 yields

$$\frac{x_l}{c_l} = \max(0, 1 - \frac{\Delta t_l}{TTI}) \tag{4.34}$$

This gives us insight that the optimal low priority user rate x_l is closely related to the packet inter-arrival time Δt_l and the channel capacity c_l

Quickly and accurately estimating capacity and the presence of other competing traffic is hard in wired and Wi-Fi networks due to shared FIFO queues. Our **key insight** is that the PF scheduler in

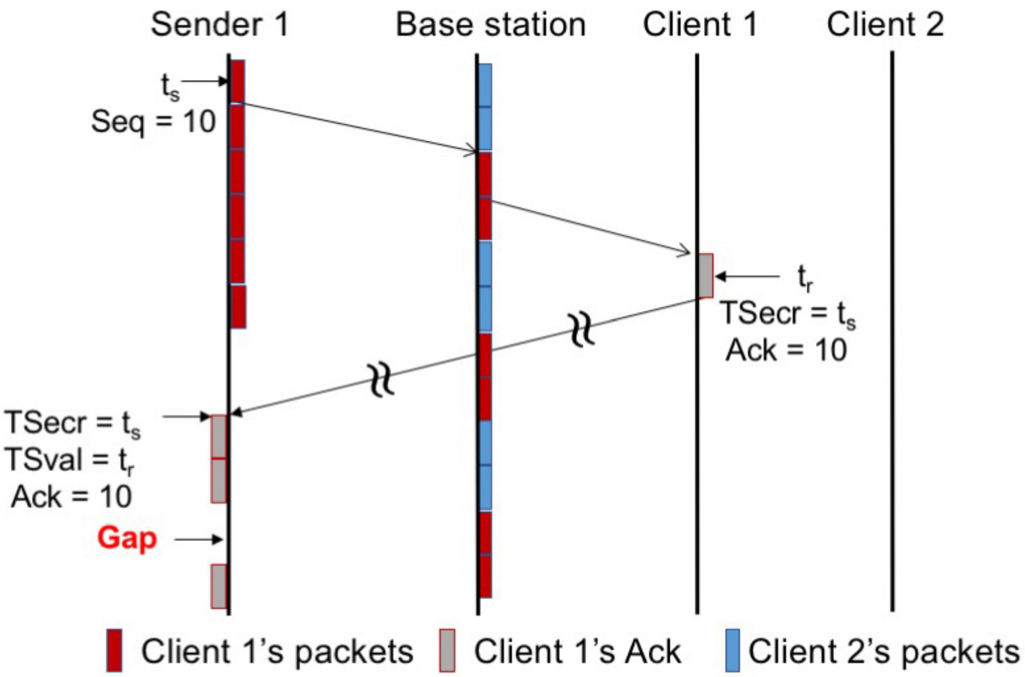


Figure 42. Scheduler and timestamps

cellular networks enables us to accurately detect capacity and competition in *only* a handful of TTIs. For *each* TTI, the scheduler selects a receiver based on its signal strength and long-term throughput; hence the scheduler is likely to service those receivers that it has not recently serviced. Also, the scheduler is likely to allocate *all* resource blocks to a single receiver in a given TTI, provided the receiver's queue has enough data (i.e., if a sender sends a sufficiently large burst of data, it is likely to receive full capacity for at least one TTI). Therefore, we estimate capacity by observing the maximum number of packets that were serviced in each TTI and we detect competing traffic by analyzing packet inter-arrivals at the receiver. We avoid making changes to client-side software by utilizing TCP timestamp option [125] so that the received timestamps of data packets can be observed via `ACKs` at the sender. TCP timestamp option is commonly enabled in clients [126, 127].

We use Figure 42 to explain how we can detect gaps in the schedule — this indicates competition or *busyness* — by observing timestamps. In this simple example, two senders share the base station (e.g., same LTE band) and send data to two cellular clients; sender 1 sends to client 1 and sender 2 to client 2; we show only sender 1 and client 1. Sender 1 sends a stream of packets, which are queued in the base station before they are scheduled to be sent to client 1. Assuming the two clients have similar signal qualities, the PF scheduler would pick one of the two senders in each TTI, provided there is enough data in queues. Therefore, client 1 is likely to receive a series of back-to-back packets until the scheduler switches to sender 2. Sender 1 observes the TCP options fields TS_{ecr} and TS_{val} of `ACKs`: TS_{ecr} indicates the timestamp at sender 1 when the packet was sent; TS_{val} indicates the timestamp at client 1 when the packet was received. If there is a "gap" in the schedule, then packets that were sent together (i.e., packets that have continuous or same TS_{ecr} values) will have a discontinuity in TS_{val} values.

4.2.4.2.2 Practical Issues

While simply observing received timestamps allows us to detect competition, delayed `ACKs` [128] and packet batching optimization at the receivers significantly complicate the detection logic. When the receiver combines and generates a cumulative `ACK` for multiple contiguous packets, the sender would infer an inflated capacity and spuriously detect gaps in the received stream. Figure 43 shows the effect of batching in the received packet stream. Without any batching or delayed `ACKs`, we clearly see gaps in the received packet stream in Figure 43(b) for the case of 2 senders, whereas there is no gap in

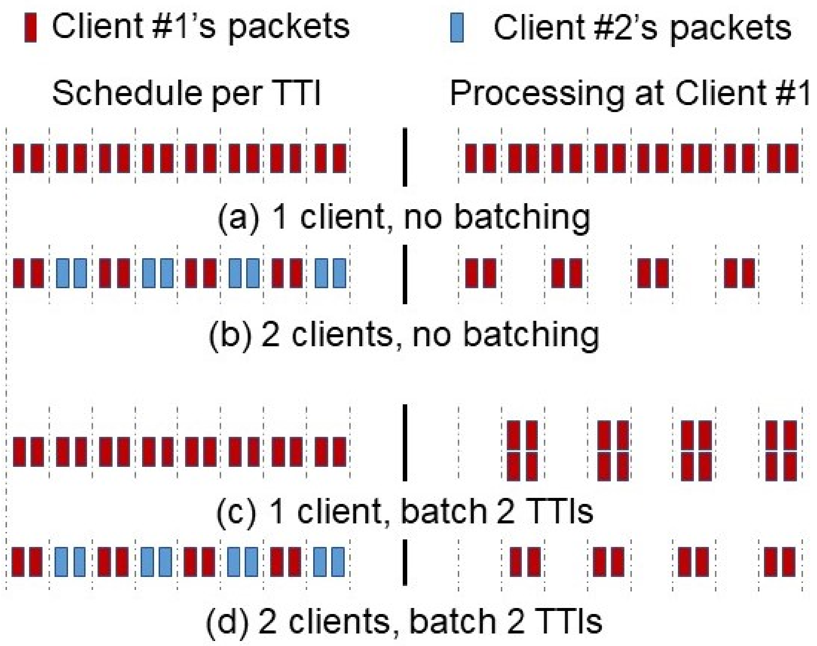

Client #1's packets
Client #2's packets
Schedule per TTI
Processing at Client #1
(a) 1 client, no batching
(b) 2 clients, no batching
(c) 1 client, batch 2 TTIs
(d) 2 clients, batch 2 TTIs

Figure 43. Scheduler with batching

Figure 43(a) when there is only one sender. In Figure 43(c), although there is only one sender (to client 1), the client 1 combines two TTIs worth of data in layer-2 before processing in higher layers, and, therefore, there is a gap in the observed timestamps. However, there is no contention and we should not throttle the sender. But, if we look for gaps in the received timestamps to identify competition, we would not distinguish between the cases of two senders without batching (Figure 43(b)), one sender with batching (Figure 43(c)) and two senders with batching (Figure 43(d)). It is hard to guess the batching behavior of receivers. Therefore, we further refine our detection logic in algorithm 1. Instead of *only* looking for gaps in the received stream, we reconstruct the schedule at the sender by observing the timestamps. The algorithm considers both the data volume and packet inter-arrival times to infer capacity and competition.

4.2.4.2.3 Capacity and Busyness Estimation Algorithm

Algorithm 1 compares the recently received sequence number and timestamp (i.e., when the corresponding data packet was received at the client) to the previous `ACK`. If the timestamps match, then we accumulate the number of acknowledged segments for the previous slot (line 22). If the timestamps do not match, *first* we see if there is a gap (i.e., the current timestamp differs from the previous timestamp by *more* than 1 TTI), as in line 8. If there is no gap, we infer that the `ACKs` are back-to-back and that the base station is *not* busy. If the `ACKs` are not back-to-back, then we compute the amount of data sent per TTI (i.e., the instantaneous rate in terms of packets per TTI) for the previous slot (line 7). Since the scheduler is expected to allocate full capacity for at least one TTI, we infer capacity by looking at the amount of data sent per TTI and by taking the maximum value (line 15). If there is no gap, we update capacity unconditionally so that the capacity can increase or decrease based on channel conditions (line 19). We detect busyness using a simple heuristic: if the instantaneous rate is more than the capacity by a factor λ, then we infer that the base station is *not* busy. Intuitively, if there is no other traffic, we expect instantaneous rate to be very close to capacity. If there are other senders or if there are not enough packets at the base station to consume all the resource blocks for one TTI, the instantaneous rate would be less than capacity. Therefore, λ lies between 0 and 1. Setting $\lambda = 0$ would *always* allow *Legilimens* senders to send in the normal mode (i.e., *Legilimens* would never yield). While $\lambda = 1$ would allow *Legilimens* senders to send traffic in the normal mode *only* when there are no other senders, it would

Algorithm 1 Estimate capacity and busyness

Input: ACK sequence number (s), timestamp (t)

Output: Capacity ($capacity$), IsBusy ($busy$)

```
function ESTIMATE($s, t$)
    if $t$ ¿ $t_0$ then
        $i \leftarrow i + 1$
        $slot[i].t \leftarrow t$
        $slot[i].n \leftarrow (s - s_0)$
        $duration \leftarrow (t - slot[i-1].t)$ / TTI
        $rate \leftarrow slot[i-1].n \ / \ duration$
        if ($duration > 1$) then                              ▷ If there is a gap
            if ($rate \geq \lambda * capacity$) then
                $busy \leftarrow$FALSE
            else
                $busy \leftarrow$TRUE
            end if
            if ($rate > capacity$) then
                $capacity \leftarrow \frac{4}{5} * capacity + \frac{1}{5} * rate$
            end if
        else
            $busy \leftarrow$FALSE
            $capacity \leftarrow \frac{4}{5} * capacity + \frac{1}{5} * rate$
        end if
    else
        $slot[i].n \leftarrow slot[i].n + (s - s_0)$
    end if
    $t_0 \leftarrow t, s_0 \leftarrow s$
```

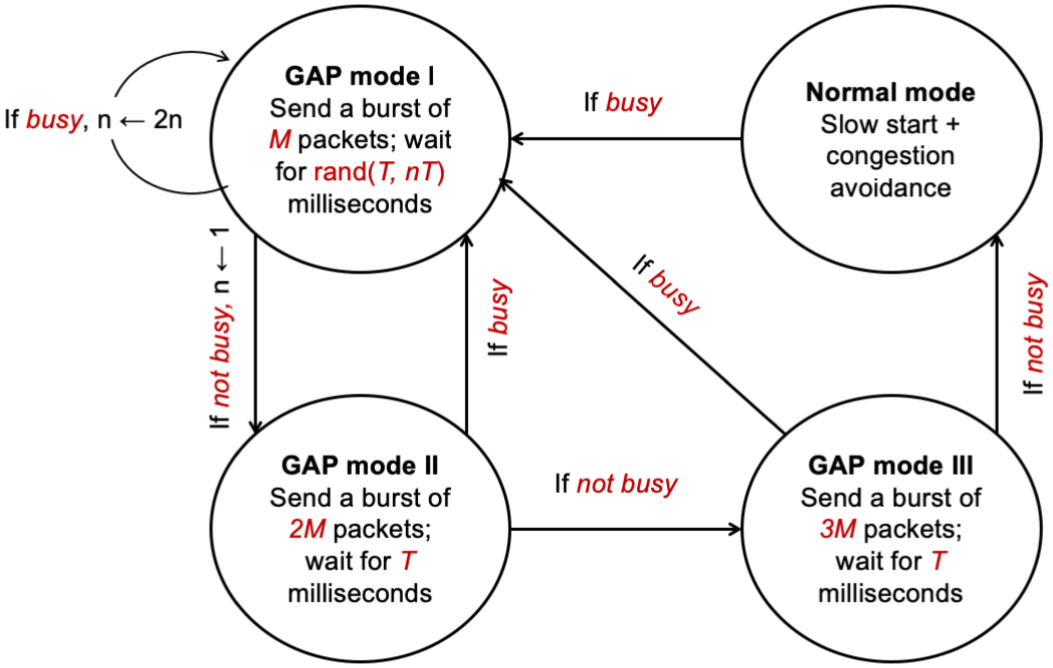

GAP mode I
Send a burst of M packets; wait for rand(T, nT) milliseconds
Normal mode
Slow start + congestion avoidance
GAP mode II
Send a burst of 2M packets; wait for T milliseconds
GAP mode III
Send a burst of 3M packets; wait for T milliseconds
If busy, n ← 2n
If busy
If not busy, n ← 1
If busy
If busy
If not busy
If not busy

Figure 44. *Legilimens* congestion control

be sub-optimal if there are other senders but they do not have enough data to saturate the capacity; we would want *Legilimens* to use spare capacity in this case. Therefore, we chose $\lambda = 0.5$ (Table Table I) in our experiments, which achieves a good trade-off between foreground and background performance. Finally, we use *exponential* averaging with a weight of $\frac{1}{5}$ to the newly inferred capacity (line 15 and line 19).

4.2.4.3 Congestion Control

4.2.4.3.1 Congestion Window ($cwnd$) Adaptation

We show *Legilimens*'s complete state machine in Figure 44. *Legilimens* relies on estimated `capacity` and `busy` signals to switch between sending (i.e., *normal mode*) and probing (i.e., GAP mode). Because normal mode aggressively sends data, a series of three probe modes is used (i.e., GAP modes I, II, and III), each with a larger burst than the previous, to increase the confidence in our capacity estimates before entering *normal mode*. During each GAP mode, we send a burst of multiples of `M` packets and wait for `T` milliseconds to estimate `capacity` and `busyness` by observing `ACKs` using algorithm 1. If the base station is not `busy`, we enter the next GAP mode, in which we increase the burst size and wait for `ACKs`. After the third GAP mode (i.e., GAP III), we enter normal mode if no busyness is detected. If, at any point, we sense the presence of other senders , we revert back to the GAP mode I. If busyness is sensed in GAP mode I, then exponential random back-off is applied in multiples of `T`.

During *normal mode*, *Legilimens* employs AIMD-style congestion control. While *Legilimens* can be implemented on top of any $cwnd$ adaptation algorithm, we implemented *Legilimens* on top of TCP RENO.

Figure 45 shows the evolution of $cwnd$ over time as *Legilimens* transitions through operation modes during one of our experiments. While we explain our methodology later, here we point to GAP modes, a key aspect of our design, which allows scheduling opportunities to other traffic. During GAP mode, we probe for spare capacity using bursts of packets and wait. Based on estimates of link capacity and busyness, we switch to normal mode with a RENO-like slow start and continue in congestion avoidance (also shown in Figure 45).

4.2.4.3.2 Fairness

Legilimens's primary objective is to satisfy demands of foreground applications and provide fairness between competing background flows. To this end, if any *Legilimens* sender estimates that there are

TABLE I

PARAMETER VALUES

Parameters	**Description**	**Value**
λ	Busy threshold	0.5
M	Probe burst size	50 KB
T	Time between GAP modes	250 ms

other competing senders, background or foreground, the sender exponentially backs off (i.e., doubles the waiting interval, `T`). This back-off can happen anytime during the probing burst as the sender estimates `busyness` on each `ACK`. Our mechanism is somewhat similar to CSMA in IEEE 802.11, and, therefore, we achieve fairness between background traffic in a similar way. We study fairness in our evaluation in the following Section. Table Table I lists our design parameters along with their default values; we analyze their sensitivity later.

4.2.4.3.3 Summary and Discussion

In cellular networks, packet delay can increase either due to (1) degradation in signal quality or (2) contention at the base station. Existing LPT protocols do not correctly identify the root cause for delay increase but respond by either completely backing off or not at all in both cases (see Figure 40). *Legilimens*'s capacity and busyness estimation algorithm, leveraging the nature of PF schedulers, deconstructs the schedule by observing timestamps and isolates the two cases. For (1), *Legilimens* correctly reduces the rate but does not completely back off. For (2), *Legilimens* backs off, enters GAP mode and waits for the base station to become free. Thus, differentiating between the two cases is key to *Legilimens*'s performance gains over existing LPT protocols.

4.2.4.4 Deployment Model

Legilimens, in its current form, only accounts for how the PF scheduler handles downlink traffic. Given this explicit optimization for cellular downlink traffic, it only makes sense to deploy *Legilimens* on servers providing data to cellular devices. While we do not foresee significant technical challenges in deploying *Legilimens* on such servers (since it is just another variant of TCP), we recognize that adoption may be slow for non-technical reasons. That said, a potential path for adoption could be deploying *Legilimens* on cellular proxy servers and TCP splitters that most cellular operators employ today [129] or on CDN edge servers that specifically serve traffic to mobile devices over cellular links. Further, network and CDN operator would utilize existing traffic classification techniques to identify background traffic and direct its delivery via servers that run *Legilimens*. This would significantly reduce the number of servers that need to support *Legilimens*, while still allowing cellular networks to enjoy its benefits.

4.2.5 Evaluation Using a Real Implementation

We evaluate *Legilimens* using three avenues: (1) real implementation on a real network, (2) real implementation on *PhantomNet* [130], and (3) simulations using ns-3. We use real implementation to analyze the performance of background flows with respect to foreground traffic, to show a proof-of-concept, and to evaluate the performance when clients are mobile.

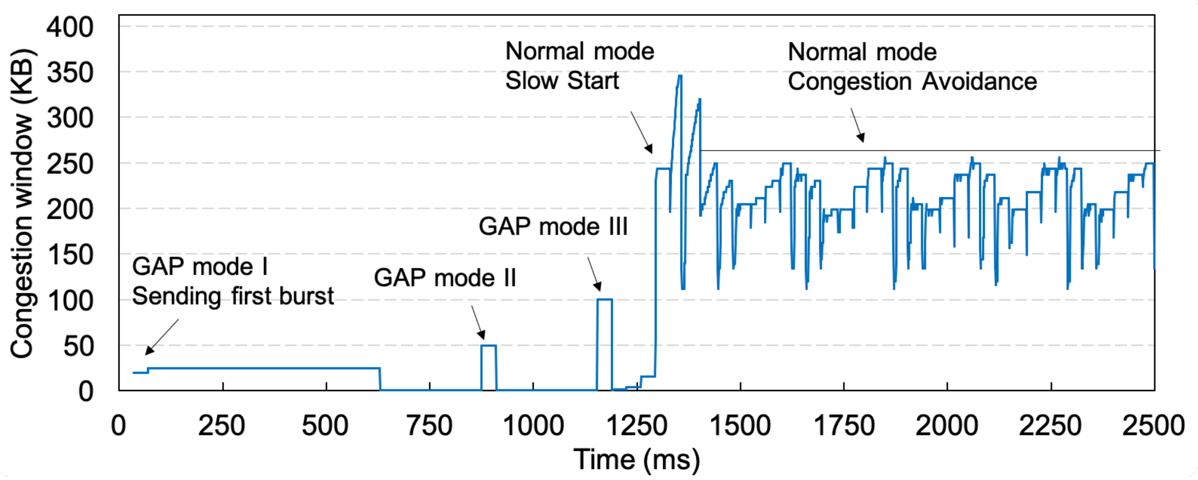


Figure 45. *Legilimens* congestion window evolution

4.2.5.1 Methodology

4.2.5.1.1 Real-network Test-bed

Although we cannot control user traffic in real cellular network, the test-bed offers a realistic setting to analyze *Legilimens*. Our test-bed consists of 4 Samsung Galaxy smartphones – 3 J7 models for foreground and 1 S6 model for background traffic, all with Android 7.0. The devices are *band-locked* to use the same LTE carrier (*i.e.,* they share the same bottleneck radio link). For the stationary experiments, test devices are located on the second story of a 3-story concrete building with wall-to-wall windows. The test devices are connected to the standard macro cell using a 10 MHz carrier. The devices register Reference Signal Received Power (RSRP) between -92 and -95 dBm and Reference Signal Received Quality (RSRQ) between -10 to -13 dB.

4.2.5.1.2 PhantomNet Test-bed

Unlike the real network, which is not isolated from outside traffic, *PhantomNet* enables us to analyze *Legilimens* in a more isolated setting. Thus, we rely on *PhantomNet* to validate our real runs and to show proof-of-concept with real implementations on clients and servers, fully controlled load, but emulated cellular network and limited control of the radio signal. Our PhantomNet test-bed emulates a cellular network using a server running open air interface (OAI) inside the LTE packet core and eNodeBs (base stations). eNodeB is based on Software-Defined Radio (SDR) running OAI (Intel NUC + USRP B210). PhantomNet provides Nexus 5 phones accessible via Android Debug Bridge (ADB), target server and a GUI interface. Log-distance path loss model is used to produce realistic varying Signal to Noise Ratio (SNR). We use 4 Nexus-5 phones, similar to our real-network scenario, with remotely connected servers generating test traffic.

4.2.5.1.3 Workload and Relevant Metrics

We evaluate *Legilimens* using two types of workloads: (1) *one-on-one* workload to show proof-of-concept that *Legilimens* utilizes spare capacity and backs off when there is foreground traffic; (2) *mixed workload* to evaluate performance of protocols used for background flows and their effect on foreground traffic.

One-on-one workload generates 1 background and 1 foreground flow. The background flow is an always-on, long flow that uses *Legilimens*. The foreground flow uses CUBIC (our baseline) and sends data in an on-off pattern. We study the per-second throughput of foreground and background flows over time to see whether *Legilimens* (background) yields to foreground flows.

Mixed workload consists of a varying number of foreground flows and background flows. While the foreground flows use CUBIC, background flows use different schemes, including *Legilimens*. We model our *mixed workload* as heavy tailed with a small number of large flows generating the vast majority of bytes [131]. Specifically, we model foreground traffic as a mix of short, medium, and long flows: (1) 64 KB *short* flows represent web objects and mobile app communication, (2) 1 MB *medium* flows represent transfers such as video *chunk size* in adaptive streaming, and (3) 32 MB *long* flows are representative of app updates and data backups. The short, medium and long contribute to 10%, 30%, and 60%, respectively, to overall (foreground) load. The background flows remain *always on*.

We use three traffic load levels for mixed workload, intended to represent different resource load levels of an LTE cell, which is typically measured by PRB utilization, U_{PRB}. The load levels are: (1) low load, representing average $U_{PRB} = 30\%$, (2) moderate load, representing average $U_{PRB} = 60\%$, and (3) high load, with average $U_{PRB} = 80\%$. We achieve the desired U_{PRB} by generating foreground traffic data rates relative to the maximum achievable throughput, and measuring the resulting U_{PRB} at each level.

We evaluate the protocols on our *mixed* workload based on the following metrics:

- Median and tail (e.g., 99^{th} percentile) flow completion times (FCT) of short flows
- Throughput of medium (T_{med}) and long flows (T_{long})
- Throughput of background flows (T_{bg})
- Overall *link* utilization (in simulations).

4.2.5.1.4 Protocol Implementations

We deployed *Legilimens* and competing protocols, CUBIC, VEGAS, TCP-LP, and LEDBAT, on a Linux 4.4 server, with TCP timestamps enabled [125].

4.2.5.2 Results from the Real Network

We use real network to study performance and robustness of our design across various settings (idle, busy, stationary, mobile, etc). Because real network conditions are dynamic (especially during busy hours), we run our experiments multiple times to get sufficient confidence. Each experiment typically lasts for 4–8 hours and transfers about 100 GB of data.

We compare *Legilimens* against CUBIC, TCP-LP, LEDBAT and VEGAS. We implemented two variants of *Legilimens*. While *L-passive* looks for a gap of 2 or more TTI to identify busyness (see algorithm 1, line 8), *L-active* looks for a gap of 3 or more TTI. Thus, *L-passive* is a conservative (passive) variant, whereas *L-active* is an aggressive (active) variant.

4.2.5.2.1 Mixed Workload During Idle Hours

We measure the *maximum achievable* throughput during idle periods using large downloads and generate 30%, 60%, and 80% of the measured throughput as foreground traffic. We run our mixed workload for 15 minutes. Using standard reports from the cellular infrastructure, we show the actual U_{PRB} level for our base station as 15-minute averages in Figure 46. The figure indicates that typical quiet hours ($U_{PRB} < 10\%$) are between midnight and 6 AM; as a result we run our experiments during these *quiet* hours to minimize the impact of other regular traffic. With our workload added, the resulting U_{PRB} is plotted for corresponding times, indicating that we are able to translate the traffic load into our desired

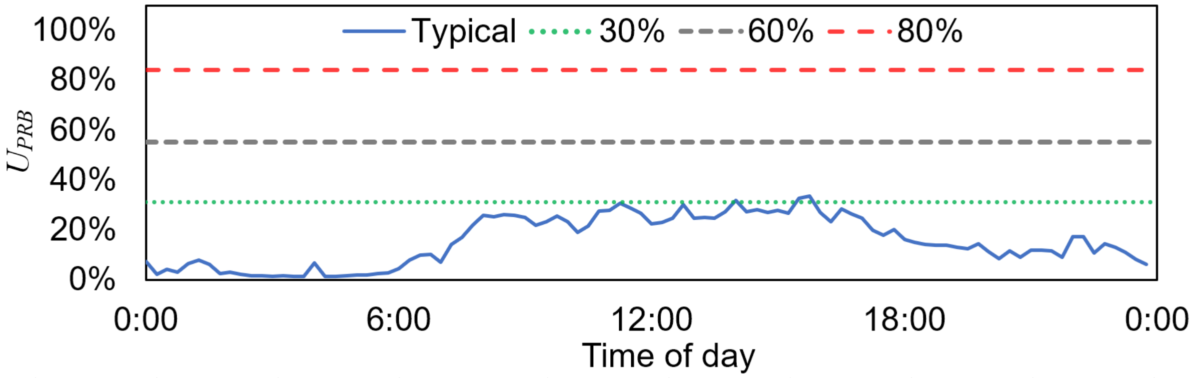


Figure 46. Typical U_{PRB} and workload impact

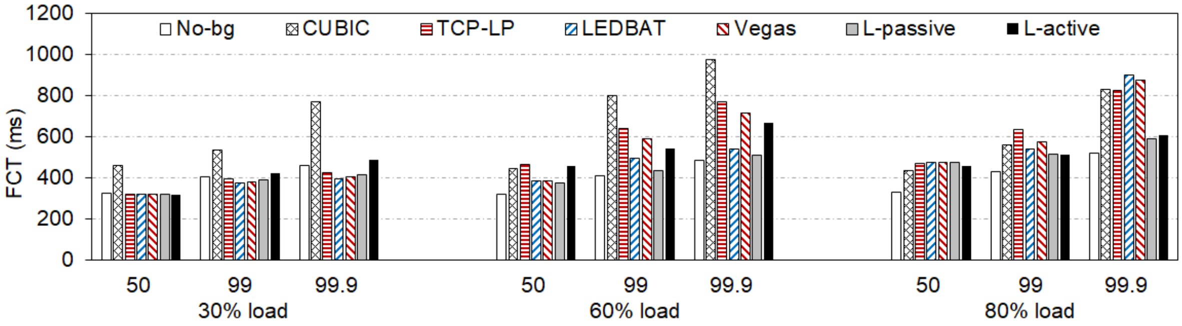


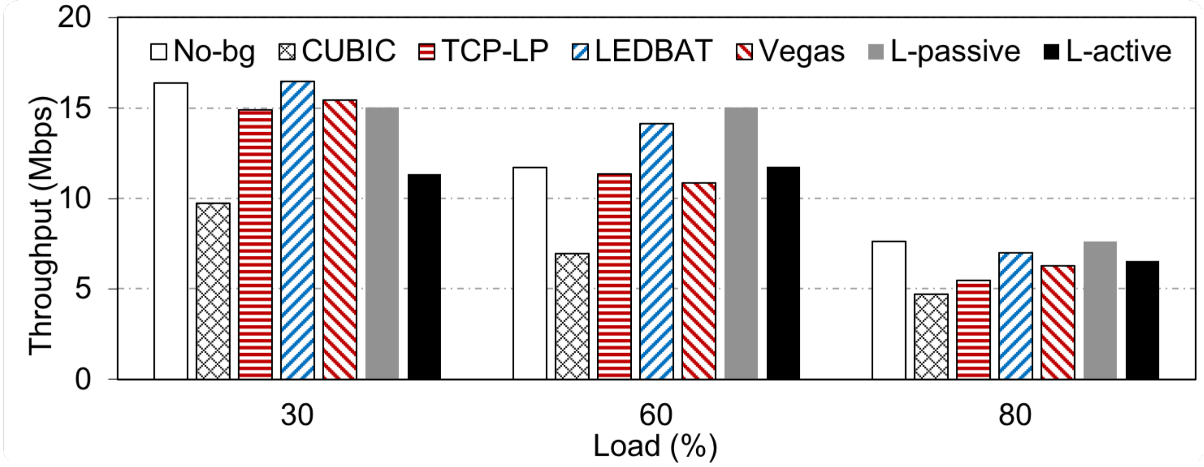
Figure 47. FCT for short flows in the real network

radio channel utilization with reasonable accuracy. In particular, the measured U_{PRB} levels are 30.1% for low, 55.3% for moderate, and 84% for high load.

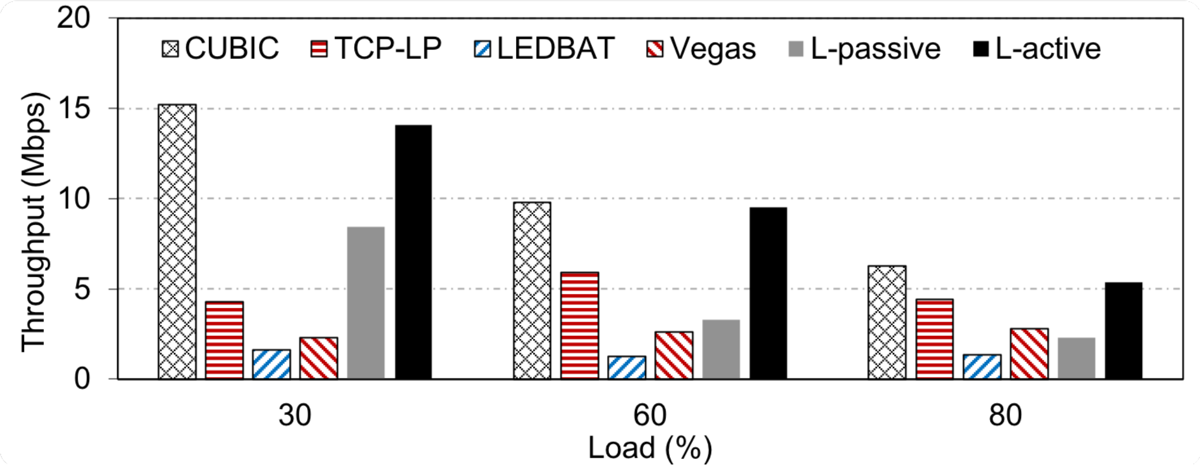
We examine the impact of different protocols (CUBIC, TCP-LP, LEDBAT, VEGAS, *L-passive*, and *L-active*) on short foreground flows using their median and tail (i.e., 99^{th} and 99.9^{th} percentiles) FCT in Figure 47. Because the FCT of foreground traffic would be unaffected by the traffic generated by an ideal background protocol, we also show “No-bg” case, which does not have any background flows, as a proxy for ideal. *L-passive* ensures the least impact on foreground short flows across loads, with no impact at low load and the lowest impact at other loads, even compared to other low priority protocols. *At the most common (moderate) load levels in cellular networks, L-passive has a distinct advantage to* CUBIC, *with median* FCT *16% lower (376 vs. 446 ms), and the tail* ($99.9th$ *%-ile) 48% lower. At high network loads, while comparable at the median, L-passive has 31% lower tail* FCT *than* CUBIC, *with similar advantage over other LPT protocols.* If higher link utilization is desired, *L-active* has minimal impact at low loads, is comparable to TCP-LP and VEGAS at moderate loads, and has superior performance at high loads, where it performs close to *L-passive*.

Next, we analyze the throughput of medium (T_{med}) and long (T_{long}) foreground flows, and the throughput of background flows (T_{bg}). An ideal protocol would utilize spare capacity to send background traffic without affecting foreground flows. For medium flows, all background protocols perform within 10% of the "No-bg" case and there was no clear winner (figure omitted). We examine T_{long} performance in Figure 48(a). As expected, the throughput of all the protocols decreases with increasing load. CUBIC, as a representative of aggressive loss-based protocols, clearly affects foreground throughput and degrades by up to 40% (e.g., the throughput drops from 16.4 to 9.7 $Mbps$ for low load). *L-passive achieves the highest average throughput across loads, with 38% to 54% higher throughput over* CUBIC. At 60% load, we see that *L-passive* is better than"No-bg" for T_{long}. This is an experimental artifact as we cannot fully control the load in a real network. So, in this case, it is likely that the load decreased slightly when we ran *L-passive*.

Finally, we show how effectively the background flows capture spare capacity in Figure 48(b). While LEDBAT and VEGAS conservatively yield to long foreground flows and allow them to achieve higher throughput, they do not capture spare capacity and also slow down short flows. TCP-LP performs well



(a) T_{long}



(b) T_{bg}

Figure 48. Throughput in the real network

in moderate and high loads but suffers under low loads, when the benefit would be the highest. We see a relation to TCP-LP's impact on tail FCT of short flows in Figure 47. At low load, it has little impact on FCT as it is quite conservative. However, it has detrimental impact at higher loads.

Comparing Figure 48(a) and Figure 48(b), we see that *other protocols lie in the two extremes*: delay-based variants (i.e., TCP-LP, LEDBAT and VEGAS) are conservative, consistent with our simple experiment in the previous section. They achieve high throughput for foreground flows but provide low throughput for background flows (especially at low loads when there is opportunity). CUBIC provides high throughput for background flows but this advantage comes at the cost of reduced throughput for foreground flows. In contrast, both *L-passive* and *L-active* achieve high throughput for both foreground and background flows. Finally, *L-passive* achieves high efficiency at low load, while it behaves close to VEGAS under moderate and high loads. We can also see that *Legilimens* can be effectively tuned to perform more aggressively if desired. *L-active* can capture nearly as much capacity as CUBIC across all loads. The cost of using *L-active* can be summarized as *18% (24%) penalty on median (tail)* FCT *only under moderate load and 14%–25% penalty on* T_{long} *across load levels, for the 35%–60% benefit of more efficient capacity utilization across load levels.*

4.2.5.2.2 Mixed Workload During Busy Hours

To validate the behavior of *Legilimens* during busy hours, we conduct test runs over 3 days between 13:00 and 17:00 hours, when regular network (Base) load is present, over which we have no control. We supply foreground traffic in lower volume, at an overall rate of about 5 $Mbps$, consisting of a mixed workload. We record the total downlink data volume and U_{PRB} measurements from the network. In each hour, we have four 15-minute timebins, each containing one of these loads: (1) Base load only, (2) foreground traffic is added, (3) background flow using CUBIC is additionally added, and (4) *L-passive* is used for a background flow. The first two cases are merely to confirm the volume of Base load and foreground traffic, while we study the cases with background traffic in Figure 49. We select 15-minute bins with similar Base load to compare the total data volume of CUBIC (C) and *Legilimens* (L) to

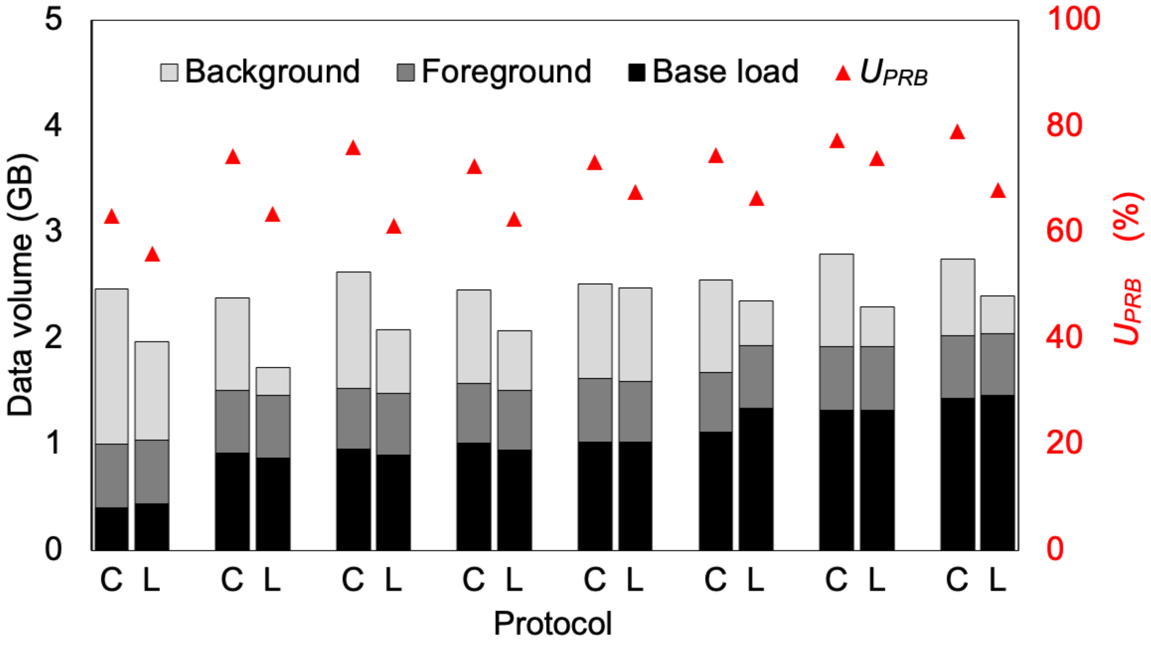
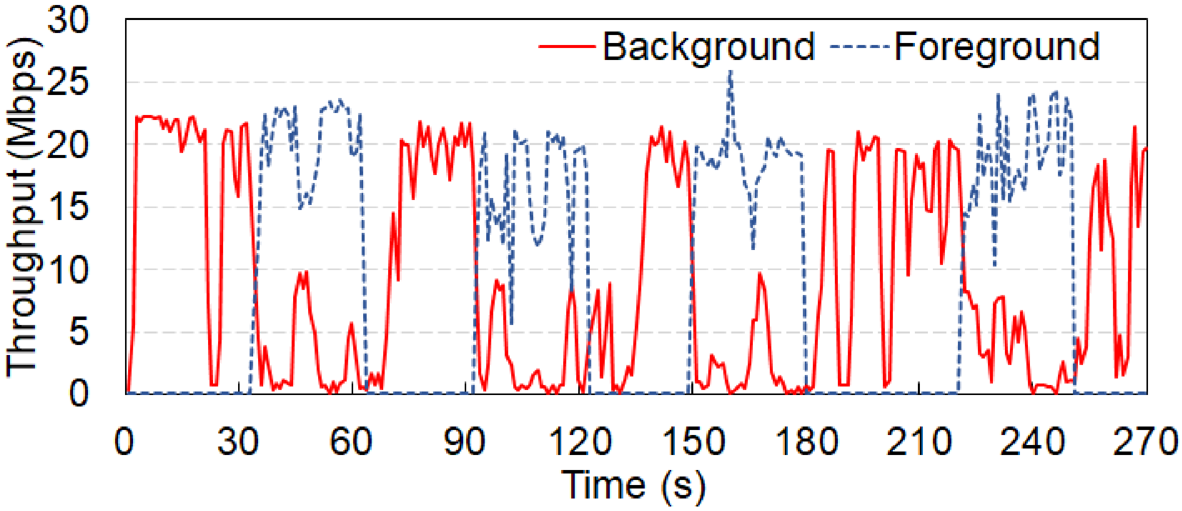
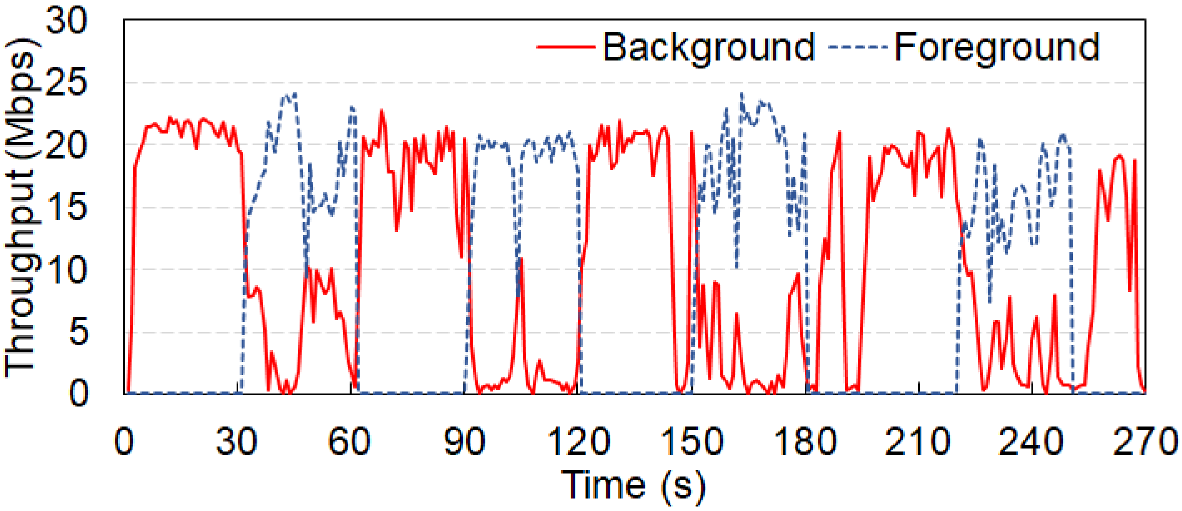


Figure 49. Cell load and utilization during busy hours

their respective U_{PRB}. We discard the timebins with Base load outliers because the load fluctuation was significant in those bins. Figure 49 shows the pairs of CUBIC/*Legilimens* timebins and their data volumes as stacked bars, sorted by increasing Base load (bottom bar sections). We can see that *Legilimens* generates less traffic (top bar sections) and results in lower U_{PRB} than CUBIC, across the range of Base loads. On average, *Legilimens* results in 13.8% lower U_{PRB} than CUBIC, which is desirable during busy hours. We observe that CUBIC generates the same amount of background load regardless of the Base load, while *Legilimens* modulates its background throughput favorably depending on base load.



(a) *L-passive* vs. CUBIC



(b) *L-active* vs. CUBIC

Figure 50. One-on-one workload in real network

4.2.5.2.3 One-on-one Workload While Stationary

Our one-on-one workload allows us to carefully analyze and contrast *Legilimens*'s behavior to those of LEDBAT and TCP-LP. (Figure 40). Figure 50 shows how *Legilimens* shares the cellular link with

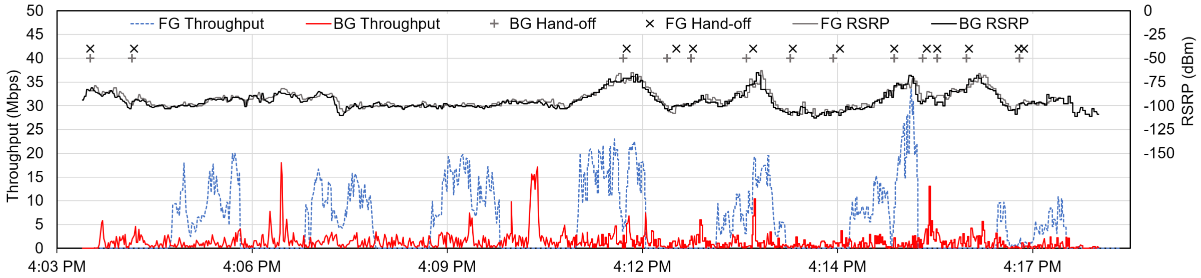


Figure 51. Throughput of *L-passive* (background) vs. CUBIC (foreground) while moving

CUBIC. Unlike LEDBAT and TCP-LP (Figure 40), *L-passive* yields to CUBIC, backs off completely in GAP mode (to allow scheduling opportunity to other flows), and efficiently recaptures spare capacity when available. This results in the pattern seen in Figure 50(a), in which *L-passive* measured 8.9 *Mbps* vs. CUBIC's 19.5 *Mbps*.

L-active shows a more aggressive capture of spare capacity in Figure 50(b), both with and without persistent foreground CUBIC flow. The measured throughputs are 10.6 *Mbps* for *L-active* vs. 18.6 *Mbps* for CUBIC.

4.2.5.2.4 One-on-one Workload While Moving

To evaluate *Legilimens* in a mobile scenario, where changing radio signal and different cell loads can be frequently encountered, we conduct a mobile test. The test lasts for about 37 minutes, starting with walking for about 8 minutes, then driving at moderate speed (30-70 km/h) for about 3 minutes, followed by freeway speed (70-110 km/h) for 10 minutes, and completing the remainder of the test at moderate driving speed. The total distance is about 24 km on freeway and residential area roads. For clarity, we show the first 14 minutes of the test. Two J7 phones are used, placed in a laptop bag, which is carried and

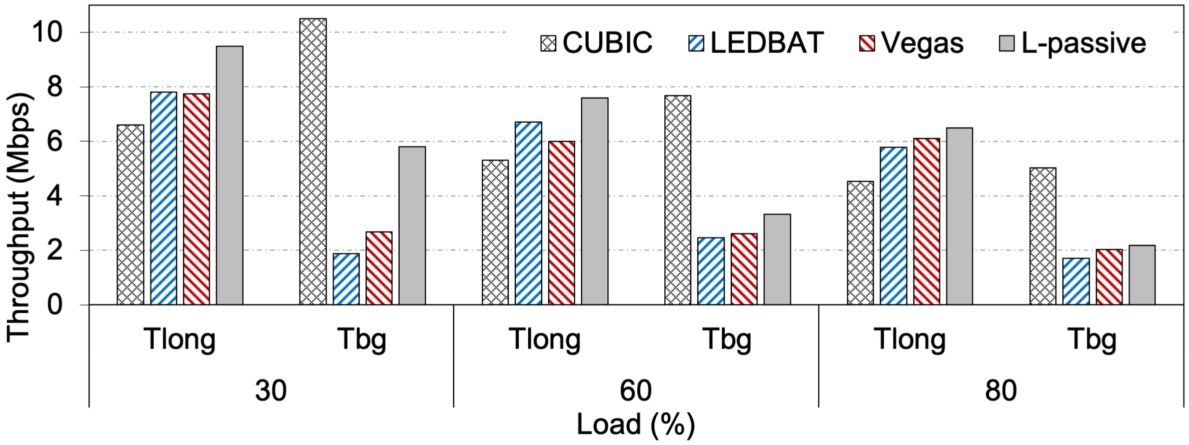


Figure 52. Throughput in PhantomNet

then placed on the vehicle seat. One phone runs a continuous *L-passive* background flow, and another phone runs a foreground CUBIC flow with 1-minute ON/OFF pattern.

Figure 51 shows the results and the mobility profile of this test. Signal strengths, measured as RSRP, varied from -111 dBm to -68 dBm. A total of 13 hand-offs were performed. While the two devices sometimes perform a hand-off a few seconds apart, their signal strength is almost the same , with small but expected deviation.

Performance-wise, we do not expect to see a clear pattern of yielding by *L-passive*, simply because we only see two flows among a large volume of regular traffic in the network. However, as expected, we see that *L-passive* performs significantly more conservative than CUBIC, which competes with other traffic. *L-passive* also does not appear to suffer major disruption by changing signal strength, especially in the second half of the time series, where driving causes frequent hand-offs.

4.2.5.3 Results from PhantomNet

We discuss relevant PhantomNet results as a proof-of-concept to further validate and clarify performance at different workloads, using representative protocols. Due to the limitation on number of devices supported at high network loads in PhantomNet, we are limited to 70% for high load.

Mixed workload: We only run *L-passive*, which provides better foreground performance than *L-active*. The short flow FCT is consistent with real network results, both in the trend over varying loads and relative performance (not shown). Figure 52 shows a combined plot of T_{long} and T_{bg} for mixed workload in PhantomNet test-bed. Our throughput performance is also generally consistent with real network. T_{long} is most positively impacted by *L-passive*, similar to real network, while T_{bg} of *L-passive* is by far superior to LEDBAT and VEGAS at low load, and still better at moderate load.

One-on-one workload: One-on-one performance results in Phantomnet are similar to those in real network, and, therefore, we omit them for brevity.

4.2.6 Evaluation Using Simulations

We rely on simulations to analyze *Legilimens* at scale (i.e., with 100 devices), to study its fairness, sensitivity to parameter settings, and its queuing behavior. We also compare *Legilimens* to a larger set of protocols, including BBR [113], for background flows. We use the open source ns-3 implementation of BBR [132, 133].

4.2.6.1 Methodology

We use `ns-3` v3.27 simulator with log-distance propagation model for the radio signal. The radio channel is configured with a single carrier with 10 MHz bandwidth and closed-loop MIMO. Network topology consists of multi-hop wired links that connect remote servers to Packet Data Network Gateway (PGW) using 1 Gbps with 10 ms delay links. The link speed between the base station and the SGW/PGW is also 1 Gbps. The cellular uplink and downlink are bottlenecks in our topology.

Our `ns-3` workloads are also based on one-on-one and mixed workloads. For mixed workloads we generate traffic load in the same way as mentioned above, and measure average U_{PRB} of 28.8%, 63.6%, and 88.1%, for low, moderate, and high loads, respectively. We ran our experiments with and without *carrier aggregation*, and we achieved similar results.

4.2.6.2 Results from Simulation

4.2.6.2.1 Scale

To generate mixed workload, we simulate a substantially larger client base with 100 devices, to exercise the impact of the PF scheduler behavior under variety of RF signal characteristics, and the resulting performance of *Legilimens* and other protocols. The key difference between the real test-bed and simulations is the scale (*i.e.,* the scheduler in our simulated network must serve flows from 100 queues as opposed to 4 in our real test-bed). The 100 devices are randomly placed in a cell and they generate a mix of foreground and background flows. We run each 15-minute test 3 times with different random seeds for client selection.

For foreground short flows, we observe that *L-passive* outperforms CUBIC and BBR, and *L-passive* performs similar to TCP-LP, LEDBAT, and VEGAS (plot not shown). We show a combined plot of T_{long} and T_{bg} in Figure 53. In this plot, we also show a system with no background flows ("No BG"), which serves as ideal. We see a clear pattern: at 30% load, all protocols except CUBIC achieve good foreground throughput but TCP-LP, LEDBAT, and VEGAS suffer in background throughput; at higher

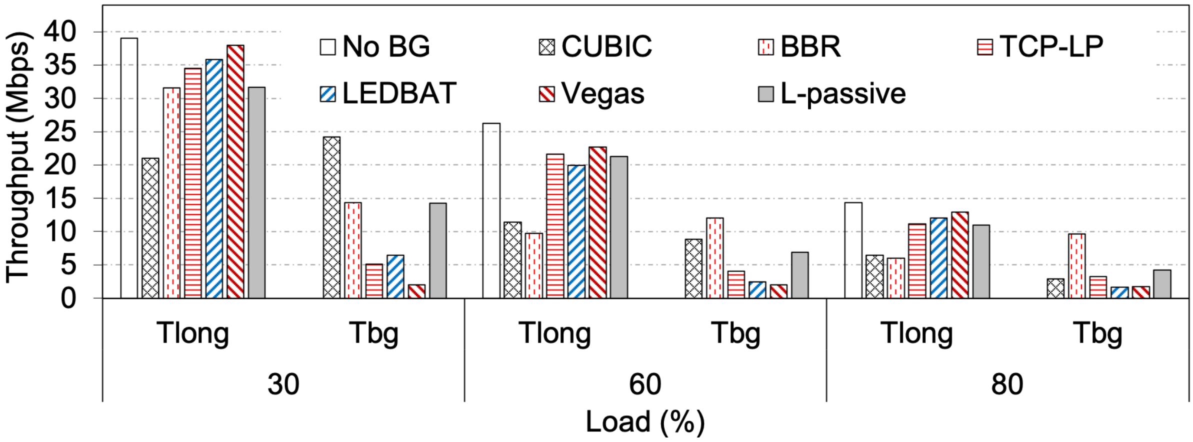


Figure 53. Throughput in a simulated network

60% and 80% loads, CUBIC and BBR suffer in foreground throughput, whereas TCP-LP, LEDBAT, and VEGAS suffer in background throughput. Across all loads, *L-passive* is able to achieve close to ideal foreground throughput (i.e., closer to “No BG”), while maximizing background throughput.

We also analyzed the overall link utilization of CUBIC, VEGAS, and *L-passive*. At low and moderate loads, when capturing spare capacity is highly desired, other protocols lag behind CUBIC allowing only 63%–85% of throughput achieved with CUBIC, while *L-passive* is able to fill 95%–100% of link capacity that CUBIC does. At high loads, all the protocols show similar overall utilization. However, *L-passive* provides a higher fraction of capacity to foreground flows.

4.2.6.2.2 Fairness

Mutual fairness is examined by starting 4 *Legilimens* flows in a staggered manner every 20 seconds (Figure 54). Since these are typically long flows, we are interested primarily in their long-term fairness, while short term oscillations are expected and acceptable. Since any single *Legilimens* flow considers

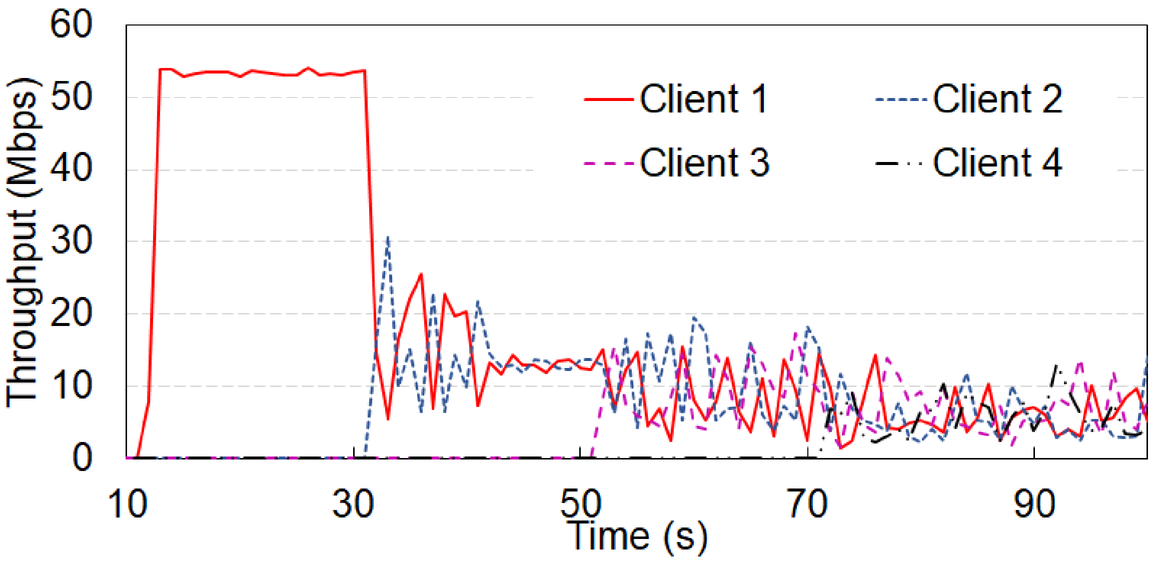


Figure 54. Fairness analysis among *Legilimens* flows

all other traffic as foreground, even if other background flows are present, it has to transition between operation modes. Figure 54 shows that *Legilimens* flows adapt their sending rate with increasing competition and converge to fair shares. The Jain's Fairness Index for the periods of 2, 3, and 4 concurrent flows are 0.999, 0.993, and 0.996, respectively, indicating high fairness. We have also tested fairness of 8 and 24 concurrent *Legilimens* flows, and they resulted in JFI of 0.992 and 0.999, respectively.

4.2.6.2.3 Sensitivity

We varied both probe interval and probe size to better understand how the performance of foreground and background flows varies with these parameters. Table Table I lists the default values for probe interval (`T`) and probe size (`M`).

We vary the interval from 100 ms to 350 ms in 50 ms steps. We found T_{bg} to be more sensitive than T_{long} and T_{bg} decreases slightly with increased probe interval. Overall, we found stable performance between 100 ms and 350 ms. We chose 250 ms, which gave good performance to foreground flows without sacrificing on background flows. Similarly, we vary probe size. We found that even small probes result in accurate estimates of capacity and busyness, and that we can tune the sending rate of probes across the wide range. We select the 50 KB probe size, which gives the best performance for medium and long flows. We omit sensitivity plots due to space constraints.

4.2.6.2.4 Queuing Behavior

Finally, we analyze the queue lengths of CUBIC, LEDBAT, and *Legilimens* at the base station to understand the effectiveness of *Legilimens* in yielding to other traffic. Figure 55 shows the queue lengths of CUBIC, LEDBAT, VEGAS, and *Legilimens*, along *y* axis versus time along *x* axis. Each background flow starts alone, having a foreground CUBIC flow joining at 5 second mark, indicated by vertical lines.

While CUBIC, LEDBAT, and VEGAS exhibit non-trivial queue lengths and compete with foreground flows for scheduling opportunity, *Legilimens* keeps nearly empty queues for long periods, which provides opportunity for other traffic. As such, *Legilimens* has a significantly lower average queue length than other protocols. *Legilimens*'s algorithm detects *busyness* quickly and backs off, which is key to its performance gains.

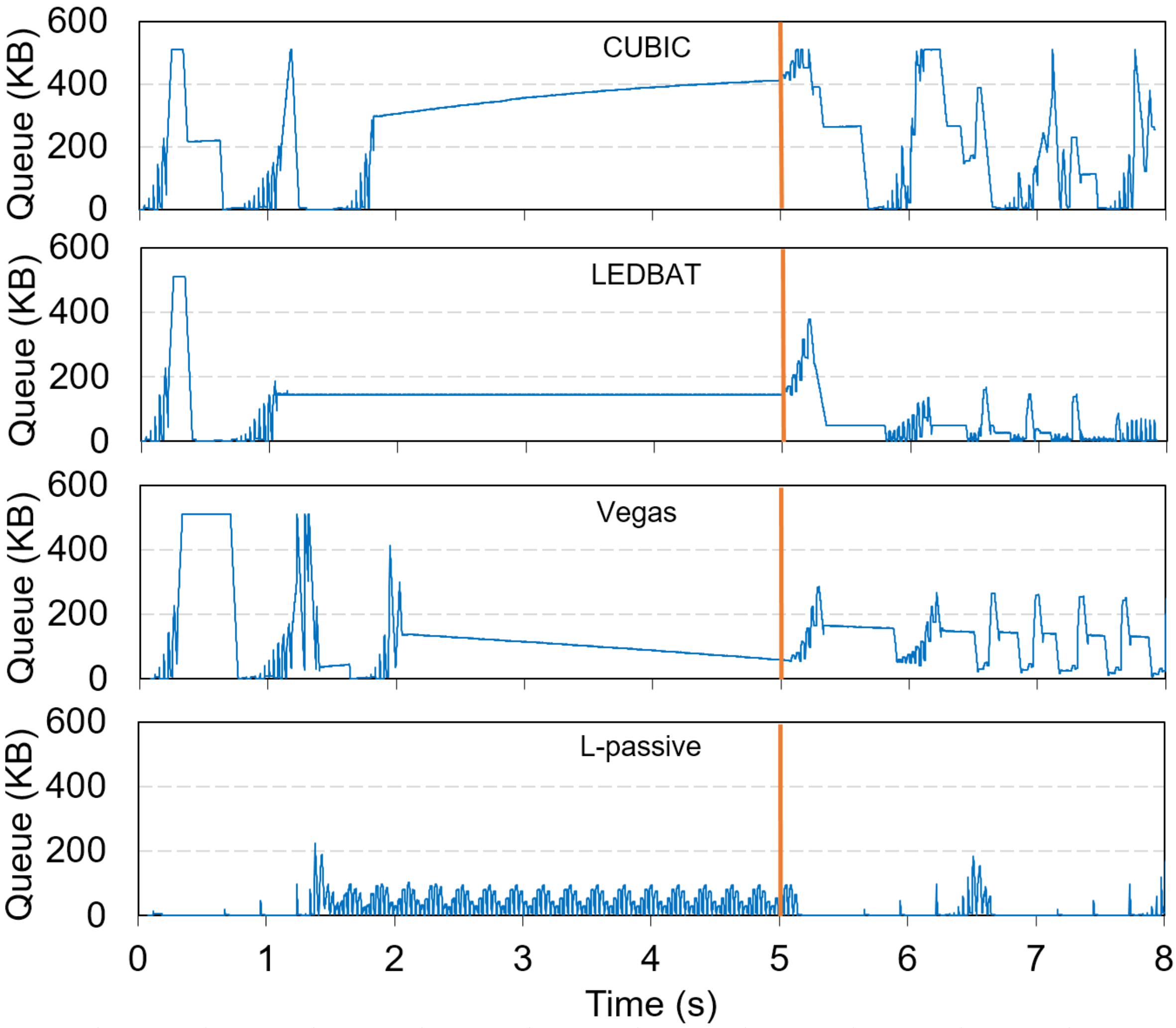


Figure 55. Queuing behavior of protocols

4.2.7 Summary and Discussion

We summarize the results of our evaluation (only *L-passive*) in Table II. The table shows our performance for foreground long flows (i.e., T_{long}) and background flows (i.e., T_{bg}), relative to CUBIC and the best performing LPT protocol in each test.

On the one hand, prevalent existing protocols (*e.g.,* CUBIC) are agnostic of network objectives and perform poorly for foreground traffic, especially at moderate and high loads (FCT and T_{long} in Table II). On the other hand, existing LPT protocols do not work well in cellular networks, are overly conservative, and perform poorly for background traffic, especially at low loads when there is spare capacity (T_{bg} in Table II). *Legilimens* outperforms CUBIC in foreground performance (especially at moderate and high loads) and outperforms LPT in background performance (especially at low loads). There is limited opportunity to improve foreground performance at low loads as there is less contention for resources. Similarly, there is limited opportunity to improve background performance at high loads as there is less spare capacity.

TABLE II

SUMMARY OF RESULTS (*L-passive*)

Test-bed	**Load**	**vs. CUBIC**			**vs. LPT**		
		FCT	T_{long}	T_{bg}	*FCT*	T_{long}	T_{bg}
Real	Low	46%	55%	-44%	-4%	-9%	97%
	Moderate	48%	117%	-66%	6%	6%	-44%
	High	29%	62%	-63%	29%	9%	-48%
Emu	Low	-9%	44%	-45%	25%	22%	209%
	Moderate	-16%	43%	-57%	35%	13%	34%
	High	-2%	44%	-57%	11%	12%	27%
Sim	Low	31%	29%	0%	-5%	-12%	642%
	Moderate	18%	75%	-34%	-6%	3%	427%
	High	11%	63%	-14%	-13%	-21%	53%

4.2.8 Conclusion

We proposed *Legilimens*, an agile transport protocol for background applications in cellular networks. *Legilimens* quickly and accurately estimates capacity and busyness, and uses the estimates to achieve two conflicting objectives — quickly yield to foreground traffic and efficiently capture spare capacity. Using a *novel* algorithm to estimate capacity and load, *Legilimens* achieves the two objectives and outperforms known protocols, which achieve either one of the objectives but not both. As background applications become more reliant on cellular networks, schemes such as *Legilimens* are needed to enforce high-level objectives that accommodate the needs of users (foreground applications), content providers (background applications), and network operators (resource utilization). We plan to explore techniques to optimize *Legilimens* for uplink cellular traffic and evaluate its performance in wired and WiFi networks in the future.

CHAPTER 5

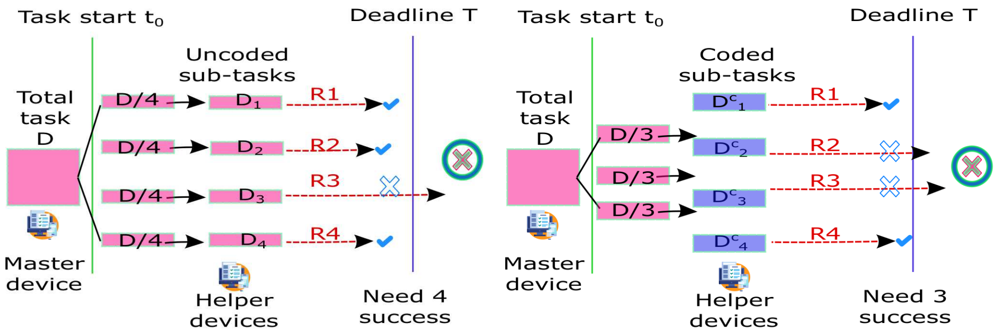
OPTIMAL CODED COMPUTATIONS WITH HARD DEADLINES

Coded distributed computation provides promises for conducting large scale machine learning algorithms while preserving system robustness against stragglers, which are the devices significantly slower than average due to the heterogeneous nature of them. However, the optimal coding strategies remains unclear when given a hard deadline. In this chapter, we consider a distributed computation system, where a master device divides computationally intensive tasks into sub-tasks and offloads them to a group of helper devices. We consider heterogeneous computation capabilities of helper devices and assume random processing time on each device. Accordingly, we characterize the performance of uncoded and (n, k) MDS (Maximum Distance Separable) coded computation. Furthermore, by taking into account the deadline, we develop optimal k for (n, k) MDS coded computation to maximize the probability of meeting the deadline. Obtaining the optimal k requires integer programming which could be time consuming, thus, we develop approximated solution k with much less time complexity to obtain. Simulation results confirm our analysis and show that our approximated solution achieves very close performance to the optimal one, and outperforms the baseline.

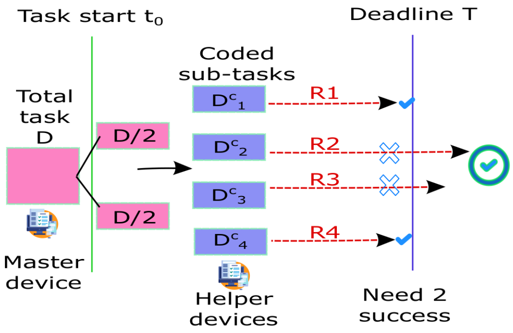
5.1 Background

The increasing number of machine learning algorithms require computationally intensive calculations, which could be challenging to be carried out by a single device. In recent years, distributed computation frameworks such as Spark [18] and MapReduce [19] provides promises to address this challenge, where a master device can divide computation intensive tasks into small sub-tasks and allocate sub-tasks to a group of helper devices. Furthermore, coding strategies have been applied in distributed computation systems to provide resiliency against the stragglers effect that could speed up the whole computation process. However, when there is a hard deadline set for the computation tasks, arbitrary coding strategies could delay the process and result in missing the deadline, which cause the whole computation tasks to fail. Therefore, it is crucial to develop optimal coding strategies for a distributed computation system with hard deadline. In this chapter, we consider a (n, k) MDS-coded distributed computation system with hard deadline, and our goal is to develop optimal k^* so that the probability of meeting the deadline is maximized.

Given k symbols, a (n, k) MDS code introduces redundancy into the symbols and encodes them into n encoded symbols using some encoding function. Upon receiving any k out of n encoded symbols, one can recover the original k symbols. In distributed computing system, we can use (n, k) MDS code to introduce redundancy into sub-tasks and thus increase the system's resiliency against stragglers. In order to better explain the problem, let us consider an example shown in Figure 56. In this setup, there is one master device and four helper devices. Master device divides the whole task D into sub-tasks and allocates them to the helpers. Helper device i ($i = 1, 2, 3, 4$) calculates the assigned sub-tasks, which takes a random R_i time to finish, and returns the results to the master device when it is done. When the process starts at time t_0, there is a hard deadline set at time T, and everything received after the deadline will be discarded. In Figure 56(a), master device uses uncoded computation scheme. It divides the whole task D into four equal-sized sub-tasks (each wth size $D/4$) and allocates D_i to



(a) Uncoded computation (b) (4,3) MDS coded computation



(c) (4,2) MDS coded computation

Figure 56. An canonical example of a distributed computing system consisting of one master device and four helper devices. There is a hard deadline T set for the whole task and master device applies (a) uncoded computation, (b) coded computation using (4,3) MDS codes and (c) coded computation using (4,2) MDS codes.

helper i ($i = 1, 2, 3, 4$), respectively. Since the master device uses uncoded computation scheme, it has to wait for all four helpers' results in order to complete the whole task D. However, if there is a straggler in the system, say the third helper as shown in Figure 56(a), its result cannot be received by the master device by the deadline T, thus, the whole task fails. With coding, the computation could speed up. However, arbitrary coding strategies may also result in task failure. For instance, in Figure 56(b), master device uses (4,3) MDS coded computation scheme. It divides the whole task D into three equal-sized sub-tasks (each with size $D/3$) and encode them to obtain four equal-sized coded sub-tasks D_i^c, (each with size $D/3$, $i = 1, 2, 3, 4$). It then allocates D_i^c to helper i respectively, and wait for their returned results. In this scenario, since master device applies (4,3) MDS codes, although it only needs any three out of four helpers' results in order to complete the whole task, each sub-task D_i^c has a increased size from $D/4$ to $D/3$ as compared to the uncoded case. Therefore, helpers are likely to spend more time processing sub-tasks. Thus, as shown in Figure 56(b), the whole task may still fail. In Figure 56(c), master device uses $(4, 2)$ MDS codes. Thus, helper i receives encoded sub-task D_i^c of size $D/2$ ($i = 1, 2$). Although each helper's computation burden has been increased as compared to the previous two cases, the master device only needs to wait for any two out of four helpers' results in order to complete the whole task. And if it successfully receives them by the deadline T, the whole task succeeds, as shown in Figure 56(c).

As shown from the example in Figure 56, coding introduces redundancy into the sub-tasks to increase resiliency against stragglers, yet increases the computation burden for each helper as compared to the uncoded case. As a result, this may increase the task processing time on each helper and decrease the probability of meeting the deadline. Therefore, there is a trade-off between system robustness and task success probability. In this chapter, we consider a distributed computation system using (n, k) MDS codes, whose task is to solve one of the most important calculations in machine learning algorithms: matrix multiplication, by a given deadline T. Assuming random task processing time on each device, our goal is to develop the optimal k^* so that the probability of meeting the deadline is maximized. The following are the key contributions of this work:

- We first consider matrix-vector multiplication problem and characterize the probability of meeting the deadline using uncoded and (n, k) MDS coded computation. We further characterize the optimal k^* for (n, k) MDS coded computation so that the probability of meeting the deadline is maximized.
- We then expand the same analysis and characterization to a more general matrix-matrix multiplication scenario.
- Obtaining the optimal k^* for (n, k) MDS codes requires integer programming, which could be time consuming. Thus, we develop an approximated solution $\hat{k}$ that has much less computation complexity.
- Simulation results confirm our analysis and show that our approximated solution achieves very close performance to the optimal one, and outperforms the baseline.

5.2 Related Work

Distributed computing system provides possibilities to process computation intensive tasks that can be too large to be done by a single device. However, it is adversely affected by anomalous system behavior and bottlenecks [134], such as unpredictable latency of stragglers in the system. It is pointed out in [135] that the causes for stragglers include run-time contention resources, disk failures, varying bandwidth and congestion in network and, imbalance in task workload.

The popular approach to address the straggler problem is coded computation [136–141]. More recently, some coding schemes are proposed for appropriate matrix multiplication. Dutta *et al.* [142] propose ”Short-Dot”, which introduces additional sparsity to encoded matrices to speed up the computation of linear transforms of long vectors. Tandon *et al.* [143] proposes Gradient coding mechanism to mitigate stragglers and increase tolerance to failure for synchronous gradient descent (SGD) calculation. Lee *et al.* [144] propose a product code for large matrix multiplication, Yu *et al.* [145] propose polynomial codes which is shown to achieve the optimum recovery threshold, defined as the minimum number of helpers needed in order to fully recover the whole task. Later Dutta *et al.* [146] develop ”MatDot” and ”PolyDot” codes which further decrease the recovery threshold at the cost of increased communication cost. As compared to this line of work, we focus on choosing the optimal k^* in existing (n, k) MDS codes to maximize the probability of success in distributed computing system with a hard deadline.

Another line of work focuses on the latency analysis of coded distributed computing system. Lee *et al.* [147] study the latency improvement of coded computation over uncoded ones in matrix multiplication and data shuffling. Lee *et al.* [148] show that erasure codes achieve better latency performance for distributed storage systems. Joshi *et al.* [149] show that coding in distributed storage reduces expected download time, in addition to providing reliability against disk failures. Kadhe *et al.* [150] study the performance of availability codes in reducing the download time in distributed storage system. As compared to this line of work, we consider there is a hard deadline for the computation and focus on the probability of meeting the deadline using coded computation scheme.

The problem of incorporating hard deadline into distributed computing system has been studied recently. Xing *et al.* [151] focus on distributed edge computing and develop a predictive edge computing framework with hard deadlines by taking into account heterogeneity in helper devices. Dutta *et al.* [152] study the convolution of two long vectors and show that coding can dramatically improve the probability of finishing the computation within a target deadline. Ostovari *et al.* [153] study the problem of efficient broadcasting with deadline constraints and propose a deadline-aware heuristic to solve this problem with linear coding. As compared to this line of work, we focus on (n, k) MDS codes and develop optimal k^* to maximize the probability of meeting deadlines.

5.3 System Model

Topology and coding schemes We consider a distributed computing framework illustrated in Figure 56 with one master device and n helper devices. The master device could use uncoded and coded schemes to process the distributed computing problem. In particular, if the master device uses uncoded scheme, it divides the whole computing task into n equal-sized sub-tasks and allocates each sub-task to each of the helper device. The master device waits for all the helpers' results in order to recover the whole task result. As for the coded scheme, we focus on (n, k) MDS codes, where master device first divides the whole task into k equal-sized sub-tasks and encodes them into n coded sub-tasks. Master device allocates each encoded sub-task to each of the helper and waits for the first k our of n helpers' result in order to recover the whole task result.

Computation tasks. In this chapter, we focus on one of the most fundamental computation tasks in machine learning: matrix multiplication, and develop our analysis for that. In particular, we first look at matrix-vector multiplication where the master device would like to calculate $y = Ax$ with $A \in \mathcal{R}^{n\times m}$, $x \in \mathcal{R}^{m\times 1}$ and $y \in \mathcal{R}^{n\times 1}$. Then we generalize the calculation to matrix-matrix multiplication where the master device would like to calculate $y = A^T B$ where $A \in \mathcal{R}^{q\times k}$, $B \in \mathcal{R}^{q\times m}$ and $y \in \mathcal{R}^{k\times m}$. In the computation, we define 1 unit of task as the computation of a vector-vector dot product.

Task processing time. We assume there exists a mother run-time distribution $F_{X_i}(t) = Pr[X_i \leq t]$ where the random variable X_i denotes the task processing time of 1 unit of task on helper i, $\forall i \in \{1, 2, \cdots, n\}$. When j units of tasks are assigned on helper i, we assume the distribution of the processing time is a scaled distribution of the mother run-time distribution. In particular, we denote random variable $X_{i,j}$ as the processing time and the CDF of $X_{i,j}$ is $F_{X_{i,j}}(t) = F_{X_i}(\frac{t}{j})$. In our setup, we assume the mother run-time distribution $F_{X_i}(t)$ of each helper i is independent and identically distributed (i.i.d.), thus, we neglect the notation of i and denote the processing time distribution for 1 and j units of tasks on any helper as $F_X(t)$ and $F_{X_j}(t) = F_X(\frac{t}{j})$, respectively.

Deadline and probability of success. We consider there is a hard deadline T for the master device. Only when it finishes the whole task before the deadline, the whole calculation process is regarded as *success*. We assume the data transmission time between the master and helper devices and the results recover time at the master device are negligible. Given deadline T, we define the probability of success Ps as the probability that master receives all the required results from helpers and recovers the whole calculation results before T.

The goal of this chapter is to characterize the probability of success Ps for uncoded and (n, k) MDS coded computation and develop optimal k^* for the MDS codes to maximize probability of success.

5.4 Matrix-vector Multiplication with Hard Deadlines

We first consider the problem of matrix-vector multiplication where the master device would like to calculate matrix-vector multiplication $y = Ax$ with $A \in \mathcal{R}^{n\times m}$, $x \in \mathcal{R}^{m\times 1}$ and $y \in \mathcal{R}^{n\times 1}$. We characterize the probability of success Ps when master device uses uncoded and (n, k) MDS coded computation schemes, respectively. We further develop the optimal value of k for (n, k) MDS codes in order to maximize Ps.

5.4.1 Uncoded Computation

In the case of uncoded computation, the master device divides A into n row vectors with each vector $A_i \in \mathcal{R}^{1\times m}$. Each sub-task consists of A_i and x for some i and is sent to each helper i. Thus, each helper receives exactly one unit of task.

Lemma 10. *Given deadline T, the probability of success for uncoded computation is*

$$Ps^{uc} = [F_X(T)]^n \tag{5.1}$$

Proof. In uncoded computation, since each helper i receives exactly 1 unit of task, the probability that helper i finishes its sub-task before deadline T is

$$Pr[X_i \leq T] = F_{X_i}(T) = F_X(T), \forall i \tag{5.2}$$

The master device has to wait for all n helpers to finish their sub-task by the deadline in order to finish the whole task. Since we assume the task processing time on each helper is i.i.d. random variable, the probability of success is

$$Ps^{uc} = \prod_{i=1}^{n} Pr[X_i \leq T] = [F_X(T)]^n \tag{5.3}$$

This concludes the proof. □

5.4.2 MDS Coded Computation

In the scenario of (n,k) MDS coded computation, The master device first divides matrix A into k equal-sized sub-matrices with $A_i \in \mathcal{R}^{p\times m}$ ($p < n$, $i = 1,2,\cdots,k$). In this chapter, we assume p divides n so that the number of sub-matrices $k = \frac{n}{p}$ is a valid integer. Then, the master device applies an (n,k) MDS code to each element of the sub-matrices to obtain n encoded sub-matrices, denoted as $[A_1^c, A_2^c, \cdots, A_n^c]$. Note that if $A_i^c = A_i$ for $1 \leq i \leq k$, the master device uses a systematic MDS code. The master device then combines A_i^c and x as the sub-task and sends them to each helper i and waits for the calculated results.

Lemma 11. *Given deadline T, the probability of success Ps^c for MDS coded computation is*

$$Ps^c = \sum_{i=k}^{n} \binom{n}{i} \left[F_X\left(\frac{k}{n}T\right)\right]^i \left[1 - F_X\left(\frac{k}{n}T\right)\right]^{n-i} \tag{5.4}$$

Proof. In MDS coded computation, since the whole task is divided into k equal-sized sub-tasks, each helper i receives n/k units of tasks. The processing time on each helper i is therefore a random variable $X_{i,n/k}$. According to assumption illustrated in *Task processing time* in Section 5.3, the probability that each helper i finishes its sub-task before the deadline T is

$$Pr[X_{i,\frac{n}{k}} \leq T] = F_{X_{i,\frac{n}{k}}}(T) = F_X(\frac{k}{n}T), \forall i \tag{5.5}$$

Upon receiving any k helpers' calculation results, the master device can use the decoding algorithm to recover the whole calculation result. Thus, the probability of success is,

$$\begin{aligned} Ps^c &= Pr[\text{at least k helpers finish before T}] \\ &= \sum_{i=k}^{n} \binom{n}{i} \left[Pr[X_{i,\frac{n}{k}} \leq T]\right]^i \left[1 - Pr[X_{i,\frac{n}{k}} \leq T]\right]^{n-i} \\ &= \sum_{i=k}^{n} \binom{n}{i} \left[F_X\left(\frac{k}{n}T\right)\right]^i \left[1 - F_X\left(\frac{k}{n}T\right)\right]^{n-i} \end{aligned} \tag{5.6}$$

This concludes the proof. □

Now that we have obtained the expression for probability of success for both uncoded and (n, k) MDS coded scenarios, we can observe that Ps^{uc} solely depends the nature of helper devices such as the task processing time, number of helpers, and the deadline T. However, Ps^c not only depends on the factors Ps^{uc} does, but also depends on k. In the next section, we derive the optimal value of the hyperparamter k for (n, k) MDS coded computation such that the probability of success Ps^c is maximized.

5.4.3 Optimal Coded Computation for Matrix-vector Multiplication and Its Approximation

The optimization problem is the following,

$$\begin{aligned} &\max_{k} \sum_{i=k}^{n} \binom{n}{i} \left[F_X\left(\frac{k}{n}T\right)\right]^i \left[1 - F_X\left(\frac{k}{n}T\right)\right]^{n-i} \\ &\text{s.t. } k \in \{1, 2, \cdots, n\} \end{aligned} \tag{5.7}$$

It can be observed that this is an integer optimization problem and the optimal solution can be obtained through brute force method. That is, we iterating over all possible values of k and choose the optimal k^* that maximizes the objective function in Equation 5.7. This is a time consuming method to obtain the optimal solution.

Alternatively, we can bound the objective function in Equation 5.7 and use the solution that maximizes the bound to approximate the optimal solution to the original problem.

In this problem, let the Bernoulli random variable Y_i denote if helper i finishes the sub-tasks before deadline T, that is,

$$\begin{cases} Y_i = 1, \text{if helper } i \text{ meets the deadline} \\ \\ Y_i = 0, \text{if helper } i \text{ misses the deadline} \end{cases} \tag{5.8}$$

Thus, $Pr[Y_i = 1] = p$ and $Pr[Y_i = 0] = 1 - p$, where $p = F_X\left(\frac{k}{n}T\right)$. We assume Y_i is i.i.d. among all the users.

Therefore, the optimization problem in Equation 5.7 can be equivalently expressed as

$$\begin{aligned} &\max_{k} \; Pr(H \geq k) \\ &\quad \text{s.t. } k \in \{1, 2, \cdots, n\} \end{aligned} \tag{5.9}$$

where $H = \sum_{i=1}^{n} Y_i$ is the binomial random variable that denotes the number of helpers that meet the deadline T. It can be observed that $Pr(H = k) = \binom{n}{k}(p)^k(1-p)^{n-k}$ where $p = F_X(\frac{k}{n}T)$.

In the objective function of this problem, since $H = \sum_{i=1}^{n} Y_i$, and Y_i is i.i.d. and bounded ($0 \leq Y_i \leq 1$), we could apply Hoeffding inequality [154] to the objective function which provides an upper bound on the probability that the sum of bounded independent random variables deviates from its expected value by more than a certain amount.

Proposition 12. *Applying Hoeffding inequality on the objective function in the optimization problem stated in Equation 5.9, the approximated optimal solution is*

$$\hat{k} = \lfloor D_p^{-1}(\frac{1}{n}) + 0.5 \rfloor \tag{5.10}$$

where $p = F_X(\frac{k}{n}T)$*,* $D_p(x)$ *is the first order derivative of function p with respect to* x*, i.e.,* $y = D_p(x) = \frac{\partial p}{\partial x}$*. And* $D_p^{-1}(y)$ *is the inverse function of* $D_p(x)$*, i.e., if* $y = D_p(x)$*, then* $x = D_p^{-1}(y)$*.*

Proof. Applying Hoeffding inequality to the special case of binomial distribution random variable H with Probability Mass Function (PMF) $Pr(H = k) = \binom{n}{k}(p)^k(1-p)^{n-k}$ [155], where $p = F_X(\frac{k}{n}T)$, we have

i) when $k > np$,

$$Pr(H \geq k) \leq e^{-2(\frac{k}{n}-p)^2 n} \tag{5.11}$$

and ii) when $k < np$,

$$Pr(H \leq k) \leq e^{-2(p-\frac{k}{n})^2 n} \tag{5.12}$$

that is,

$$Pr(H \geq k) = 1 - Pr(H \leq k) \geq 1 - e^{-2(p-\frac{k}{n})^2 n} \tag{5.13}$$

Equation 5.11 and Equation 5.13 provide bounds on $Pr(H \geq k)$ depends on the value of k. In our approximation process, we maximize the upper bound of $Pr(H \geq k)$. In particular,

i) when $k > np$, we

$$\max_{k} \; e^{-2(\frac{k}{n}-p)^2 n} \qquad \text{s.t. } k \in 1, 2, \cdots, n \tag{5.14}$$

which is equivalently expressed as,

$$\min_{k} \; (\frac{k}{n} - p)^2 \qquad \text{s.t. } k \in 1, 2, \cdots, n \tag{5.15}$$

And ii) when $k < np$, we

$$\max_{k} \; 1 - e^{-2(p-\frac{k}{n})^2 n} \qquad \text{s.t. } k \in 1, 2, \cdots, n \tag{5.16}$$

which is equivalently expressed as,

$$\max_{k} \; (p - \frac{k}{n})^2 \qquad \text{s.t. } k \in 1, 2, \cdots, n \tag{5.17}$$

In order to solve problem Equation 5.15 and Equation 5.17, We further first approximate k as real number k' and try to solve the optimization problems in Equation 5.15 and Equation 5.17 with compact solutions and then round the solution of k' to its nearest integer to get an approximated solution $\hat{k}$ to the original problem. Note that $p = F_X(\frac{k'}{n}T)$ in these problems and we assume p is continuous and first order differentiable on k'.

In particular,

i) when $k' > np$, the optimization problem is

$$\begin{aligned} &\min_{k'} \ (\frac{k'}{n} - p)^2 \\ &\text{s.t. } 1 \leq k' \leq n \end{aligned} \tag{5.18}$$

and the optimal solution can be obtained according to

$$\frac{\partial(\frac{k'}{n} - p)^2}{\partial k'} = 0 \tag{5.19}$$

$$2(\frac{k'}{n} - p)(\frac{1}{n} - \frac{\partial p}{\partial k'}) = 0 \tag{5.20}$$

Since k' has to satisfy $k' > np$, the optimal solution is

$$k' = D_p^{-1}(\frac{1}{n}) \tag{5.21}$$

where $D_p(x)$ is the first order derivative of function p with respect to x, i.e., $D_p(x) = \frac{\partial p}{\partial x}$. And $D_p^{-1}(x)$ is the inverse function of $D_p(x)$.

ii) When $k' < np$, the optimization problem is

$$\begin{aligned} &\max_{k'} \ (p - \frac{k'}{n})^2 \\ &\text{s.t. } 1 \leq k' \leq n \end{aligned} \tag{5.22}$$

and the optimal solution can be obtained according to

$$\frac{\partial(p - \frac{k'}{n})^2}{\partial k'} = 0 \tag{5.23}$$

$$2(p - \frac{k'}{n})(\frac{\partial p}{\partial k'} - \frac{1}{n}) = 0 \tag{5.24}$$

Since k' has to satisfy $k' < np$, the optimal solution is

$$k' = D_p^{-1}(\frac{1}{n}) \tag{5.25}$$

Taking into account both of the cases when i) $k > np$ and ii) $k < np$, by Equation 5.21 and Equation 5.25, the optimal solution that maximizes the upper bound of $Pr(H \geq k)$ is

$$k' = D_p^{-1}(\frac{1}{n}) \tag{5.26}$$

Finally, we round k' to its nearest integer $\hat{k}$ to obtain the integer approximated solution to our original problem in Equation 5.9. That is,

$$\hat{k} = \lfloor k' + 0.5 \rfloor = \lfloor D_p^{-1}(\frac{1}{n}) + 0.5 \rfloor \tag{5.27}$$

This concludes the proof. □

By Proposition 12, when the task processing time follows exponential distribution with $p = 1 - e^{-\frac{k'}{n}\lambda T}$ the approximated optimal $\hat{k}$ is

$$\hat{k} = \lfloor \frac{n \ln(\lambda T)}{\lambda T} + 0.5 \rfloor \tag{5.28}$$

where λ is a constant.

And when the task processing time follows shifted exponential distribution with $p = 1 - e^{-\lambda(\frac{k'}{n}T - s)}$, the approximated optimal $\hat{k}$ is

$$\hat{k} = \lfloor \frac{n}{T}(\frac{\ln(\lambda T)}{\lambda} + s) + 0.5 \rfloor \tag{5.29}$$

where λ and s are constants.

5.5 Matrix-matrix Multiplication with Hard Deadlines

In this section, we consider a more general form of matrix multiplication, that is, matrix-matrix multiplication where the master device would like to calculate $y = A^T B$ with $A \in \mathcal{R}^{q \times r}$, $B \in \mathcal{R}^{q \times m}$ and $y \in \mathcal{R}^{r \times m}$. Let us denote the i^{th} column of matrix A and B by $a_i \in \mathcal{R}^{q \times 1}$ and $b_i \in \mathcal{R}^{q \times 1}$, respectively. Thus, we can rewrite matrix A and B as $A = [a_1, a_2, \cdots, a_r]$ and $B = [b_1, b_2, \cdots, b_m]$, respectively. The matrix multiplication is then expressed as

$$
\begin{aligned}
A^T B &= \begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_r^T \end{bmatrix} \begin{bmatrix} b_1 & b_2 & \cdots & b_m \end{bmatrix} \\
&= \begin{bmatrix} a_1^T b_1 & a_1^T b_2 & \cdots & a_1^T b_m \\ a_2^T b_1 & a_2^T b_2 & \cdots & a_2^T b_m \\ \vdots & \vdots & \vdots & \vdots \\ a_r^T b_1 & a_r^T b_2 & \cdots & a_r^T b_m \end{bmatrix}
\end{aligned}
\tag{5.30}
$$

Note that since computing $A^T B$ requires rm dot products of $a_i^T b_j$, the total units of tasks are therefore rm. For the ease of explanation, we assume the total number of helpers $n = rm$.

5.5.1 Uncoded Computation

In the case of uncoded computation, master device divides matrix A and B into r and m equal-sized parts, respectively. Each part is $A_i = a_i$, $i = 1, 2, \cdots, r$ and $B_j = b_j$, $j = 1, 2, \cdots, m$. Master device then assigns one dot product of $a_i^T b_j$ as the sub-task to each of the helper respectively. Thus, each helper receives exactly 1 unit of task.

Lemma 13. *Given deadline T, the probability of success for uncoded computation is*

$$Ps^{uc} = [F_X(T)]^{rm} \tag{5.31}$$

Proof. The proof follows exactly the same logic as the proof for Lemma 10 with $n = rm$ in this case. □

5.5.2 One-dimensional MDS Coded Computation

In one-dimensional MDS coded computation, master device uses MDS codes to encode matrix A only. More specifically, master device considers the multiplication $A^T B$ as m instances of small matrix-vector multiplications, that is,

$$A^T B = [A^T b_1, A^T b_2, \cdots, A^T b_m] \tag{5.32}$$

In this setup, rm helpers are divided into m groups of size r. The number of groups m is fixed and r helpers in each of the m group are assigned to compute $A^T b_j$ for some $j \in \{1, 2, \cdots, m\}$. Within each group, it can be observed that r helpers are performing matrix-vector multiplication tasks. Thus, according to Section 5.4, we can apply (r, k) MDS codes in each group to compute $A^T b_j$. For example, suppose the first group is assigned to compute $A^T b_1$. The master device would divide matrix A into k equal-sized sub-matrices $[A_1, A_2, \cdots, A_k]$. Then it applies (r, k) MDS codes to each of A_i to obtain r encoded sub-matrices $[A_1^c, A_2^c, \cdots, A_r^c]$. Lastly, master device assigns computation of sub-task $A_i^{cT} b_1$

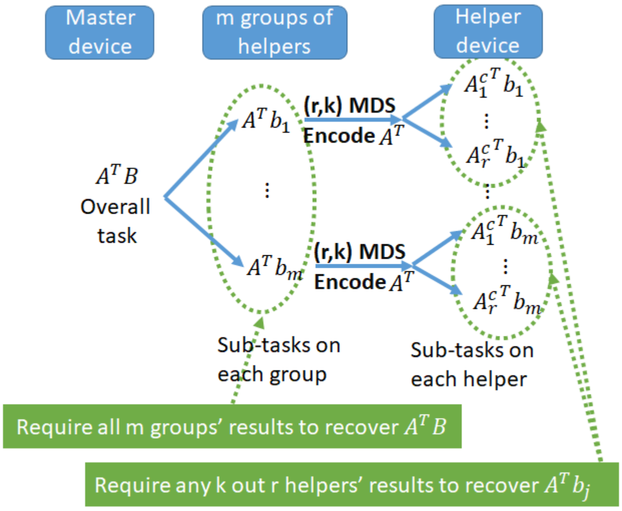


Figure 57. Illustration of one-dimensional MDS coded computation for matrix-matrix multiplication $A^T B$.

to the i^{th} helper in this group. Similarly, the r helpers in the j^{th} group are assigned to jointly compute $A^T b_j$. In this setup, we assume master device applies same (r, k) MDS codes to every group and the goal is to obtain the optimal k^* so that the probability of success is maximized. The one-dimensional MDS coded scheme is illustrated in Figure 57.

Lemma 14. *Let master device apply same (r, k) MDS codes to each of the m groups to compute $A^T B$, given deadline T, the probability of success Ps^c using one-dimensional MDS coded computation is*

$$Ps^c = \left[\sum_{i=k}^{r} \binom{r}{i} \left[F_X \left(\frac{k}{r} T \right) \right]^i \left[1 - F_X \left(\frac{k}{r} T \right) \right]^{r-i} \right]^m \tag{5.33}$$

Proof. The probability of success of the whole calculation depends on the probability of success of sub-tasks in each of the m groups. By Lemma 11, the probability of success within each group $g \in \{1, 2, \cdots, m\}$ is

$$Ps_g^c = \sum_{i=k}^{r} \binom{r}{i} \left[F_X \left(\frac{k}{r} T \right) \right]^i \left[1 - F_X \left(\frac{k}{r} T \right) \right]^{r-i} \tag{5.34}$$

Thus, the probability of success of the whole calculation task is

$$\begin{aligned} Ps^c &= Pr[\text{All } m \text{ groups finish sub-tasks before } T] \\ &= \prod_{g=1}^{m} Ps_g^c \end{aligned} \tag{5.35}$$

This concludes the proof. □

5.5.3 Two-dimensional MDS Coded Computation

In two-dimensional MDS coded computation, we apply MDS codes to encode both of matrix A and B Recall that $A^T B = [A^T b_1, A^T b_2, \cdots, A^T b_m]$. To perform two-dimensional MDS coded computation, master device first applies (M, m) $(M \geq m)$ MDS codes to each column of matrix B to obtain $B^c = [b_1^c, b_2^c, \cdots, b_M^c]$. Then, it considers the multiplication of $A^T B^c$ as M instances of small matrix-vector multiplications, that is,

$$A^T B^c = [A^T b_1^c, A^T b_2^c, \cdots, A^T b_m^c, \cdots, A^T b_M^c] \tag{5.36}$$

Note that in order to recover the results for $A^T B$, master device only needs any m out of M elements of $A^T b_j^c$ $(j = 1, 2, \cdots, M)$.

Thus, to carry out the two-dimensional MDS coded computation, master device divides rm helpers into $M \geq m$ groups of size $n' = \lfloor \frac{rm}{M} \rfloor \leq r$. Each group is assigned to compute $A^T b_j^c$ for some $j \in \{1, 2, \cdots, M\}$. Within each group, it can be observed that n' helpers are performing matrix-vector multiplication. Thus, similar to what we have done in one-dimensional MDS coded computation, master device applies (n', k) MDS codes in each group to compute $A^T b_j^c$. It can observed from the construction of two-dimensional MDS coded computation. Master device only needs any m out of M groups' results to recover the whole computation task. And within each group, master device only needs any k out of n' helpers' results to recover the group's sub-task $A^T b_j^c$.

In this setup, we assume every group has the same number of helpers n' and master device applies the same (n', k) MDS codes to each group. Therefore, given total number of rm helpers and matrix-matrix multiplication task $A^T B$, the goal is to obtain the optimal M and k so that the probability of success is maximized.

The two-dimensional MDS coded scheme is illustrated in Figure 58.

Lemma 15. *Let M denote the number of groups master device divides the rm helpers and let master device apply the same (n', k) ($n' = \lfloor \frac{rm}{M} \rfloor$) MDS codes to the helpers in each group, given deadline T, the probability of success Ps^c using two-dimensional MDS coded computation is*

$$Ps^c = \sum_{i=m}^{M} \binom{M}{i} \left(Ps_g^c\right)^i \left(1 - Ps_g^c\right)^{M-i} \tag{5.37}$$

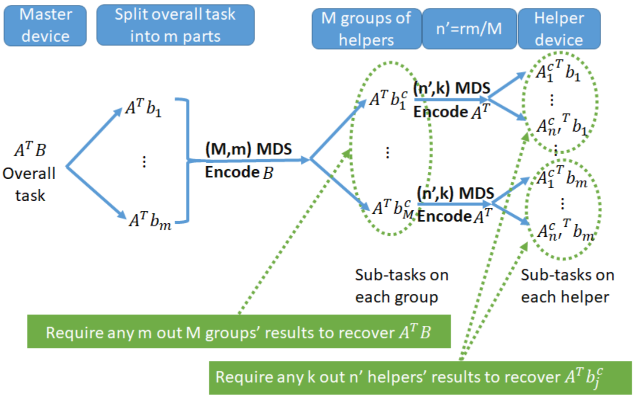


Figure 58. Illustration of two-dimensional MDS coded computation for matrix-matrix multiplication $A^T B$.

where Ps_g^c *is the success probability of each group* $g \in \{1, 2, \cdots, M\}$*. That is,*

$$Ps_g^c = \sum_{i=k}^{n'} \binom{n'}{i} \left[F_X\left(\frac{k}{r}T\right)\right]^i \left[1 - F_X\left(\frac{k}{r}T\right)\right]^{n'-i} \tag{5.38}$$

where $n' = \frac{rm}{M}$.

Proof. By the construction of two-dimensional MDS coded computation, the probability of success of the whole calculation depends on the probability that master device receives any m out of M groups' calculation results by the deadline T. By Lemma 11, the probability of success within each group $g \in \{1, 2, \cdots, m\}$ is

$$Ps_g^c = \sum_{i=k}^{n'} \binom{n'}{i} \left[F_X\left(\frac{k}{r}T\right)\right]^i \left[1 - F_X\left(\frac{k}{r}T\right)\right]^{n'-i} \tag{5.39}$$

Thus, the probability of success of the whole calculation task is

$$\begin{aligned} Ps^c &= Pr[\text{At least } m \text{ groups finish sub-tasks before } T] \\ &= \sum_{i=m}^{M} \binom{M}{i} \left(Ps_g^c\right)^i \left(1 - Ps_g^c\right)^{M-i} \end{aligned} \tag{5.40}$$

This concludes the proof. □

5.5.4 Optimal Coded Computation for Matrix-matrix Multiplication and Its Approximation

In this section, we focus on maximizing the probability of success of MDS coded computation and develop i) optimal and approximated value of k for one-dimensional coded computation and ii) optimal and approximated value for M and k for two-dimensional coded computation.

5.5.4.1 Optimal and Approximated k for One-dimensional Coded Computation

The optimization problem is

$$\begin{aligned} &\max_{k} \left[\sum_{i=k}^{r} \binom{k}{i} \left[F_X\left(\frac{k}{r}T\right) \right]^i \left[1 - F_X\left(\frac{k}{r}T\right) \right]^{r-i} \right]^m \\ &\text{s.t. } k \in \{1, 2, \cdots, r\} \end{aligned} \tag{5.41}$$

This is an integer programming problem, the optimal solution can be found by iterating all the possible value of k in each group.

As for the approximated solution, We can find out that in order to solve the problem, we only need to maximize $Ps_g^c = \sum_{i=k}^{r} \binom{r}{i} \left[F_X\left(\frac{k}{r}T\right)\right]^i \left[1 - F_X\left(\frac{k}{r}T\right)\right]^{r-i}$ with respect to k. Ps_g^c is nothing but the probability of success within each group g that is performing matrix-vector multiplication using (r, k)

MDS codes. Therefore, assuming $p = F_X(\frac{k'}{r}T)$ is continuous and first order differentiable on k', by Proposition 12, the approximated optimal solution is

$$\hat{k} = \lfloor D_p^{-1}(\frac{1}{r}) + 0.5 \rfloor \tag{5.42}$$

where $D_p(x)$ is the first order derivative of function p with respect to x, i.e., $y = D_p(x) = \frac{\partial p}{\partial x}$. And $D_p^{-1}(y)$ is the inverse function of $D_p(x)$.

5.5.4.2 Optimal and Approximated M and k for Two-dimensional Coded Computation

The optimization problem is

$$\begin{aligned} &\max_{k,M} \sum_{i=m}^{M} \binom{M}{i} \left(Ps_g^c\right)^i \left(1 - Ps_g^c\right)^{M-i} \\ &\text{s.t. } k \in \{1, 2, \cdots, r\}, M \in \{m, \cdots, rm\} \end{aligned} \tag{5.43}$$

where $Ps_g^c = \sum_{i=k}^{n'} \binom{n'}{i} \left[F_X\left(\frac{k}{r}T\right)\right]^i \left[1 - F_X\left(\frac{k}{r}T\right)\right]^{r-i}$ and $n' = \frac{rm}{M}$.

This is again an integer programming problem, the optimal solution can be found by iterating all the possible value of M and k.

As for the approximated solution, it can be noted that, given the number of groups M, the objective function is maximized if Ps_g^c is maximized. Again, by Proposition 12, under the same assumption of continuity, the approximated optimal solution that maximizes Ps_g^c is

$$\hat{k} = \lfloor D_p^{-1}(\frac{1}{n'}) + 0.5 \rfloor = \lfloor D_p^{-1}(\frac{M}{rm}) + 0.5 \rfloor \tag{5.44}$$

where $p = F_X(\frac{n'}{r}T)$

Therefore, once M is decided, the whole solution to the problem can be obtained. In order to obtain the best $(M, \hat{k})$ values, we could iterate over all the possible values of M and find the one $(M, \hat{k})$ pair that maximizes the objective function in Equation 5.43. The search space of M is $O(rm)$.

5.6 Evaluation and Simulation

It can be observed from Section 5.4 and 5.5, the calculation of the optimal k^* depends on the distribution of processing time $F_X(\frac{k}{n}T)$. In this section, we consider two special distributions that characterize the processing time, namely, exponential distribution and shifted exponential distribution. And we derive the optimal and approximated k for matrix-vector multiplication; optimal and approximated M and k for matrix-matrix multiplication. We compare the performance of the approximated solution to the optimal and uncoded solution.

Recently, MDS coded computation scheme in distributed computing system is discussed in [144, 147] for matrix-vector and matrix-matrix multiplication. However, the authors do not optimize k in the (n, k) MDS codes. We consider the MDS coded scheme illustrated in [144, 147] as the baseline and compare it to our approximated solution.

5.6.1 Matrix-vector Multiplication

Let us first consider the matrix-vector multiplication problem illustrated in Section 5.4. In this setup, we consider the number of helpers is $n = 10$.

5.6.1.1 Exponential Distribution

First, let us assume 1 unit of task's processing time on each helper i follows exponential distribution with mean $\frac{1}{\lambda}$. In the evaluation, we consider $\lambda = 1$. The processing time random variable are i.i.d. Thus, the CDF of 1 unit of task's processing time on each helper i is

$$F_{X_i}(t) = F_X(t) = 1 - e^{-\lambda t}, \forall i \tag{5.45}$$

Given dealine T, by Lemma 10 and Lemma 11, the probability of success for uncoded and MDS coded computation is

$$Ps^{uc} = (1 - e^{-\lambda T})^n \tag{5.46}$$

and

$$Ps^{c} = \sum_{i=k}^{n} \binom{n}{i} \left[1 - e^{-\lambda \frac{k}{n} T}\right]^i \left[e^{-\lambda \frac{k}{n} T}\right]^{n-i} \tag{5.47}$$

Figure 59 presents the probability of success Ps using uncoded and MDS coded computation for different deadlines T. It can be observed that, in this example, coded computation always achieves higher probability of success as compared to the uncoded case for a given deadline. As for coded computation, when the deadline is tight, smaller k achieves higher probability of success and when deadline is large, some larger k achieves better performance.

Figure 60 presents the probability of success as k changes. The deadline is $T = 1$. In this specific scenario, it can be observed that the performance of MDS coded computation decreases as k increases. When $k = n$, coded computation becomes the uncoded case.

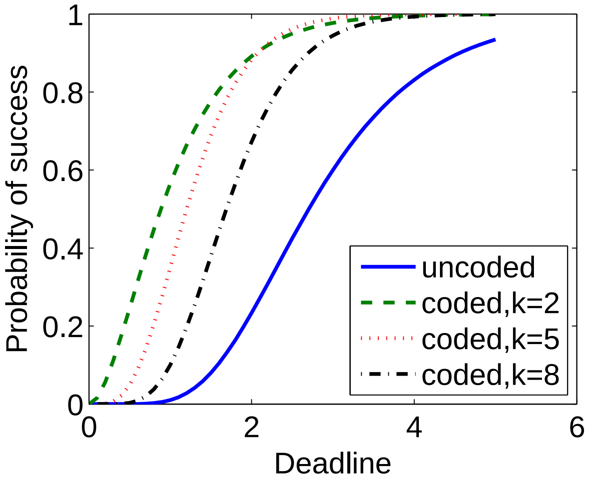


Figure 59. Probability of success versus different deadlines for the case of uncoded and MDS coded computation. The processing time on each helper follows exponential distribution with mean $\frac{1}{\lambda} = 1$.

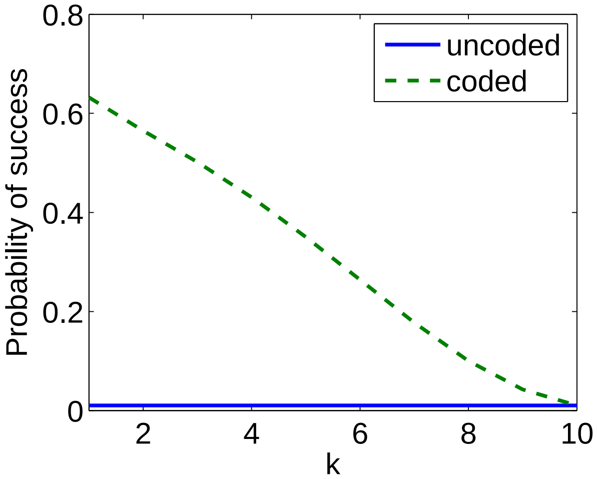


Figure 60. Probability of success versus k in MDS coded computation. The processing time follows exponential distribution with mean $\frac{1}{\lambda} = 1$. The deadline is $T = 1$.

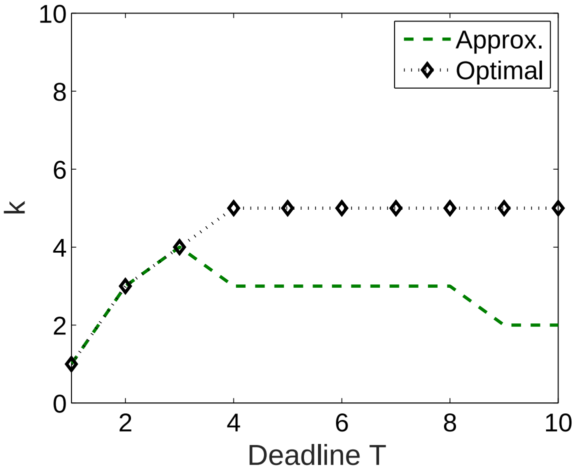


Figure 61. Optimal and approximated k for coded computation. The processing time follows exponential distribution with $\lambda = 1$.

As for k, by Proposition 12, with $p = 1 - e^{-\frac{k'}{n}\lambda T}$ the approximated $\hat{k}$ for MDS coded computation is

$$\hat{k} = \lfloor \frac{n \ln(\lambda T)}{\lambda T} + 0.5 \rfloor \tag{5.48}$$

Figure 61 presents the optimal k^* and approximated $\hat{k}$ for coded computation when the deadline T is changing. It can observed that our approximation calculation is the same as the optimal value for $T = 1, 2$ and 3. For $T > 3$, the approximated value $\hat{k}$ is smaller than the optimal value k^*.

Figure 62 presents the probability of success for different values of deadline T. We compare the MDS coded computation with our approximated $\hat{k}$ to the MDS coded computation with the optimal k^*, baseline $k = 8$ and uncoded computation. It can be observed that coded computation achieve higher probability of success than the uncoded case. Furthermore, the performance of coded computation using

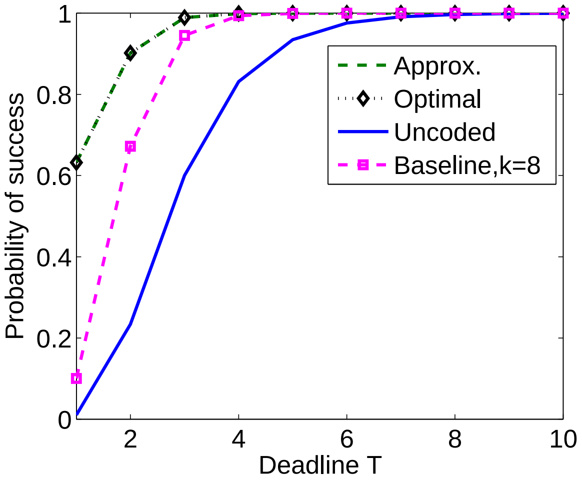


Figure 62. Probability of master device meeting the deadline using coded computation with optimal k^*, approximated $\hat{k}$ and uncoded schemes. The processing time follows exponential distribution with $\lambda = 1$.

our approximated $\hat{k}$ achieves almost the same performance as the optimal solution, and it outperforms the baseline.

5.6.1.2 Shifted Exponential Distribution

Next, we consider another distribution that characterizes the processing time of 1 unit of task on helper i, which is the shifted exponential distribution with CDF as follows

$$F_{X_i^1}(t) = F_X(t) = 1 - e^{-\lambda(t-s)}, \forall t \geq s, \forall i \tag{5.49}$$

where $\lambda > 0$ and $s > 0$ are constant parameters. In the evaluation, we consider $\lambda = 1$ and $s = 2$.

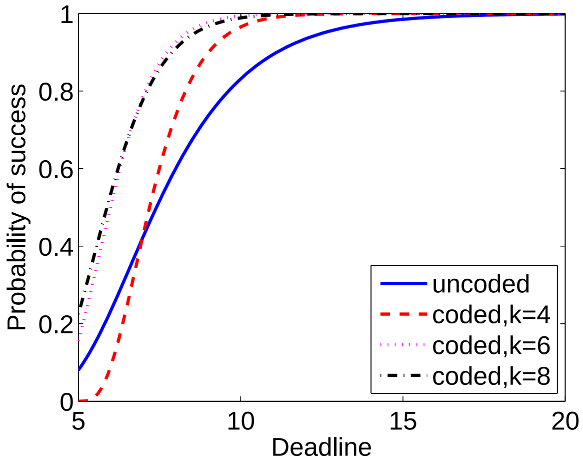


Figure 63. Probability of success versus different deadlines for the case of uncoded and coded computation. The processing time follows shifted exponential distribution with $\lambda = 1$ and $s = 2$.

Given dealine T, by Lemma 10 and Lemma 11, the probability of success for uncoded and MDS coded computation is

$$Ps^{uc} = (1 - e^{-\lambda(T-s)})^n \tag{5.50}$$

and

$$Ps^{c} = \sum_{i=k}^{n} \binom{n}{i} \left[1 - e^{-\lambda(\frac{k}{n}T-s)}\right]^{i} \left[e^{-\lambda(\frac{k}{n}T-s)}\right]^{n-i} \tag{5.51}$$

Figure 63 presents the probability of success using uncoded and MDS coded computation for different deadlines. It can be observed that, when the deadline is tight (T is small), uncoded computation performs better than coded computation with small k. As the deadline increases, coded computation achieves higher probability of success as compared to the uncoded case. This result confirms with our argument that arbitrary MDS coding strategy may not benefit the system.

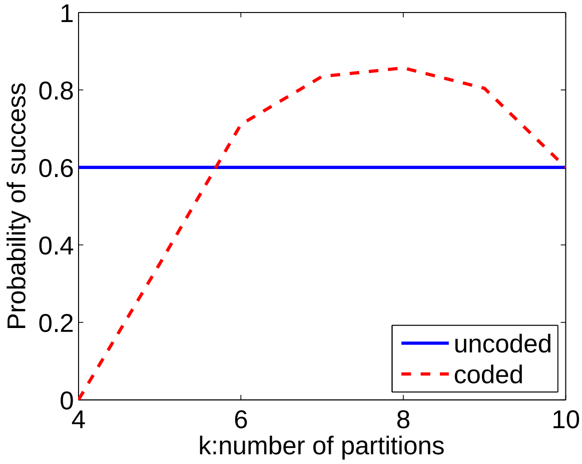


Figure 64. Probability of success versus k in coded computation. The processing time follows shifted exponential distribution with $\lambda = 1$ and $s = 2$. The deadline is $T = 5$.

Figure 64 presents the probability of success as k changes. The deadline is $T = 5$. It can be observed that the performance of coded computation fluctuates as k increases. In this specific example, the probability of success using coded computation first increases as k increase, then achieves maximum value of around 85% when $k = 8$, and decreases after that. On the other hand, the probability of success using uncoded computation is always 60%. As seen, it is crucial to develop optimal k^* to maximize the probability of success.

By Proposition 12, with $p = 1 - e^{-\lambda(\frac{k'}{n}T - s)}$ the approximated $\hat{k}$ for MDS coded computation is

$$\hat{k} = \lfloor \frac{n}{T} (\frac{\ln(\lambda T)}{\lambda} + s) + 0.5 \rfloor \tag{5.52}$$

Figure 65 presents the optimal k^* and approximated $\hat{k}$ for coded computation when the deadline T is changing. It can observed that our approximated solution mimics the optimal one when deadline is

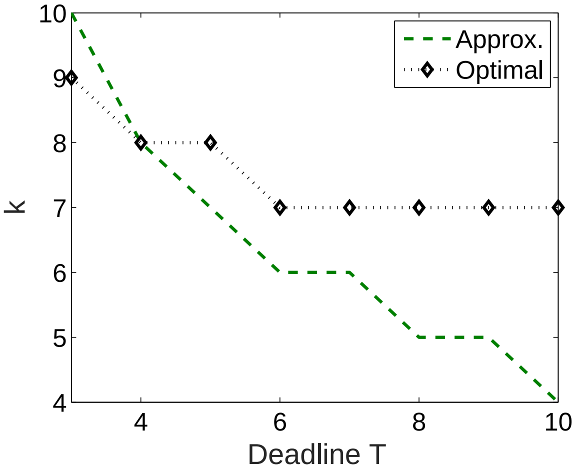


Figure 65. Optimal and approximated k for coded computation. The processing time follows shifted exponential distribution with $\lambda = 1$ and $s = 2$.

tight and deviates from the optimal solution when deadline is large. However, this deviation is tolerable because when deadline T is large, the probability of success at the master device is high for a large range of k values.

Figure 66 presents the probability of success at the master device for different values of T. We compare our approximated solution to the optimal solution, uncoded computation and the baseline with $k = 9$. Similar to the results in Figure 62, it can be observed that coded computation achieves higher probability of success than the uncoded case. Furthermore, the performance of coded computation using our approximated $\hat{k}$ achieves almost the same performance as the optimal solution, and it outperforms the baseline.

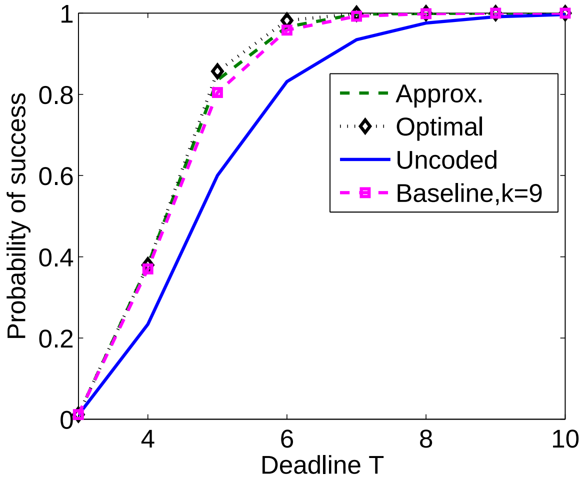


Figure 66. Probability of success using coded computation with optimal k^*, approximated $\hat{k}$ and uncoded schemes. The processing time follows shifted exponential distribution with $\lambda = 1$ and $s = 2$.

5.6.2 Matrix-matrix Multiplication

Now let us consider the matrix-matrix multiplication problem illustrated in Section 5.5. In this setup, we consider number of columns of matrix A and B is $r = 10$ and $m = 10$, respectively. Thus, the number of helpers is $n = rm = 100$.

5.6.2.1 Exponential Distribution

First, let us also assume 1 unit of task's processing time on each helper i follows exponential distribution with mean $\frac{1}{\lambda} = 1$.

$$F_{X_i}(t) = F_X(t) = 1 - e^{-\lambda t}, \forall i \tag{5.53}$$

Given deadline T, for one-dimensional (n, k) MDS coded computation, the approximated optimal $\hat{k}$ is

$$\hat{k} = \lfloor \frac{k \ln(\lambda T)}{\lambda T} + 0.5 \rfloor \tag{5.54}$$

And as for two-dimensional MDS coded computation, given the number of groups M, the approximated $\hat{k}$ in each group is

$$\hat{k} = \lfloor \frac{km \ln(\lambda T)}{M \lambda T} + 0.5 \rfloor \tag{5.55}$$

Figure 67 presents the probability of success using one-dimensional MDS, two-dimensional MDS and uncoded computation schemes. In one-dimensional MDS coding, there are no redundancy group wise, thus the number of groups is $M = m = 10$. As for two-dimensional MDS coding, the number of groups in this simulation is $M = 11$. It can be observed that coded computation outperforms the uncoded one in general. When the deadline T is large, two-dimensional MDS coding with larger k achieves the best performance and when the deadline is small, two-dimensional MDS coding with smaller k achieves the best performance.

Figure 68 presents the probability of success for different number of groups M in two-dimensional MDS coding. $k = 2$ for both one and two-dimensional MDS coding. It can be observed that two-dimensional MDS outperforms the uncoded and one-dimensional MDS coded computation. Furthermore, the performance of two-dimensional MDS coding improves as the number of groups M increases.

Figure 69 presents the optimal and approximated solution to k and M as illustrated in Section 5.5.4 in order to maximize the probability of success. It can be observed that as deadline T increases, for two-dimensional MDS coded computation, our approximated solution is close to the optimal one. On

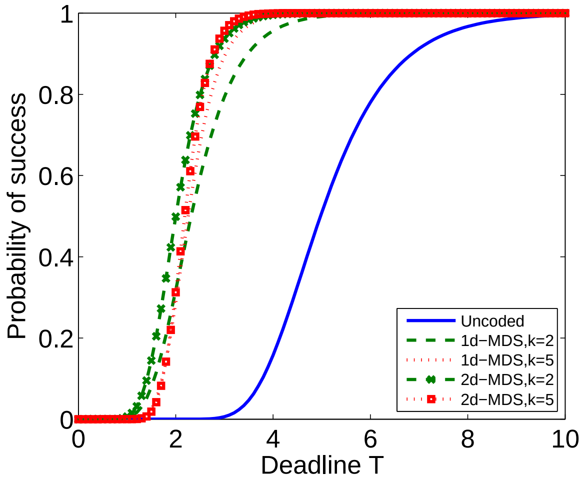


Figure 67. The probability of success using one-dimensional MDS, two-dimensional MDS and uncoded computation schemes. In two-dimensional MDS coded computation, the number of groups is $M = 11$. The processing time follows exponential distribution with $\lambda = 1$.

the other hand, in one-dimensional MDS coded computation, since there are no redundancy group wise, the number of groups is always $M = 10$, and k varies accordingly.

Figure 70 shows the corresponding probability of success using the k and M shown in Figure 69. We also add two-dimensional MDS coded computation with baseline $k = 8$ and the uncoded computation in comparison. It can be observed that coded computation achieves higher probability of success than the uncoded case. Furthermore, our approximated two-dimensional MDS coding achieves almost the same performance as the optimal one and outperforms the one-dimensional MDS coding, which further outperforms the baseline.

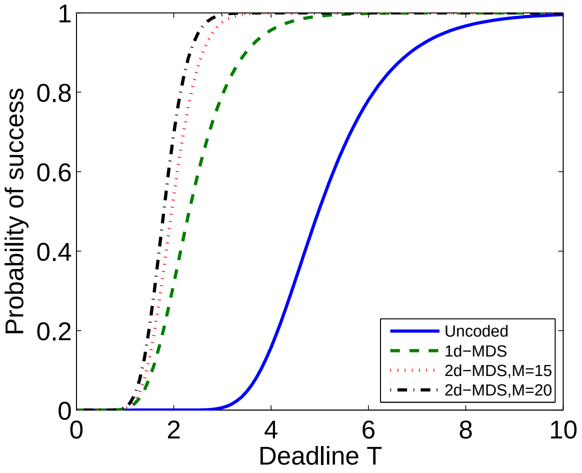


Figure 68. The probability of success using one-dimensional, two-dimensional and uncoded computation schemes. In both one-dimensional and two-dimensional MDS coded computation, $k = 2$. The processing time follows exponential distribution with $\lambda = 1$.

5.6.2.2 Shifted Exponential Distribution

Now let us assume 1 unit of task's processing time on each helper i follows shifted exponential distribution expressed as

$$F_{X_i^1}(t) = F_X(t) = 1 - e^{-\lambda(t-s)}, \forall t \geq s, \forall i \tag{5.56}$$

In the evaluation, we consider $\lambda = 1$ and $s = 2$.

Given deadline T, for one-dimensional (n, k) MDS coded computation, the approximated optimal $\hat{k}$ is

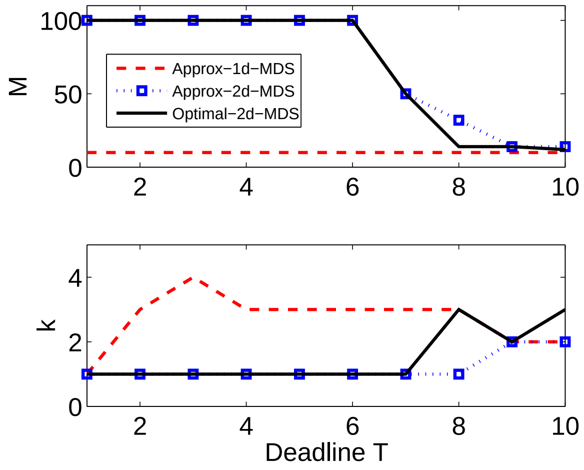


Figure 69. The approximated solution of k and M using one-dimensional MDS coding and two-dimensional MDS coding, and optimal solution of two-dimensional MDS coding.

$$\hat{k} = \lfloor \frac{k}{T}(\frac{\ln(\lambda T)}{\lambda} + s) + 0.5 \rfloor \tag{5.57}$$

And as for two-dimensional MDS coded computation, given the number of groups M, the corresponding $\hat{k}$ is

$$\hat{k} = \lfloor \frac{km}{MT}(\frac{\ln(\lambda T)}{\lambda} + s) + 0.5 \rfloor \tag{5.58}$$

Figure 71 presents the probability of success using one-dimensional MDS, two-dimensional MDS and uncoded computation schemes. the number of groups in two-dimensional MDS coding is $M = 11$. Similar to exponential distribution, it can be observed that coded computation outperforms the uncoded one in general.

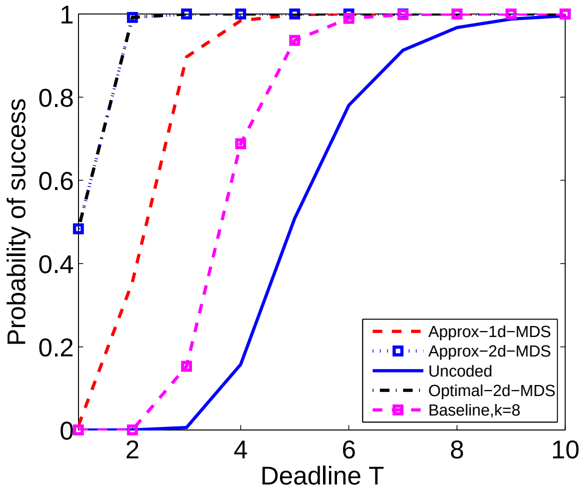


Figure 70. Probability of success using approximated solution to one-dimensional MDS coding and two-dimensional MDS coding, and optimal solution to two-dimensional MDS coding. The processing time follows exponential distribution with $\lambda = 1$.

Figure 72 presents the probability of success for different number of groups M in two-dimensional MDS coding. $k = 5$ for both one and two-dimensional MDS coding. Again, we observe similar result to the exponential distribution case.

Figure 73 presents the optimal and approximated solution to k and M as illustrated in Section 5.5.4 in order to maximize the probability of success. It can be observed the approximated solution is close the optimal one. Figure 74 shows the corresponding probability of success using the k and M shown in Figure 73. We also add two-dimensional MDS coded computation with baseline $k = 9$ and the uncoded computation in comparison. It can be observed that coded computation achieves higher probability of success than the uncoded case in general. In addition, our approximated solution for both one-dimensional and two-dimensional MDS coding achieve close performance to the optimal two-

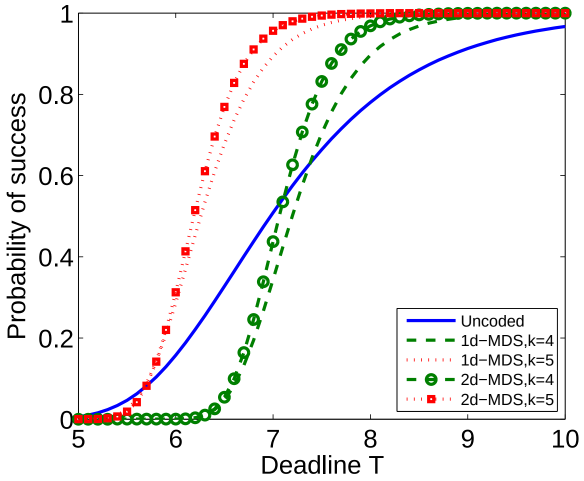


Figure 71. The probability of success using one-dimensional MDS, two-dimensional MDS and uncoded computation schemes. In two-dimensional MDS coded computation, the number of groups is $M = 11$. The processing time follows shifted exponential distribution with $\lambda = 1$ and $s = 2$.

dimensional MDS coded computation, with two-dimensional MDS coding slightly outperforming the one-dimensional one. Meanwhile, both of our approximated solutions outperform the baseline.

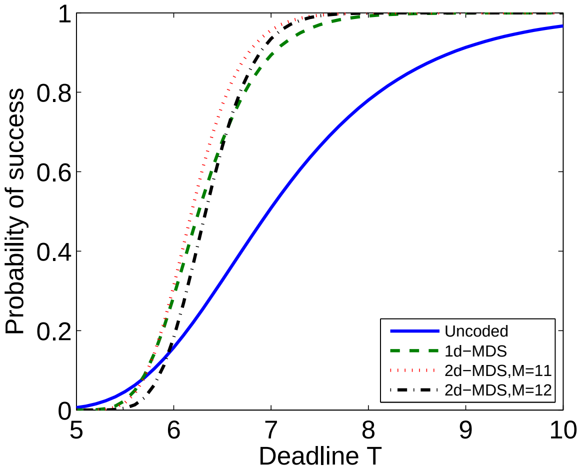


Figure 72. The probability of success using one-dimensional, two-dimensional and uncoded computation schemes. In both one-dimensional and two-dimensional MDS coded computation, $k = 5$. The processing time follows shifted exponential distribution with $\lambda = 1$ and $s = 2$.

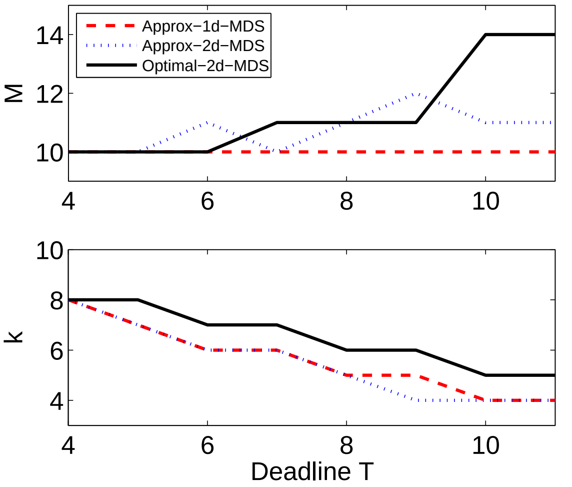


Figure 73. The approximated solution of k and M using one-dimensional MDS coding and two-dimensional MDS coding, and optimal solution of two-dimensional MDS coding.

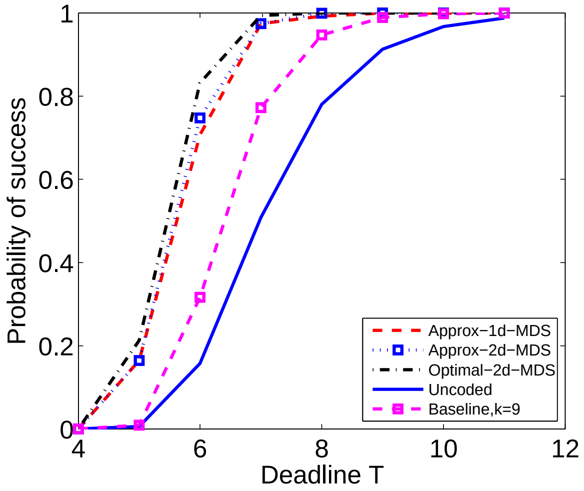


Figure 74. Probability of success using approximated solution to one-dimensional MDS coding and two-dimensional MDS coding, and optimal solution to two-dimensional MDS coding. The processing time follows shifted exponential distribution with $\lambda = 1$ and $s = 2$.

5.7 Conclusion

In this chapter, we consider a distributed computing system with hard deadlines and study the performance of (n, k) MDS codes in terms of the probability of master device meeting the deadline. We focus on matrix multiplication calculation and develop optimal and approximated solution to k in order to maximize the probability of success. Numerical results confirm our analysis and show that i) (n, k) MDS coded computation outperforms the uncoded computation in general and ii) our approximated solution of k in (n, k) MDS codes achieves near optimal performance while significantly reducing the calculation complexity.

CHAPTER 6

CONCLUSION

In this thesis, we focus on design and optimization for heterogeneous networks with emphasis on transportation, wireless and distributed computing networks. In particular,

1. We considered a transportation system of heterogeneously connected vehicles, where not all vehicles are able to communicate. We developed a connectivity-aware max-weight scheduling (CAMW) algorithm by taking into account the connectivity of vehicles. The crucial components of CAMW are expectation and learning components, which determine the estimated number of vehicles that can pass through the intersections by taking into account the heterogeneous communications. The simulations results show that CAMW algorithm significantly improves the intersection efficiency over max-weight.
2. We investigated the blocking problem which naturally arises in transportation networks, where multiple vehicles with different itineraries share available resources. We characterized waiting times at intersections of transportation systems by taking into account blocking probability as well as the heterogeneous communication probability of vehicles. Then, by using average waiting times at intersection, we developed a shortest delay algorithm that calculates the routes with shortest delays between two points in a transportation network. Our simulation results show that our shortest delay algorithm significantly improves over baselines that are unaware of the blocking problem.

3. We investigated the performance of heterogeneous (per-flow and FIFO) queues over wireless networks and characterized the support region of this system for arbitrary number of per-flow and FIFO queues and flows. We developed inner bound on the stability region, and developed resource allocation schemes; dFC and qFC, which achieve optimal operating point in the convex inner bound. Simulation results show that our algorithms significantly improve throughput in a wireless network with FIFO queues as compared to the well-known queue-based flow control and max-weight scheduling schemes.
4. We presented *Sneaker*, an in-network arbiter that provides differentiation between foreground and background flows, while enabling high performance and maintaining fairness within each traffic class. We have formulated the problem of per-flow prioritization using NUM framework and shown that *Sneaker* achieves the desired optimality. Further, we have extensively evaluated our design using both targeted small-scale simulations and realistic large-scale simulations.
5. We proposed *Legilimens*, an agile transport protocol for background applications in cellular networks. *Legilimens* quickly and accurately estimates capacity and busyness, and uses the estimates to achieve two conflicting objectives — quickly yield to foreground traffic and efficiently capture spare capacity. Using a *novel* algorithm to estimate capacity and load, *Legilimens* achieves the two objectives and outperforms known protocols, which achieve either one of the objectives but not both. As background applications become more reliant on cellular networks, schemes such as *Legilimens* are needed to enforce high-level objectives that accommodate the needs of users (foreground applications), content providers (background applications), and network operators (resource utilization).

6. We considered a distributed computing system with heterogeneous helper devices by taking into accoutn hard deadlines. We studied the performance of (n, k) MDS coded matrix-matrix multiplication in terms of the probability of master device meeting the deadline and developed optimal and approximated k in order to maximize this probability. Numerical results confirm our analysis and show that i) (n, k) MDS coded computation outperforms the uncoded computation in general and ii) our approximated solution of k in (n, k) MDS codes achieves near optimal performance while significantly reducing the calculation complexity.

APPENDIX

COPYRIGHT PERMISSIONS

This Appendix includes the copyright permissions granted from the IEEE and ACM to use published work in thesis.

Copyright Permission from IEEE: The following statement has been copied from the CopyRight Clearance Center (Rightslink).

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you must follow the requirements listed below:

Textual Material

Using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line ©2011 IEEE.

In the case of illustrations or tabular material, we require that the copyright line ©[Year of original publication] IEEE appear prominently with each reprinted figure and/or table.

If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Full-Text Article

If you are using the entire IEEE copyright owned article, the following IEEE copyright/ credit notice should be placed prominently in the references: ©[year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]

APPENDIX (Continued)

Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.

In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to `http://www.ieee.org/publications_standards/publications/rights/rights_link.html` to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Copyright Permission from ACM: The following statement has been copied from ACM Information for Authors (`https://authors.acm.org/main.html`).

Authors can include partial or complete papers of their own (and no fee is expected) in a dissertation as long as citations and DOI pointers to the Versions of Record in the ACM Digital Library are included. Authors can use any portion of their own work in presentations and in the classroom (and no fee is expected).

CITED LITERATURE

1. Zhou, S. and Seferoglu, H.: Connectivity-aware traffic phase scheduling for heterogeneously connected vehicles. In Proceedings of the First ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services, CarSys '16, pages 44–51, New York, NY, USA, 2016. ACM.

2. Zhou, S. and Seferoglu, H.: Blocking avoidance in transportation systems. In Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on, pages 83–90. IEEE, October 2015.

3. Zhou, S., Seferoglu, H., and Koyuncu, E.: Blocking avoidance in wireless networks. In Information Theory and Applications Workshop (ITA), 2016, pages 1–6. IEEE, February 2016.

4. Zhou, S., Chaudhry, M. U., Gopalakrishnan, V., Halepovic, E., Vamanan, B., and Seferoglu, H.: Managing background traffic in cellular networks. In 2019 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN). IEEE, 2019.

5. Schrank, D., Eisele, B., and Lomax, T.: Tti's 2012 urban mobility report. Texas A&M Transportation Institute. The Texas A&M University System, page 4, 2012.

6. Transportation energy databook, edition 26. Time Magazine. `http://cta.ornl.gov/data/new_for_edition26.shtml`.

7. Dresner, K. and Stone, P.: A multiagent approach to autonomous intersection management. Journal of artificial intelligence research, 31:591–656, 2008.

8. Papageorgiou, M., Diakaki, C., Dinopoulou, V., Kotsialos, A., and Wang, Y.: Review of road traffic control strategies. Proceedings of the IEEE, 91(12):2043–2067, 2003.

9. Gregoire, J., Frazzoli, E., De La Fortelle, A., and Wongpiromsarn, T.: Back-pressure traffic signal control with unknown routing rates. IFAC Proceedings Volumes, 47(3):11332–11337, 2014.

10. Aboudolas, K., Papageorgiou, M., and Kosmatopoulos, E.: Store-and-forward based methods for the signal control problem in large-scale congested urban road networks. Transportation Research Part C: Emerging Technologies, 17(2):163–174, 2009.

11. Cisco visual networking index: Global mobile data traffic forecast update. 2010 - 2015.

12. Ericsson mobility report. 2013.

13. Tassiulas, L. and Ephremides, A.: Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. IEEE transactions on automatic control, 37(12):1936–1948, 1992.

14. Tassiulas, L. and Ephremides, A.: Dynamic server allocation to parallel queues with randomly varying connectivity. IEEE Transactions on Information Theory, 39(2):466–478, 1993.

15. Neely, M. J., Modiano, E., and Li, C.: Fairness and optimal stochastic control for heterogeneous networks. IEEE/ACM Transactions On Networking, 16(2):396–409, 2008.

16. Cisco visual networking index: Global mobile data traffic forecast update, 2016?2021 white paper. 2016-2021.

17. Ericsson mobility report. June 2017.

18. Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I.: Spark: Cluster computing with working sets. HotCloud, 10(10-10):95, 2010.

19. Dean, J. and Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Communications of the ACM, 51(1):107–113, 2008.

20. Tassiulas, L. and Ephremides, A.: Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. IEEE transactions on automatic control, 37(12):1936–1948, 1992.

21. Varaiya, P.: Max pressure control of a network of signalized intersections. Transportation Research Part C: Emerging Technologies, 36:177–195, 2013.

22. Varaiya, P.: The max-pressure controller for arbitrary networks of signalized intersections. In Advances in Dynamic Network Modeling in Complex Transportation Systems, pages 27–66. Springer, 2013.

23. Wongpiromsarn, T., Uthaicharoenpong, T., Wang, Y., Frazzoli, E., and Wang, D.: Distributed traffic signal control for maximum network throughput. In Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on, pages 588–595. IEEE, 2012.

24. Miller, A.: Settings for fixed-cycle traffic signals. Journal of the Operational Research Society, 14(4):373–386, 1963.

25. Gartner, N.: OPAC: A demand-responsive strategy for traffic signal control. Number 906. 1983.

26. Cascetta, E., Gallo, M., and Montella, B.: Models and algorithms for the optimization of signal settings on urban networks with stochastic assignment models. Annals of Operations Research, 144(1):301–328, 2006.

27. Mirchandani, P. B. and Zou, N.: Queuing models for analysis of traffic adaptive signal control. IEEE Transactions on Intelligent Transportation Systems, 8(1):50–59, 2007.

28. Diakaki, C., Papageorgiou, M., and Aboudolas, K.: A multivariable regulator approach to traffic-responsive network-wide signal control. Control Engineering Practice, 10(2):183–195, 2002.

29. Henry, J., Farges, J., and Tuffal, J.: The prodyn real time traffic algorithm. In Control in Transportation Systems, pages 305–310. Elsevier, 1984.

30. Hunt, P., Robertson, D., Bretherton, R., and Royle, M. C.: The scoot on-line traffic signal optimisation technique. Traffic Engineering & Control, 23(4), 1982.

31. Lowrie, P.: Scats, sydney co-ordinated adaptive traffic system: A traffic responsive method of controlling urban traffic. 1990.

32. Mirchandani, P. and Head, L.: A real-time traffic signal control system: architecture, algorithms, and analysis. Transportation Research Part C: Emerging Technologies, 9(6):415–432, 2001.

33. Webster, F. V.: Traffic signal settings. Road Res. Lab., Ministry Transport, HMSO, London, U.K., pages 1–43, 1958.

34. Newell, G. F.: Approximation methods for queues with application to the fixed-cycle traffic light. Siam Review, 7(2):223–240, 1965.

35. Heidemann, D.: Queue length and delay distributions at traffic signals. Transportation Research Part B: Methodological, 28(5):377–389, 1994.

36. Osorio, C. and Bierlaire, M.: A surrogate model for traffic optimization of congested networks: an analytic queueing network approach. Technical report, 2009.

37. Lo, H. K., Chang, E., and Chan, Y. C.: Dynamic network traffic control. Transportation Research Part A: Policy and Practice, 35(8):721–744, 2001.

38. Di Febbraro, A., Giglio, D., and Sacco, N.: On applying petri nets to determine optimal offsets for coordinated traffic light timings. In Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on, pages 773–778. IEEE, 2002.

39. Neely, M. J.: Dynamic power allocation and routing for satellite and wireless networks with time varying channels. Doctoral dissertation, Massachusetts Institute of Technology, PhD thesis, LIDS, 2003.

40. Neely, M. J., Modiano, E., and Rohrs, C. E.: Dynamic power allocation and routing for time-varying wireless networks. IEEE Journal on Selected Areas in Communications, 23(1):89–103, 2005.

41. Varaiya, P.: A universal feedback control policy for arbitrary networks of signalized intersections. Published online. URL: http://paleale. eecs. berkeley. edu/˜ varaiya/papers_ps. dir/090801-Intersectionsv5. pdf, 2009.

42. Gregoire, J., Qian, X., Frazzoli, E., De La Fortelle, A., and Wongpiromsarn, T.: Capacity-aware backpressure traffic signal control. IEEE Transactions on Control of Network Systems, 2(2):164–173, 2015.

43. Lu, Y. and Yang, X.: Estimating dynamic queue distribution in a signalized network through a probability generating model. IEEE Transactions on Intelligent Transportation Systems, 15(1):334–344, 2014.

44. Kimber, R. and Hollis, E.: Peak-period traffic delays at road junctions and other bottlenecks. Traffic Engineering & Control, 19(N10), 1978.

45. Akçelik, R.: Time-dependent expressions for delay, stop rate and queue length at traffic signals. 1980.

46. Akçelik, R. and Rouphail, N. M.: Estimation of delays at traffic signals for variable demand conditions. Transportation Research Part B: Methodological, 27(2):109–131, 1993.

47. Comert, G. and Cetin, M.: Analytical evaluation of the error in queue length estimation at traffic signals from probe vehicle data. IEEE Transactions on Intelligent Transportation Systems, 12(2):563–573, 2011.

48. Ndoye, M., Totten, V. F., Krogmeier, J. V., and Bullock, D. M.: Sensing and signal processing for vehicle reidentification and travel time estimation. IEEE Transactions on Intelligent Transportation Systems, 12(1):119–131, 2011.

49. Zhang, G. and Wang, Y.: Optimizing minimum and maximum green time settings for traffic actuated control at isolated intersections. IEEE Transactions on Intelligent Transportation Systems, 12(1):164–173, 2011.

50. Astuti, D.: Packet handling. In Seminar on Transport of Multimedia Streams in Wireless Internet, Helsinki, Finland, volume 10, 2003.

51. Floyd, S. and Fall, K.: Router mechanisms to support end-to-end congestion control. Technical report, Citeseer, 1997.

52. http://madwifi-project.org/wiki/Chipsets.

53. http://www.linuxjournal.com/content/queueing-linux-network-stack?page=0,2.

54. Karol, M., Hluchyj, M., and Morgan, S.: Input versus output queueing on a space-division packet switch. IEEE Transactions on communications, 35(12):1347–1356, 1987.

55. Neely, M. J.: Stochastic network optimization with application to communication and queueing systems. Synthesis Lectures on Communication Networks, 3(1):1–211, 2010.

56. Tassiulas, L.: Scheduling and performance limits of networks with constantly changing topology. IEEE Transactions on Information Theory, 43(3):1067–1073, 1997.

57. Kahale, N. and Wright, P. E.: Dynamic global packet routing in wireless networks. In INFOCOM'97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution., Proceedings IEEE, volume 3, pages 1414–1421. IEEE, 1997.

58. Andrews, M., Kumaran, K., Ramanan, K., Stolyar, A., Whiting, P., and Vijayakumar, R.: Providing quality of service over a shared wireless link. IEEE Communications magazine, 39(2):150–154, 2001.

59. Neely, M., Modiano, E., and Rohrs, C. E.: Dynamic power allocation and routing for time-varying wireless networks. IEEE Journal on Selected Areas in Communications, 23(1):89–103, 2005.

60. Stolyar, A. L.: Greedy primal-dual algorithm for dynamic resource allocation in complex networks. Queueing Systems, 54(3):203–220, 2006.

61. Liu, J., Stolyar, A. L., Chiang, M., and Poor, H. V.: Queue back-pressure random access in multihop wireless networks: optimality and stability. IEEE Transactions on Information Theory, 55(9):4087–4098, 2009.

62. Radunović, B., Gkantsidis, C., Gunawardena, D., and Key, P.: Horizon: Balancing tcp over multiple paths in wireless mesh network. In Proceedings of the 14th ACM international conference on Mobile computing and networking, pages 247–258. ACM, 2008.

63. Warrier, A., Janakiraman, S., Ha, S., and Rhee, I.: Diffq: Practical differential backlog congestion control for wireless networks. In INFOCOM 2009, IEEE, pages 262–270. IEEE, 2009.

64. Akyol, U., Andrews, M., Gupta, P., Hobby, J., Saniee, I., and Stolyar, A.: Joint scheduling and congestion control in mobile ad-hoc networks. In INFOCOM 2008. The 27th Conference on Computer Communications. IEEE, pages 619–627. IEEE, 2008.

65. Sridharan, A., Moeller, S., and Krishnamachari, B.: Making distributed rate control using lyapunov drifts a reality in wireless sensor networks. In Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops, 2008. WiOPT 2008. 6th International Symposium on, pages 452–461. IEEE, 2008.

66. Moeller, S., Sridharan, A., Krishnamachari, B., and Gnawali, O.: Routing without routes: The backpressure collection protocol. In Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, pages 279–290. ACM, 2010.

67. Laufer, R., Salonidis, T., Lundgren, H., and Le Guyadec, P.: Xpress: A cross-layer backpressure architecture for wireless multi-hop networks. In Proceedings of the 17th annual international conference on Mobile computing and networking, pages 49–60. ACM, 2011.

68. Athanasopoulou, E., Bui, L. X., Ji, T., Srikant, R., and Stolyar, A.: Back-pressure-based packet-by-packet adaptive routing in communication networks. IEEE/ACM transactions on networking, 21(1):244–257, 2013.

69. Bui, L. X., Srikant, R., and Stolyar, A.: A novel architecture for reduction of delay and queueing structure complexity in the back-pressure algorithm. IEEE/ACM Transactions on Networking (TON), 19(6):1597–1609, 2011.

70. Li, B. and Srikant, R.: Queue-proportional rate allocation with per-link information in multihop networks. In ACM SIGMETRICS Performance Evaluation Review, volume 43, pages 97–108. ACM, 2015.

71. Ji, B., Joo, C., , and Shroff, N. B.: Throughput-optimal scheduling in multihop wireless networks without per-flow information. IEEE/ACM Transactions on Networking (TON), 21(2):634–647, 2013.

72. Walton, N. S.: Concave switching in single and multihop networks. In ACM SIGMETRICS Performance Evaluation Review, volume 42, pages 139–151. ACM, 2014.

73. Bramson, M., D'Auria, B., and Walton, N.: Proportional switching in fifo networks. arXiv preprint arXiv:1412.4390, 2014.

74. Seferoglu, H. and Modiano, E.: Diff-max: Separation of routing and scheduling in backpressure-based wireless networks. In INFOCOM, 2013 Proceedings IEEE, pages 1555–1563. IEEE, 2013.

75. Kuzmanovic, A. and Knightly, E.: Tcp-lp: low-priority service via end-point congestion control. IEEE/ACM Transactions on Networking (TON), 14(4):739–752, 2006.

76. Floyd, S. and Jacobson, V.: Random early detection gateways for congestion avoidance. IEEE/ACM ToN, (4):397–413, 1993.

77. Floyd, S. and Jacobson, V.: Random early detection gateways for congestion avoidance. IEEE/ACM ToN, 1(4), 1993.

78. Ott, T. J., Lakshman, T., and Wong, L. H.: SRED: stabilized RED. In IEEE INFOCOM'99, volume 3, pages 1346–1355, 1999.

79. Kunniyur, S. and Srikant, R.: Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management. In ACM SIGCOMM Comp. Comm. Review, volume 31. ACM, 2001.

80. Feng, W.-c., Shin, K. G., Kandlur, D. D., and Saha, D.: The BLUE active queue management algorithms. IEEE/ACM Transactions on Networking (ToN), 10(4):513–528, 2002.

81. Pan, R., Prabhakar, B., and Psounis, K.: CHOKe – A stateless active queue management scheme for approximating fair bandwidth allocation. In IEEE INFOCOM 2000, volume 2, pages 942–951, 2000.

82. Wang, J., Huang, A., Wang, W., Zhang, Z., and Lau, V.: On the transmission opportunity and tcp throughput in cognitive radio networks. Int. J. Commun. Syst., 27(2):303–321, May 2012.

83. Gurtov, A. and Ludwig, R.: Responding to spurious timeouts in tcp. In Proc. of IEEE INFOCOM, volume 3, pages 2312–2322, 2003.

84. Ludwig, R. and Katz, R.: The eifel algorithm: Making tcp robust against spurious retransmissions. ACM Computer Communication Review, 30:30–36, Janurary 2000.

85. Liu, X., Sridharan, A., Machiraju, S., Seshadri, M., and Zang, H.: Experiences in a 3g network: Interplay between the wireless channel and applications. In Proc. of the 14th ACM international conference on Mobile computing and networking, pages 211–222. ACM, 2008.

86. Lu, F., Du, H., Jain, A., Voelker, G., Snoeren†, A., and Terzis, A.: Cqic: Revisiting cross-layer congestion control for cellular networks. In Proc. of the 16th International Workshop on Mobile Computing Systems and Applications, pages 45–50. ACM, February 2015.

87. Klein, T., Leung, K., and Zheng, H.: Enhanced scheduling algorithms for improved tcp performance in wireless ip networks. In Proc. of IEEE Globecom, volume 4, pages 2744–2759, 2004.

88. Rossi, D., Testa, C., Valenti, S., and Muscariello, L.: Ledbat: The new bittorrent congestion control protocol. In ICCCN, pages 1–6, 2010.

89. Venkataramani, A., Kokku, R., and Dahlin, M.: Tcp nice: A mechanism for background transfers. ACM SIGOPS Operating Systems Review, 36(SI):329–343, 2002.

90. Holfeld, B., Wieruch, D., Wirth, T., Thiele, L., Ashraf, S., Huschke, J., Aktas, I., and Ansari, J.: Wireless communication for factory automation: an opportunity for lte and 5g systems. IEEE Communications Magazine, 54(6):36–43, 2016.

91. Winstein, K., Sivaraman, A., and Balakrishnan, H.: Stochastic forecasts achieve high throughput and low delay over cellular networks. In Proceedings of NSDI, pages 459–471, 2013.

92. Capozzi, F., Piro, G., Grieco, L. A., Boggia, G., and Camarda, P.: Downlink packet scheduling in LTE cellular networks: Key design issues and a survey. IEEE Communications Surveys & Tutorials, 15(2):678–700, 2013.

93. Padhye, J., Firoiu, V., Towsley, D., and Kurose, J.: Modeling tcp reno performance: a simple model and its empirical validation. IEEE/ACM Transactions on Networking (ToN), 8(2):133–145, 2000.

94. Low, S. H.: A duality model of TCP and queue management algorithms. IEEE/ACM ToN, 11(4):525–536, 2003.

95. Kuhn, H. and Tucker, A.: Nonlinear programming. In Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability, pages 481–492, 1951.

96. Liu, E., Zhang, Q., and Leung, K. K.: Asymptotic analysis of proportionally fair scheduling in rayleigh fading. IEEE Transactions on Wireless Communications, 10(6):1764–1775, 2011.

97. Srikant, R. and Ying, L.: Communication networks: an optimization, control, and stochastic networks perspective. Cambridge University Press, 2013.

98. NS-3 network simulator. `http://www.nsnam.org/`.

99. Kuusela, P., Lassila, P., Virtamo, J., and Key, P.: Modeling RED with idealized TCP sources. Proceedings of IFIP ATM & IP, 2001.

100. Andrade, C. E., Byers, S. D., Gopalakrishnan, V., Halepovic, E., Majmundar, M., Poole, D. J., Tran, L. K., and Volinsky, C. T.: ”managing massive firmware-over-the-air updates for connected cars in cellular networks”. In Proceedings of the 2nd ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services, CarSys ’17, pages 65–72, New York, NY, USA, 2017. ACM.

101. Zhang, Y., Arvidsson, Å., Siekkinen, M., and Urvoy-Keller, G.: Understanding HTTP flow rates in cellular networks. In 2014 IFIP Networking Conference, pages 1–8, June 2014.

102. Fall, K. and Floyd, S.: Simulation-based comparisons of tahoe, reno and sack tcp. SIGCOMM Comput. Commun. Rev., 26(3):5–21, July 1996.

103. Henderson, T., Floyd, S., Gurtov, A., and Nishida, Y.: ”the newreno modification to tcp's fast recovery algorithm”. RFC 6582, RFC Editor, April 2012.

104. Jacobson, V.: ”congestion avoidance and control”. In Symposium Proceedings on Communications Architectures and Protocols, SIGCOMM '88, pages 314–329, New York, NY, USA, 1988. ACM.

105. Rhee, I., Xu, L., Ha, S., Zimmermann, A., Eggert, L., and Scheffenegger, R.: ”cubic for fast long-distance networks”. RFC 8312, RFC Editor, February 2018.

106. Jiang, H., Wang, Y., Lee, K., and Rhee, I.: ”tackling bufferbloat in 3g/4g networks”. In Proceedings of the 2012 Internet Measurement Conference, IMC '12, pages 329–342, New York, NY, USA, 2012. ACM.

107. Brakmo, L. S. and Peterson, L. L.: TCP Vegas: End to end congestion avoidance on a global Internet. IEEE Journal on Selected Areas in Communications, 13(8):1465–1480, 1995.

108. Kuzmanovic, A. and Knightly, E. W.: ”tcp-lp: A distributed algorithm for low priority data transfer”. In INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, volume 3, pages 1691–1701. IEEE, 2003.

109. Shalunov, S., Hazel, G., Iyengar, J., and Kuehlewind, M.: ”low extra delay background transport (ledbat)”. RFC 6817, RFC Editor, December 2012.

110. Venkataramani, A., Kokku, R., and Dahlin, M.: ”tcp nice: A mechanism for background transfers”. ACM SIGOPS Operating Systems Review, 36(SI):329–343, 2002.

111. Winstein, K., Sivaraman, A., and Balakrishnan, H.: ”stochastic forecasts achieve high throughput and low delay over cellular networks”. In Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation, nsdi'13, pages 459–472, Berkeley, CA, USA, 2013. USENIX Association.

112. Zaki, Y., Pötsch, T., Chen, J., Subramanian, L., and Görg, C.: "adaptive congestion control for unpredictable cellular networks". In Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM '15, pages 509–522, New York, NY, USA, 2015. ACM.

113. Cardwell, N., Cheng, Y., Gunn, C. S., Yeganeh, S. H., and Jacobson, V.: "bbr: Congestion-based congestion control". Queue, 14(5):50, 2016.

114. Leong, W. K., Wang, Z., and Leong, B.: "tcp congestion control beyond bandwidth-delay product for mobile cellular networks". In Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies, CoNEXT '17, pages 167–179, New York, NY, USA, 2017. ACM.

115. Chakraborty, A., Navda, V., Padmanabhan, V. N., and Ramjee, R.: Coordinating cellular background transfers using Loadsense. In Proceedings of the 19th annual international conference on Mobile computing & networking, pages 63–74. ACM, 2013.

116. Keshav, S.: "Congestion Control in Computer Networks". Doctoral dissertation, EECS Department, University of California, Berkeley, Sep 1991.

117. Morton, A. and den Berghe, S. V.: "framework for metric composition". RFC 5835, RFC Editor, April 2010.

118. Prasad, R., Dovrolis, C., Murray, M., and Claffy, K.: Bandwidth estimation: metrics, measurement techniques, and tools. IEEE network, 17(6):27–35, 2003.

119. Dovrolis, C., Ramanathan, P., and Moore, D.: Packet-dispersion techniques and a capacity-estimation methodology. IEEE/ACM Transactions on Networking, 12(6):963–977, Dec 2004.

120. Hu, N., Li, L. E., Mao, Z. M., Steenkiste, P., and Wang, J.: Locating Internet Bottlenecks: Algorithms, Measurements, and Implications. In Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM '04, pages 41–54, New York, NY, USA, 2004. ACM.

121. Baranasuriya, N., Navda, V., Padmanabhan, V. N., and Gilbert, S.: "qprobe: Locating the bottleneck in cellular communication". In Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies, CoNEXT '15, pages 33:1–33:7, New York, NY, USA, 2015. ACM.

122. Park, S., Lee, J., Kim, J., Lee, J., Ha, S., and Lee, K.: ”exll: An extremely low-latency congestion control for mobile cellular networks”. In Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies, CoNEXT '18, pages 307–319, New York, NY, USA, 2018. ACM.

123. Srikant, R. and Ying, L.: ”Communication networks: an optimization, control, and stochastic networks perspective”. Cambridge University Press, 2013.

124. Carofiglio, G., Muscariello, L., Rossi, D., and Testa, C.: A hands-on assessment of transport protocols with lower than best effort priority. In Local Computer Networks (LCN), 2010 IEEE 35th Conference on, pages 8–15. IEEE, 2010.

125. Jacobson, V., Braden, B., and Borman, D.: ”tcp extensions for high performance”. RFC 1323, RFC Editor, May 1992.

126. Murray, D. and Koziniec, T.: The state of enterprise network traffic in 2012. In 2012 18th Asia-Pacific Conference on Communications (APCC), pages 179–184. IEEE, 2012.

127. Halepovic, E., Pang, J., and Spatscheck, O.: ”can you get me now?: Estimating the time-to-first-byte of http transactions with passive measurements”. In Proceedings of the 2012 Internet Measurement Conference, IMC '12, pages 115–122, New York, NY, USA, 2012. ACM.

128. Braden, R.: ”requirements for internet hosts - communication layers”. STD 3, RFC Editor, October 1989.

129. Xu, X., Jiang, Y., Flach, T., Katz-Bassett, E., Choffnes, D. R., and Govindan, R.: ”investigating transparent web proxies in cellular networks”. In PAM, 2015.

130. Banerjee, A., Cho, J., Eide, E., Duerig, J., Nguyen, B., Ricci, R., Van der Merwe, J., Webb, K., and Wong, G.: ”phantomnet: Research infrastructure for mobile networking, cloud computing and software-defined networking”. GetMobile: Mobile Computing and Communications, 19(2):28–33, 2015.

131. Huang, J., Qian, F., Guo, Y., Zhou, Y., Xu, Q., Mao, Z. M., Sen, S., and Spatscheck, O.: ”an in-depth study of lte: Effect of network protocol and application behavior on performance”. In Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, SIGCOMM '13, pages 363–374, New York, NY, USA, 2013. ACM.

132. Claypool, M., Chung, J. W., and Li, F.: BBR: An implementation of bottleneck bandwidth and round-trip time congestion control for ns-3”. In Proceedings of the 10th Workshop on Ns-3, WNS3 ’18, pages 1–8, New York, NY, USA, 2018. ACM.

133. Claypool, M.: An implementation of bottleneck bandwidth and round-trip time congestion control for ns-3. `https://github.com/mark-claypool/bbr`.

134. Dean, J. and Barroso, L. A.: The tail at scale. Communications of the ACM, 56(2):74–80, 2013.

135. Ananthanarayanan, G., Kandula, S., Greenberg, A. G., Stoica, I., Lu, Y., Saha, B., and Harris, E.: Reining in the outliers in map-reduce clusters using mantri. In Osdi, volume 10, page 24, 2010.

136. Dimakis, A. G., Godfrey, P. B., Wu, Y., Wainwright, M. J., and Ramchandran, K.: Network coding for distributed storage systems. IEEE transactions on information theory, 56(9):4539–4551, 2010.

137. Rashmi, K. V., Shah, N. B., and Kumar, P. V.: Optimal exact-regenerating codes for distributed storage at the msr and mbr points via a product-matrix construction. IEEE Transactions on Information Theory, 57(8):5227–5239, 2011.

138. Papailiopoulos, D. S., Dimakis, A. G., and Cadambe, V. R.: Repair optimal erasure codes through hadamard designs. IEEE Transactions on Information Theory, 59(5):3021–3037, 2013.

139. Gopalan, P., Huang, C., Simitci, H., and Yekhanin, S.: On the locality of codeword symbols. IEEE Transactions on Information Theory, 58(11):6925–6934, 2012.

140. Rawat, A. S., Koyluoglu, O. O., Silberstein, N., and Vishwanath, S.: Optimal locally repairable and secure codes for distributed storage systems. IEEE Transactions on Information Theory, 60(1):212–236, 2014.

141. Oggier, F. and Datta, A.: Self-repairing homomorphic codes for distributed storage systems. In 2011 Proceedings IEEE INFOCOM, pages 1215–1223. IEEE, 2011.

142. Dutta, S., Cadambe, V., and Grover, P.: Short-dot: Computing large linear transforms distributedly using coded short dot products. In Advances In Neural Information Processing Systems, pages 2100–2108, 2016.

143. Tandon, R., Lei, Q., Dimakis, A. G., and Karampatziakis, N.: Gradient coding: Avoiding stragglers in distributed learning. In International Conference on Machine Learning, pages 3368–3376, 2017.

144. Lee, K., Suh, C., and Ramchandran, K.: High-dimensional coded matrix multiplication. In 2017 IEEE International Symposium on Information Theory (ISIT), pages 2418–2422. IEEE, 2017.

145. Yu, Q., Maddah-Ali, M., and Avestimehr, S.: Polynomial codes: an optimal design for high-dimensional coded matrix multiplication. In Advances in Neural Information Processing Systems, pages 4403–4413, 2017.

146. Dutta, S., Fahim, M., Haddadpour, F., Jeong, H., Cadambe, V., and Grover, P.: On the optimal recovery threshold of coded matrix multiplication. arXiv preprint arXiv:1801.10292, 2018.

147. Lee, K., Lam, M., Pedarsani, R., Papailiopoulos, D., and Ramchandran, K.: Speeding up distributed machine learning using codes. IEEE Transactions on Information Theory, 64(3):1514–1529, 2018.

148. Lee, K., Shah, N. B., Huang, L., and Ramchandran, K.: The mds queue: Analysing the latency performance of erasure codes. IEEE Transactions on Information Theory, 63(5):2822–2842, 2017.

149. Joshi, G., Liu, Y., and Soljanin, E.: On the delay-storage trade-off in content download from coded distributed storage systems. IEEE Journal on Selected Areas in Communications, 32(5):989–997, 2014.

150. Kadhe, S., Soljanin, E., and Sprintson, A.: When do the availability codes make the stored data more available? In 2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton), pages 956–963. IEEE, 2015.

151. Xing, Y. and Seferoglu, H.: Predictive edge computing with hard deadlines. In 2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), pages 13–18. IEEE, 2018.

152. Dutta, S., Cadambe, V., and Grover, P.: Coded convolution for parallel and distributed computing within a deadline. In 2017 IEEE International Symposium on Information Theory (ISIT), pages 2403–2407, June 2017.

153. Ostovari, P., Khreishah, A., and Wu, J.: Broadcasting with hard deadlines in wireless multihop networks using network coding. Wireless Communications and Mobile Computing, 15(5):983–999, 2015.

154. Raginsky, M., Sason, I., et al.: Concentration of measure inequalities in information theory, communications, and coding. Foundations and Trends® in Communications and Information Theory, 10(1-2):1–246, 2013.

155. Hoeffding's inequality. `https://en.wikipedia.org/wiki/Hoeffding\%27s\_\ \inequality`. Accessed: 2019-06-17.

156. Stolyar, A.: On the asymptotic optimality of the gradient scheduling algorithm for multiuser throughput allocation. Operations research, 53(1):12–25, 2005.

157. Agrawal, R. and Subramanian, V.: Optimality of certain channel aware scheduling policies. In Proceedings of the Annual Allerton Conference on Communication Control and Computing, volume 40, pages 1533–1542. The University; 1998, 2002.

158. Huang, J., Qian, F., Guo, Y., Zhou, Y., Xu, Q., Mao, Z., Sen, S., and Spatscheck, O.: An in-depth study of lte: Effect of network protocol and application behavior on performance. ACM SIGCOMM Computer Communication Review, 43(4):363–374, October 2013.

159. Sulthana, S. and Nakkeeran, R.: Study of downlink scheduling algorithms in lte networks. JNW, 9(12):3381–3391, 2014.

160. Seferoglu, H. and Modiano, E.: Tcp-aware backpressure routing and scheduling. IEEE Transactions on Mobile Computing, 15(7):1783–1796, 2016.

161. Aggarwal, V., Jana, R., Pang, J., Ramakrishnan, K., and Shankaranarayanan, N.: Characterizing fairness for 3g wireless networks. In Local & Metropolitan Area Networks (LANMAN), 2011 18th IEEE Workshop on, pages 1–6. IEEE, 2011.

162. Liu, K. and Lee, J.: On improving tcp performance over mobile data networks. IEEE Transactions on Mobile Computing, 15(10):2522–2536, 2016.

163. Wu, E. and Chen, M.: Jtcp: Jitter-based tcp for heterogeneous wireless networks. IEEE Journal on Selected Areas in Communications, 22(4):757–766, 2004.

164. Ren, F. and Lin, C.: Modeling and improving tcp performance over cellular link with variable bandwidth. IEEE Transactions on Mobile Computing, 10(8):1057–1070, 2011.

165. Assaad, M. and Zeghlache, D.: Cross-layer design in hsdpa system to reduce the tcp effect. IEEE Journal on selected areas in communications, 24(3):614–625, 2006.

166. Brakmo, L. and Peterson, L.: Tcp vegas: End to end congestion avoidance on a global internet. IEEE Journal on selected Areas in communications, 13(8):1465–1480, 1995.

167. Hayes, D. and Armitage, G.: Revisiting tcp congestion control using delay gradients. In International Conference on Research in Networking, pages 328–341. Springer, 2011.

168. Nagaraj, K., Bharadia, D., Mao, H., Chinchali, S., Alizadeh, M., and Katti, S.: Numfabric: Fast and flexible bandwidth allocation in datacenters. In Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference, pages 188–201. ACM, 2016.

169. Shah, S., Chen, K., and Nahrstedt, K.: Available bandwidth estimation in ieee 802.11-based wireless networks. In Proceedings of 1st ISMA/CAIDA Workshop on Bandwidth Estimation (BEst), 2003.

170. Lakshminarayanan, K., Padmanabhan, V., and Padhye, J.: Bandwidth estimation in broadband access networks. In Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, pages 314–321. ACM, 2004.

171. Lee, H., Hall, V., Yum, K., Kim, K., and Kim, E.: Bandwidth estimation in wireless lans for multimedia streaming services. Advances in Multimedia, 2007, 2007.

172. Kapoor, R., Chen, L., Lao, L., Gerla, M., and Sanadidi, M.: Capprobe: A simple and accurate capacity estimation technique. ACM SIGCOMM Computer Communication Review, 34(4):67–78, 2004.

173. Johnsson, A., Melander, B., and Björkman, M.: Bandwidth measurement in wireless networks. In Challenges in Ad Hoc Networking, pages 89–98. Springer, 2006.

174. Li, M., Claypool, M., and Kinicki, R.: Wbest: A bandwidth estimation tool for ieee 802.11 wireless networks. In Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on, pages 374–381. IEEE, 2008.

175. Jacobson, V.: Congestion avoidance and control. In ACM SIGCOMM computer communication review, volume 18, pages 314–329. ACM, 1988.

176. Keshav, S.: A control-theoretic approach to flow control, volume 21. ACM, 1991.

177. Bolot, J.: End-to-end packet delay and loss behavior in the internet. In ACM SIGCOMM Computer Communication Review, volume 23, pages 289–298. ACM, 1993.

178. Dovrolis, C., Ramanathan, P., and Moore, D.: Packet-dispersion techniques and a capacity-estimation methodology. IEEE/ACM Transactions On Networking, 12(6):963–977, 2004.

179. Yang, T., Jin, Y., Chen, Y., and Jin, Y.: Rt-wabest: A novel end-to-end bandwidth estimation tool in ieee 802.11 wireless network. International Journal of Distributed Sensor Networks, 13(2), 2017.

180. Bodrog, L., Horváth, G., and Vulkán, C.: Analytical tcp throughput model for high-speed downlink packet access. IET software, 3(6):480–494, 2009.

181. Padhye, J., Firoiu, V., Towsley, D., and Kurose, J.: Cyber-physical systems: the next computing revolution. In Proceedings of ACM SIGCOMM. ACM, 1998.

182. S., L.: A duality model of tcp and queue management algorithms. IEEE/ACM Transactions on Networking, pages 525–536, 2002.

183. Regional transportation performance measurement. `http://www.cmap.illinois.gov/mobility/roads/cmp/performance-measurement`. Accessed: 2016-03-25.

184. Rajkumar, R. R., Lee, I., Sha, L., and Stankovic, J.: Cyber-physical systems: the next computing revolution. In Proceedings of the 47th design automation conference, pages 731–736. ACM, 2010.

185. Huang, J., Qian, F., Guo, Y., Zhou, Y., Xu, Q., Mao, Z. M., Sen, S., and Spatscheck, O.: An in-depth study of lte: effect of network protocol and application behavior on performance. ACM SIGCOMM Computer Communication Review, 43(4):363–374, 2013.

186. Klein, T., Leung, K., and Zheng, H.: Enhanced scheduling algorithms for improved tcp performance in wireless ip networks. In IEEE Globecom, volume 4, pages 2744–2759, 2004.

187. Keller, L., Le, A., Cici, B., Seferoglu, H., Fragouli, C., and Markopoulou, A.: Microcast: Cooperative video streaming on smartphones. In Proceedings of the 10th international conference on Mobile systems, applications, and services, pages 57–70. ACM, 2012.

188. Hahne, E. L.: Round-robin scheduling for max-min fairness in data networks. IEEE Journal on Selected Areas in communications, 9(7):1024–1039, 1991.

189. Ekstrom, H.: QoS control in the 3gpp evolved packet system. IEEE Communications Magazine, 47(2):76–83, February 2009.

190. Floyd, S., Gummadi, R., Shenker, S., et al.: Adaptive RED: An algorithm for increasing the robustness of RED's active queue management, 2001.

191. Xu, Y.-D., Wang, Z.-Y., and Wang, H.: ARED: a novel adaptive congestion controller. In Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on, volume 2, pages 708–714. IEEE, 2005.

192. Athuraliya, S., Low, S. H., Li, V. H., and Yin, Q.: REM: Active queue management. IEEE network, 15(3):48–53, 2001.

193. Nichols, K. and Jacobson, V.: Controlling queue delay. Communications of the ACM, 55(7):42–50, 2012.

194. Ismail, A. H., El-Sayed, A., Elsaghir, Z., and Morsi, I. Z.: Enhanced random early detection (ENRED). International Journal of Computer Applications, 92(9), 2014.

195. Baklizi, M., Abdel-jaber, H., Ramadass, S., Abdullah, N., and Anbar, M.: Performance assessment of AGRED, RED and GRED congestion control algorithms. Information Technology Journal, 11(2):255, 2012.

196. Baklizi, M., Abdel-Jaber, H., Abu-Alhaj, M. M., Abdullah, N., Ramadass, S., and ALmomani, A.: Dynamic stochastic early discovery: a new congestion control technique to improve networks performance. ICIC Int, 9(3):1113–1126, 2013.

197. Floyd, S.: Recommendations on using the gentle variant of RED. http://www. aciri. org/floyd/red/gentle. html, 2000.

198. Zhao, W., Ramamritham, K., and Stankovic, J. A.: Preemptive scheduling under time and resource constraints. IEEE Transactions on computers, 100(8):949–960, 1987.

199. Tayyar, H. F. and Alnuweiri, H.: Weighted fair queuing scheduler, March 20 2007. US Patent 7,194,741.

200. Monghal, G., Pedersen, K. I., Kovacs, I. Z., and Mogensen, P. E.: QoS oriented time and frequency domain packet schedulers for the UTRAN long term evolution. In Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE, pages 2532–2536. IEEE, 2008.

201. Zaki, Y., Weerawardane, T., Gorg, C., and Timm-Giel, A.: Multi-QoS-aware fair scheduling for LTE. In Vehicular technology conference (VTC spring), 2011 IEEE 73rd, pages 1–5. IEEE, 2011.

202. Skoutas, D. N. and Rouskas, A. N.: Scheduling with QoS provisioning in mobile broadband wireless systems. In Wireless Conference (EW), 2010 European, pages 422–428. IEEE, 2010.

203. Sandrasegaran, K., Ramli, H. A. M., and Basukala, R.: Delay-prioritized scheduling (DPS) for real time traffic in 3GPP LTE system. In Wireless Communications and Networking Conference (WCNC), 2010 IEEE, pages 1–6. IEEE, 2010.

204. Monghal, G., Laselva, D., Michaelsen, P.-H., and Wigard, J.: Dynamic packet scheduling for traffic mixes of best effort and VoIP users in E-UTRAN downlink. In Vehicular Technology Conference (VTC 2010-Spring), 2010 IEEE 71st, pages 1–5. IEEE, 2010.

205. Choi, S., Jun, K., Shin, Y., Kang, S., and Choi, B.: MAC scheduling scheme for VoIP traffic service in 3G LTE. In Vehicular Technology Conference, 2007. VTC-2007 Fall. 2007 IEEE 66th, pages 1441–1445. IEEE, 2007.

206. Wengerter, C., Ohlhorst, J., and von Elbwart, A. G. E.: Fairness and throughput analysis for generalized proportional fair frequency scheduling in OFDMA. In Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st, volume 3, pages 1903–1907. IEEE, 2005.

207. Andrade, C. E., Byers, S. D., Gopalakrishnan, V., Halepovic, E., Majmundar, M., Poole, D. J., Tran, L. K., and Volinsky, C. T.: Managing massive firmware-over-the-air updates for connected cars in cellular networks. In Proceedings of CarSys '17, pages 65–72, 2017.

208. Balasubramanian, P.: ”ledbat++: Low priority tcp congestion control in windows”. ICCRG meeting, IETF 100, 2017.

209. Wikipedia: LEDBAT: Low Extra Delay Background Transport. https://en.wikipedia.org/wiki/LEDBAT.

210. Bodrog, L., Horváth, G., and Vulkán, C.: ”analytical tcp throughput model for high-speed downlink packet access”. IET software, 3(6):480–494, 2009.

211. Falaki, H., Lymberopoulos, D., Mahajan, R., Kandula, S., and Estrin, D.: ”a first look at traffic on smartphones”. In Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, IMC ’10, pages 281–287, New York, NY, USA, 2010. ACM.

212. Chen, J., Mahindra, R., Khojastepour, M. A., Rangarajan, S., and Chiang, M.: ”a scheduling framework for adaptive video delivery over cellular networks”. In Proceedings of the 19th Annual International Conference on Mobile Computing &#38; Networking, MobiCom ’13, pages 389–400, New York, NY, USA, 2013. ACM.

213. Xie, X., Zhang, X., and Zhu, S.: ”accelerating mobile web loading using cellular link information”. In Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys ’17, pages 427–439, New York, NY, USA, 2017. ACM.

214. Takahashi, E., Suzuki, T., Onishi, T., and Satoda, K.: ”autonomous off-peak data transfer by passively estimating overall lte cell load”. In 2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC), pages 754–759, Jan 2017.

215. Goyal, P., Alizadeh, M., and Balakrishnan, H.: ”rethinking congestion control for cellular networks”. In Proceedings of the 16th ACM Workshop on Hot Topics in Networks, HotNets-XVI, pages 29–35, New York, NY, USA, 2017. ACM.

216. Leong, W. K., Xu, Y., Leong, B., and Wang, Z.: ”mitigating egregious ack delays in cellular data networks by eliminating tcp ack clocking”. In 2013 21st IEEE International Conference on Network Protocols (ICNP), pages 1–10, Oct 2013.

217. Michelinakis, F., Bui, N., Fioravantti, G., Widmer, J., Kaup, F., and Hausheer, D.: Lightweight mobile bandwidth availability measurement. In 2015 IFIP Networking Conference (IFIP Networking), pages 1–9. IEEE, may 2015.

218. Oshiba, T., Nogami, K., Nihei, K., and Satoda, K.: ”robust available bandwidth estimation against dynamic behavior of packet scheduler in operational lte networks”. In 2016 IEEE Symposium on Computers and Communication (ISCC), pages 1276–1283, June 2016.

219. Paul, A. K., Tachibana, A., and Hasegawa, T.: ”an enhanced available bandwidth estimation technique for an end-to-end network path”. IEEE Transactions on Network and Service Management, 13(4):768–781, Dec 2016.

220. Paul, A. K., Tachibana, A., and Hasegawa, T.: ”next-fit: Available bandwidth measurement over 4g/lte networks – a curve-fitting approach”. In 2016 IEEE 30th International

Conference on Advanced Information Networking and Applications (AINA), pages 25–32, March 2016.

221. Cisco visual networking index: Forecast and methodology, 2016–2021, 2017.

222. Internet of things (iot) connected devices installed base worldwide from 2015 to 2025 (in billions), 2015.

223. Kelly, F. P., Maulloo, A. K., and Tan, D. K. H.: Rate control for communication networks: Shadow prices, proportional fairness and stability. The Journal of the Operational Research Society, 49(3):237–252, 1998.

224. Alizadeh, M., Greenberg, A., Maltz, D. A., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., and Sridharan, M.: ”data center tcp (dctcp)”. In Proceedings of the ACM SIGCOMM 2010 conference, SIGCOMM '10, pages 63–74, New York, NY, USA, 2010. ACM.

225. Demers, A., Keshav, S., and Shenker, S.: Analysis and simulation of a fair queueing algorithm. In Symposium Proceedings on Communications Architectures &Amp; Protocols, SIGCOMM '89, pages 1–12, New York, NY, USA, 1989. ACM.

226. Hardy, G. H., Wright, E. M., et al.: An introduction to the theory of numbers. Oxford university press, 1979.

227. Ananthanarayanan, G., Ghodsi, A., Shenker, S., and Stoica, I.: Effective straggler mitigation: Attack of the clones. In Presented as part of the 10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13), pages 185–198, 2013.

228. Shah, N. B., Lee, K., and Ramchandran, K.: When do redundant requests reduce latency? IEEE Transactions on Communications, 64(2):715–722, 2016.

229. Wang, D., Joshi, G., and Wornell, G.: Efficient task replication for fast response times in parallel computation. In ACM SIGMETRICS Performance Evaluation Review, volume 42, pages 599–600. ACM, 2014.

230. Gardner, K., Zbarsky, S., Doroudi, S., Harchol-Balter, M., and Hyytia, E.: Reducing latency via redundant requests: Exact analysis. ACM SIGMETRICS Performance Evaluation Review, 43(1):347–360, 2015.

231. Lee, K., Pedarsani, R., and Ramchandran, K.: On scheduling redundant requests with cancellation overheads. IEEE/ACM Transactions on Networking, 25(2):1279–1290, 2017.

232. Joshi, G., Soljanin, E., and Wornell, G.: Efficient redundancy techniques for latency reduction in cloud systems. ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS), 2(2):12, 2017.

233. Suh, C. and Ramchandran, K.: Exact-repair mds code construction using interference alignment. IEEE Transactions on Information Theory, 57(3):1425–1442, 2011.

234. Tamo, I., Wang, Z., and Bruck, J.: Zigzag codes: Mds array codes with optimal rebuilding. IEEE Transactions on Information Theory, 59(3):1597–1616, 2013.

VITA

SHANYU ZHOU

Education	Ph.D. Electrical and Computer Engineering University of Illinois at Chicago	2014 – 2019
	B.Eng. Communication Engineering University of Electronic Science and Technology of China	2010 – 2014
Publications	Shanyu Zhou, Muhammad Usama Chaudhry, Emir Halepovic, Vijay Gopalakrishnan, Balajee Vamanan, Hulya Seferoglu, “Sneaker: Managing Background Traffic in Cellular Networks,” *in Proc. of IEEE LANMAN*, 2019.	
	Shanyu Zhou, Hulya Seferoglu, “Connectivity-Aware Traffic Phase Scheduling for Heterogeneously Connected Vehicles,” *in Proc. of ACM CarSys Workshop*, 2016.	
	Shanyu Zhou, Hulya Seferoglu, Erdem Koyuncu, “Blocking Avoidance in Wireless Networks,” *in Proc. of ACM CarSys Workshop*, 2016.	
	Shanyu Zhou, Hulya Seferoglu, “Blocking Avoidance in Transportation Systems,” *in Proc. of IEEE Allerton*, 2015.	
Awards	National Science Foundation (NSF) Travel Grant (ONAP Academic Summit, USA, 2018)	
	Student Presenter Award (University of Illinois at Chicago, 2017)	
	Graduate Student Council (GSC) Travel Award (University of Illinois at Chicago, 2016)	
Presentations	“Learning to Slice Future Networks for Industrial Automation” (Presented to Nokia CTO & Bell Labs President) Nokia Bell Labs, Murray Hill, NJ, Aug. 2018.	

	“Connectivity-Aware Traffic Phase Scheduling for Heterogeneously Connected Vehicles”, ACM CarSys ’16 workshop, New York City, NY, Oct. 2016.
	“Blocking Avoidance in Transportation systems”, IEEE Allerton, Monticello, IL, Oct. 2015.
Experience	Summer Intern at Nokia Bell Labs (Jun. 2018 - Aug. 2018)
	Research Assistant at University of Illinois at Chicago (Aug. 2014 - Jul. 2019)