DELAY-ORIENTED ANALYSIS AND DESIGN OF OPTIMAL SCHEDULING ALGORITHMS

DISSERTATION

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of the Ohio State University

By
Dongyue Xue, B.S.
Graduate Program in Electrical and Computer Engineering

The Ohio State University
2013

Dissertation Committee:
Prof. Eylem Ekici, Advisor
Prof. Ness B. Shroff
Prof. Atilla Eryilmaz
Prof. Rebecca Kim
© Copyright by
Dongyue Xue
2013
ABSTRACT

The main focus of this research is in the area of optimal cross-layer scheduling in wireless networks. In the literature, back-pressure-based scheduling algorithms (e.g., maximal weight matching algorithm [100]) have been developed typically focusing on the throughput/utility optimality, while little emphasis is given to delay constraints or queuing delay analysis. However, in practical implementations, delay is an important metric for wireless multimedia applications such as Video-on-Demand (VoD), Voice-over-IP (VoIP), IPTV, etc. In this research, delay-oriented scheduling with optimal throughput/utility is studied. An important delay-related factor is finite buffer property. By Little’s Law, finite buffer sizes lead to upper-bounded end-to-end delay on a per-flow basis. In addition, the finite buffer property is also an important factor for Quality of Service (QoS) sensitive wireless network applications. Algorithms with finite buffer property can find their application in resource-limited wireless networks such as wireless sensor networks.

In this research, to improve delay performance and ensure finite buffer sizes, a novel virtual queue is first proposed that acts as a weight on the actual queue when scheduling link rates. Specifically, the product of the virtual and actual queue backlogs has been employed as a weight in the maximal weight matching schedules in [110][109][112] and as a weight on the random access probability in the CSMA algorithm in [117]. The network stability is achieved by shifting the burden from actual queues to the proposed virtual queues, while the actual queues are upper-bounded
by a finite buffer size. As a direct result, the delay performance is significantly improved. In the proof of the stability and throughput/utility optimality, a novel type of Lyapunov function is employed which multiplies the virtual queue backlog to the quadratic term of actual queue backlogs. This particular structure of Lyapunov function leads to the product form of virtual and actual queue backlogs in the proposed algorithms.

The scheduling algorithms with finite buffers are proposed and analyzed in both centralized and distributed settings. Specifically, in multi-hop wireless networks, a centralized cross-layer scheduling algorithm [110] is designed with a novel concept of virtual queues to achieve a throughput “ε-close” to the optimality with a tradeoff of $O(\frac{1}{\epsilon})$ in average end-to-end delay guarantees. The structure of this algorithm is employed to design centralized optimal power control algorithms with finite buffers in [109][112]. In [117], distributed cross-layer CSMA algorithms are proposed in a single-hop setting to guarantee finite buffer sizes. The algorithm is shown to achieve a utility arbitrarily close to the optimal value with a tradeoff in the finite buffer sizes. Both implementation and numerical results [117] show a far better delay performance for comparable utility levels in comparison with recent optimal CSMA algorithms (e.g., Q-CSMA in [83]).

An underlying reason for large delays under the throughput-optimal random access algorithms [83, 85, 28] is temporal starvation [61], the phenomenon of links “being starved for prolonged periods indefinitely often despite having good stationary throughput”. Temporal starvation also leads to bursty service and undesirable jitter performance. To address the delay and temporal starvation issues, A $v(t)$-regulated CSMA algorithm is proposed in [111] that achieves fully local implementation without global message passing. Under the proposed algorithm, only links with weights above
a certain threshold qualify to be scheduled. Since link weights are increasing functions of packet queue lengths, resources are scheduled to links with sufficiently large queue lengths only. This approach potentially alleviates the “persistent” scheduling problem of regular throughput-optimal CSMA algorithms.

In practical wireless scenarios (i.e., IEEE 802.11 standards and OFDM systems), the entire bandwidth is usually partitioned into multiple channels. While a single-channel setting is studied for throughput-optimal CSMA algorithms in [51, 83, 85, 48, 117], the queueing behavior and delay performance of these algorithms have not been addressed in a multi-channel scenario. To fill in the gap, the queueing behavior and the closed-form queuing delay of general CSMA algorithms are studied in [107] in a many-channel regime. Law-of-Large-Numbers convergence results and a steady state queuing analysis have been carried out theoretically and confirmed via numerical results. Furthermore, the queue backlog in steady state has been approximated in closed-form in a many-channel regime. This many-channel regime analysis is further extended to a cognitive radio scenario in [114], where a distributed scheduling algorithm is proposed to achieve at least a guaranteed fraction of the optimal throughput.
This document is dedicated to my wonderful family.
Without the help of the following people, I would not have been able to complete my dissertation. My heartfelt thanks to:

Dr. Eylem Ekici, for his guidance. I could not have asked for a better mentor. Without his help, I would not have had the opportunity to choose wireless networking as my specialization, nor would I have enjoyed the level of success I have achieved in this area of research.

Dr. Robert Murawski, for his help with hardware implementation of CSMA algorithms that ultimately developed into our WiOpt 2012 paper on distributed throughput-optimal scheduling with finite buffers.

Dr. Mehmet Can Vuran, for his contribution and advice on cognitive radio sensing that ultimately developed into our WiOpt 2012 paper on correlation-based cooperative spectrum sensing in cognitive radio networks.

Dr. Moustafa Youssef and Rania Ibrahim, for their help with hardware implementation of our v(t)-regulated CSMA algorithm.
VITA

1987 ............................  Born in Shanghai, China

2009 ............................  B.S. Information Engineering,
                                Shanghai Jiao Tong University
                                Shanghai, China

2009-Present ......................  Graduate Research Associate,
                                     The Ohio State University
                                     Columbus, OH, USA

Dongyue Xue and Eylem Ekici, *Delay-Guaranteed Cross-Layer Scheduling in Multi-Hop Wireless Networks*, IEEE/ACM Transactions on Networking (ToN), DOI: 10.1109/T-NET.2012.2230404, December 2012


Dongyue Xue and Eylem Ekici, *Efficient Distributed Scheduling in Cognitive Radio Networks under the Many-Channel Regime*, WiOpt, Tsukuba Science City, Japan, May 2013

Dongyue Xue and Eylem Ekici, *Queueing Behavior of Queue-Backlog-Based CSMA Scheduling under the Many-Channel Regime*, WiOpt, Tsukuba Science City, Japan, May 2013


Dongyue Xue, Robert Murawski, and Eylem Ekici, *Distributed Throughput-Optimal Scheduling with Finite Buffers*, WiOpt, Paderborn, Germany, May 2012

Dongyue Xue and Eylem Ekici, *Optimal Power Allocation in Multi-Hop Wireless Networks with Finite Buffers*, IEEE International Conference on Communications, Kyoto, Japan, June 2011

Dongyue Xue and Eylem Ekici, *Guaranteed Opportunistic Scheduling in Multi-Hop Cognitive Radio Networks*, INFOCOM 2011, Shanghai, China, April 2011

FIELDS OF STUDY

Major Field: Electrical and Computer Engineering

Specialization: Wireless Networking
# TABLE OF CONTENTS

Abstract .................................................. ii
Dedication .................................................. iv
Acknowledgments .......................................... vi
Vita ............................................................ vii
List of Figures ............................................ xiv

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction and Background</td>
</tr>
<tr>
<td>2</td>
<td>Related Work</td>
</tr>
<tr>
<td>2.1</td>
<td>Related Work</td>
</tr>
<tr>
<td>2.2</td>
<td>Preliminary Work</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Opportunistic Scheduling in Multi-Hop Cognitive Radio Networks (CRNs) with Collision Rate Guarantees</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Correlation-Based Cooperative Spectrum Sensing in Cognitive Radio Networks</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Cross-Layer Scheduling for Cooperative Multi-Hop Cognitive Radio Networks</td>
</tr>
<tr>
<td>3</td>
<td>Cross-Layer Centralized Scheduling with Finite Buffer</td>
</tr>
<tr>
<td>3.1</td>
<td>Delay-Guaranteed Cross-Layer Scheduling in Multi-Hop Wireless Networks</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Network Model</td>
</tr>
<tr>
<td>3.1.3</td>
<td>Control Scheduling Algorithm for Multi-Hop Wireless Networks</td>
</tr>
<tr>
<td>3.1.4</td>
<td>Further Discussions</td>
</tr>
<tr>
<td>3.1.5</td>
<td>Numerical Results</td>
</tr>
<tr>
<td>3.1.6</td>
<td>Conclusions</td>
</tr>
</tbody>
</table>
3.2 Power Optimal Control with Finite Buffers .......................... 67
  3.2.1 Introduction .................................................. 67
  3.2.2 Network Model ............................................... 69
  3.2.3 Power-Optimal Scheduling Algorithm (PSA) for Multi-Hop Wireless Networks ........................................... 77
  3.2.4 Throughput-Optimal Algorithm (TSA) for Multi-Hop Wireless Networks .................................................. 87
  3.2.5 Numerical Results ............................................. 93
  3.2.6 Conclusions .................................................. 99

4 Distributed Utility-Optimal Scheduling with Finite Buffers ............ 100
  4.1 Introduction .................................................... 100
  4.2 Network Model .................................................. 103
    4.2.1 Network Elements ........................................ 103
    4.2.2 Network Constraints and Approaches ....................... 105
  4.3 The Proposed Algorithm for Single-Hop Networks ................... 107
    4.3.1 Distributed Link Rate Scheduler .......................... 115
    4.3.2 Proof of Theorem 4.3.4 .................................. 116
  4.4 Constructing Individual Average Delay Upper-bound ................. 122
  4.5 Approaching Time-Scale Separation ................................ 125
    4.5.1 Small Changes in $w_l(t)$ ................................ 125
    4.5.2 Slow Updates in $w_l(t)$ .................................. 126
  4.6 Implementation Study ........................................... 129
  4.7 Further Numerical Results ...................................... 131
    4.7.1 Effect of Parameter $V$ ................................... 132
    4.7.2 Comparative Performance Evaluation ....................... 133
    4.7.3 Simulation Comparison in Grid Topology .................... 135
  4.8 Conclusions .................................................. 137

5 $v(t)$ Regulated CSMA Algorithm ........................................ 139
  5.1 Introduction .................................................... 139
  5.2 $v(t)$-Regulated CSMA Algorithm .................................. 142
    5.2.1 Network Model ............................................. 142
    5.2.2 $v(t)$-Regulated CSMA Algorithm and its Distributed Implementation .................................................. 144
    5.2.3 Throughput Optimality of the $v(t)$-Regulated CSMA Algorithm .................................................. 148
    5.2.4 Proof of Lemma 5.2.2 ..................................... 151
    5.2.5 Proof of Theorem 5.2.4 ................................... 154
  5.3 Further Discussions ............................................. 157
    5.3.1 Approaching Time-Scale Separation ......................... 157
    5.3.2 Proof of Theorem 5.3.1 ................................... 159
6.10.5 Proof of Theorem 6.10.2 .............................................. 229
6.10.6 Proof of Lemma 6.10.5 ............................................. 235
6.10.7 Proof of Theorem 6.10.6 ............................................. 236

7 Conclusions ........................................................................ 240

7.1 Future Work ................................................................... 240
7.1.1 Queueing Behavior of CSMA Algorithm Employing Virtual
Queues ............................................................................. 241
7.1.2 Queueing Behavior of CSMA Algorithms with More General
Interference Models ......................................................... 243
7.1.3 The Case of the Numbers of Users and Channels Scaling
Together .......................................................................... 247

Bibliography ................................................................. 250


## LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Network topology</td>
</tr>
<tr>
<td>2.2</td>
<td>Transition diagram of primary activity</td>
</tr>
<tr>
<td>2.3</td>
<td>Performances of the optimal and GMM algorithms</td>
</tr>
<tr>
<td>2.4</td>
<td>Virtual Queue Structures for Sensing Quality Control</td>
</tr>
<tr>
<td>2.5</td>
<td>Energy consumption vs. number of nodes for (a) different sensing quality rates and (b) different correlation weights among neighbors</td>
</tr>
<tr>
<td>2.6</td>
<td>Energy consumption vs. (a) $P_{Tx}$ and (b) number of nodes $N$. $R_D = 0.9$, $w_{ij}(t) = 0.6$</td>
</tr>
<tr>
<td>2.7</td>
<td>Cooperative CRN model</td>
</tr>
<tr>
<td>2.8</td>
<td>Cooperative CRN topology for simulation</td>
</tr>
<tr>
<td>2.9</td>
<td>Algorithm performance with varying $\rho$</td>
</tr>
<tr>
<td>3.1</td>
<td>Queue relationship diagram where for clarity we only show the three types of virtual queues and the actual queue at the source node</td>
</tr>
<tr>
<td>3.2</td>
<td>Network topology for simulations</td>
</tr>
<tr>
<td>3.3</td>
<td>Virtual queue evolutions with backlogged sources</td>
</tr>
<tr>
<td>3.4</td>
<td>Throughput and delay tradeoff under Alg. with performances compared to Finite Buffer algorithm and BP algorithm, with varying arrival rates at the transport layer</td>
</tr>
<tr>
<td>3.5</td>
<td>Queue relationship diagram with decision variables $R_c(t)$, $\mu_{s(c)b(c)}(t)$, $(\mu_{mn}(t))$ and $P(t)$, where virtual queues $X_{mn}(t)$ are specially employed for $TSA$.</td>
</tr>
<tr>
<td>3.6</td>
<td>Network topology for simulations</td>
</tr>
</tbody>
</table>
3.7 Virtual queue dynamics for PSA algorithm ........................................... 95
3.8 Virtual queue dynamics for TSA algorithm ........................................... 96
3.9 Performance for PSA with fixed buffer size $q_M = 100$ ......................... 96
3.10 Performance for TSA with fixed minimum data rate requirement $a_c = 0.6$ and control parameter $V = 100$ .................................................. 98
4.1 Queue relationships ................................................................. 108
4.2 Distributed link rate scheduler for networks with single-hop transmissions. ................................................................. 117
4.3 Implementation results .............................................................. 131
4.4 Virtual queue evolutions ............................................................ 133
4.5 Algorithm comparison with different buffer sizes ............................ 135
4.6 Simulation comparison for constantly backlogged sources: delay versus throughput ................................................................. 136
4.7 Grid network topology for simulation comparison ............................ 137
4.8 Simulation comparison for constantly backlogged sources: delay versus throughput in grid topology ................................................................. 138
5.1 $v(t)$-regulated CSMA algorithm ................................................. 146
5.2 An equivalent scheduler for the proof in Lemma 5.2.2 ....................... 153
5.3 $v_T(t)$-regulated CSMA algorithm ................................................. 158
5.4 Distributed implementation of the $v(t)$-regulated CSMA algorithm ................................. 167
5.5 Implementation results on delay and 2nd moment of inter-service intervals with respect to arrival rate under topology in Figure 4.3(b), given $\eta_l = \eta_C$, $\forall l \in L$ .................................................. 168
5.6 Numerical results: Snapshot of Link schedules when arrival rate $\lambda = 0.19170$ ................................. 171
5.7 Numerical performance of delay and 2nd moment of inter-service interval with respect to threshold $\eta$ under topology in Figure 4.3(b), given arrival rate $\lambda = 0.1$ .................................................. 171
5.8 Numerical performance comparison of delay and 2nd moment of inter-service interval under grid topology in Figure 4.7, with $\eta_l = \eta_C$, $\forall l \in L$ 173
5.9 Numerical performance of delay and 2nd moment of inter-service interval with respect to threshold \( \eta \) under grid topology in Figure 4.7, when \( \lambda = 0.9 \) .................................................. 173

6.1 Average queue backlogs and service rates with different \( N \) ............... 206
6.2 Average queue backlogs and service rates with different arrival processes. 208
6.3 Average queue backlogs and service rates with different \( \alpha \) .......... 209
6.4 Dynamics of average queue backlogs with different weight function \( h \) 211
6.5 Dynamics of queue backlogs with heterogenous arrival rates ............ 211
6.6 Dynamics of average queue backlogs under fading channels .......... 213
6.7 Transition diagram of PU activity ............................................. 227
6.8 Queue dynamics, \( \rho = 0.03 \) ................................................. 227
6.9 Queue dynamics, \( \rho = 0.1 \) .................................................. 228
7.1 A single-cell system ............................................................. 244
7.2 Sample topologies of degree-\( d \) interference graph under a node-exclusive model ................................................................. 246
7.3 A system with overlapping cells .............................................. 247
CHAPTER 1
INTRODUCTION AND BACKGROUND

Modern wireless systems have seen increasing demand in data rates and quality of service in recent years as wireless services expand and new wireless applications emerge. To meet this demand, it is essential to fully exploit the capacity available to the wireless networks. As a result, optimal cross-layer design of congestion control, routing and scheduling algorithms has been one of the hottest topics in wireless networking. Earlier works have focused on throughput/utility optimality. Among these, the back-pressure algorithm first proposed in [100] and its extensions have been widely employed in developing throughput optimal dynamic resource allocation and scheduling algorithms for wireless systems. Back-pressure-based scheduling algorithms have also been employed in wireless networks with time-varying channels [82][22][70]. Congestion controllers at the transport layer have assisted the cross-layer design of scheduling algorithms in [26][79][20], so that the admitted arrival rate is guaranteed to lie within the network capacity region. However delay investigations are not included in these works.

Recent advances in wireless communication networks have called for delay-oriented scheduling. In practical implementations, delay is an important metric for wireless multimedia applications such as Video-on-Demand (VoD), Voice-over-IP (VoIP), IPTV, etc. Delay issues in single-hop wireless networks have been addressed in [33, 42, 41, 40, 39, 80, 55]. Delay-related scheduling in multi-hop wireless networks
have been proposed in [106, 104, 88, 13, 120, 54, 32]. However, the above-mentioned works assume the availability of infinite buffers and do not provide explicit queue backlog or delay guarantees.

Among the delay-related metrics, we are particularly interested in guaranteeing the finite buffer property in scheduling, since the finite buffer property leads to an upper-bounded average end-to-end delay given a lower-bounded per-flow data rate. In addition, buffer size is always finite in practical wireless systems, and limiting buffer sizes is also essential to mitigate playout buffer overflow problems in multimedia applications.

In this research, we design and analyze scheduling algorithms that guarantee the finite buffer property, in an aim to improve delay performances. We first consider a centralized network setting. In [110], we propose a cross-layer scheduling algorithm that achieves a throughput “$\epsilon$-close” to the optimal throughput in multi-hop wireless networks with a tradeoff of $O(\frac{1}{\epsilon})$ in average end-to-end delay guarantees. The algorithm guarantees finite buffer sizes and aims to solve a joint congestion control, routing, and scheduling problem in a multi-hop wireless network while satisfying per-flow average end-to-end delay constraints and minimum data rate requirements. As an extension of [110], we focus on the cross-layer design of optimal power control with finite buffers in [109][112]. The key technical contribution of [110][109][112] is the construction of a novel virtual queue at the source node. Instead of employing actual queue backlogs as weights in traditional back-pressure algorithms, we set the weight as the product of the virtual queue backlog and the actual queue backlog. This approach ensures both the finite buffer size property and throughput/power optimality.

However, the high complexity of the above introduced centralized algorithms (NP hard for general interference models [92]) render them impractical for implementation.
Low-complexity and distributed alternatives have been proposed in [105][14][65], but they can only guarantee a fraction of the capacity region in general. Recently, a class of random access algorithms has been introduced that achieves optimal throughput [51, 83, 85, 28]. These algorithms can be implemented in a distributed manner with local feedback from surrounding nodes. However, the delay performance of these algorithms, as evaluated through simulations, leave room for significant improvements [83]. Moreover, the delay analysis still remains elusive despite some preliminary results [48] that show a polynomial delay upper-bound for a fraction of the network capacity region.

To design the distributed implementation of our works [110][109][112], we propose a distributed cross-layer scheduling algorithm for networks with single-hop transmissions [117] that can guarantee finite buffer sizes and meet minimum utility requirements. Different from [51, 83, 85, 28], the link rate scheduler of our algorithm assigns the random access probabilities of a link as the product of the actual packet queue backlog and a virtual queue backlog. As a result, the network stability is achieved by shifting the burden from actual packet queues to our proposed virtual queues, while the actual queues are upper-bounded by a finite buffer size. The algorithm is theoretically shown to asymptotically achieve optimal throughput with growing buffer size. Our numerical and implementation results confirm the achievement of optimal throughput/utility and reveal superior delay performance of our algorithm.

An underlying reason for large delays under the throughput-optimal random access algorithms [83, 85, 28] is temporal starvation [61], the phenomenon of links being starved for prolonged periods indefinitely often despite having good stationary

\[^{1}\text{Multi-hop extensions have been discussed in the corresponding technical report.}\]
throughput”. Temporal starvation leads to bursty service and undesirable jitter performance. The reason for this behavior is the operation of the regular throughput-optimal CSMA algorithms [83, 85, 28]: These algorithms schedule a link that was already active with high probability for prolonged periods, even if there are few (or even no) packets in its queue, during which its neighboring links suffer from starvation. To address the delay and temporal starvation issues, in [111], we propose a $v(t)$-regulated CSMA algorithm that achieves fully local implementation without global message passing. Under the proposed algorithm, only links with weights above a certain threshold qualify to be scheduled. Since link weights are increasing functions of packet queue lengths, resources are scheduled to links with sufficiently large queue lengths only. This approach potentially alleviates the “persistent” scheduling problem of regular throughput-optimal CSMA algorithms.

In practical wireless settings (i.e., IEEE 802.11 standards and OFDM systems), the entire bandwidth is usually divided into multiple channels. While analysis has been provided for throughput/utility-optimal CSMA algorithms (including [117]) in a single-channel setting, the performance and queueing behavior of such CSMA algorithms have not been addressed in a multi-channel setting. To fill in the gap, we study the asymptotic queueing behavior of CSMA-based scheduling algorithms with a general Glauber dynamics (e.g., [51, 83, 85, 28]) for a fully-connected OFDM system under the many-channel regime in [107]. Law-of-large-numbers (LLNs) results have been established for individual queue backlogs and service rate. These results are the first of their kind for CSMA scheduling, which makes the queueing model of throughput-optimal CSMA algorithms tractable and scalable under a many-channel regime. In addition, the queue backlog in steady state has been approximated in closed form under the many-channel regime, to bring to light the congestion level of the considered system. While the upper-bound [48] of queue backlogs in the steady
state for a single-channel setting is an order result and is generally not tight, we show that the closed-form steady state queue backlog approximation [107] becomes accurate as the number of channel increases. Following the many-channel regime analysis in [107], we propose a distributed scheduling algorithm for a cognitive radio scenario in [114] that achieves at least a guaranteed fraction of the optimal throughput for secondary users.

The rest of the dissertation is organized as follows: Chapter 2 introduces the related work and the preliminary works we have carried out on scheduling algorithms in the area of cognitive radio networks. In Chapter 3, we describe the delay-guaranteed cross-layer scheduling algorithm [110] and provide its extension to power allocation schemes [109][112]. In Chapter 4, we introduce the CSMA algorithm utilizing the product form of actual and virtual queue backlogs proposed in [117]. The issue of delays and temporal starvation is addressed in Chapter 5. We study the queuing behavior of CSMA-based algorithms under the many-channel regime in Chapter 6. The dissertation is concluded in Chapter 7.
CHAPTER 2
RELATED WORK

2.1 Related Work

Recent advances in wireless communication, such as 4G cellular network, WiMAX, and cognitive radio network, have witnessed an increasing demand in both throughput and delay performance. To fully exploit the capacity available in the resource-limited wireless networks, researchers have been working on throughput/utility-optimal scheduling over the last two decades. The seminal work [100] on centralized back-pressure-based scheduling is proposed to achieve throughput-optimality by solving a maximal weight matching (MWM) problem, with each single-hop wireless link weighted by the corresponding packet queue backlog. Lyapunov optimization method has been employed to prove the stability of the network under the back-pressure-based algorithm. Lyapunov optimization has also been employed in optimal scheduling in [98][22][82][26][79][20]. Specifically, throughput-optimal scheduling is proposed for wireless networks with Markovian network state process and single-hop transmission in [98] via a quadratic Lyapunov function analysis. In [22], a fading channel is considered under a class of scheduling policies which generalize the work in [100][98] by assigning weight in MWM as a function of packet queue backlogs for optimal link rate scheduling. A time-varying channel is also considered in [82] for optimal power allocation and routing, where rate-power curves are utilized in the maximal weight
matching. While authors in [100][22][82] assume that arrival rates are within the stabilizable network capacity region, throughput-optimal cross-layer scheduling is studied in [26][79][20]. A congestion control component is added in [26][79][20], such that the admitted arrival rate is guaranteed to lie within the network capacity region. Specifically, a primal-dual congestion controller is proposed in [20] and its optimality is proved via a Laypunov drift analysis. Based on the joint power allocation and scheduling [82], the authors in [26][79] proposed an admission controller with a control parameter that trades off between the optimality and queue backlogs. A tutorial on cross-layer scheduling can be found in [68], where a primal-dual approach is employed with the associated Lagrange multiplier being a scalar multiple of the queue length. There are a number of works in the literature that do not utilize queue length information in optimal resource allocation. An opportunistic scheduling framework with fairness guarantees is proposed in [69] for utility optimization where the entire wireless resource is shared among single-hop wireless users. The optimality has been shown in [69] over the set of stationary scheduling policies. Via a dual optimization based approach in a multi-hop network, utility-optimal scheduling is proposed in [57][73]. With a same optimization problem, the algorithms [57][73] differ in their technical approaches: the optimization problem is decomposed into and solved separately via a user subproblem and a network subproblem in [57], whereas a gradient projection algorithm is proposed in [73]. However delay investigations are not included in these works.

In practical implementations, delay is an important metric for wireless multimedia applications. These delay-sensitive applications include voice/video-over-IP (e.g., Skype, Line, FaceTime, Google Voice, Google Hangouts), video-on-demand (e.g., Netflix, Amazon Instant Video), IPTV (e.g., U-verse, ZaapTV), live streaming (e.g., YouTube Live, Ustream), etc. Delay is an important metric for these services in
that: (i) Too much delay results in packet dropping; (ii) Excessive latency can render an application (e.g., voice-over-IP, online gaming) unusable; (iii) The variation in delay can seriously affect the quality of streaming audio/video, i.e., jitter/temporal starvation.

Delay issues in single-channel wireless resource scheduling have been addressed in [33, 80, 55, 106, 88, 13, 120, 72]. Especially, by studying the queue dynamics of single-hop traffic under an exclusive set (the set of communication links where no more than one link can be simultaneously scheduled), lower-bound and upper-bound on queuing delays have been studied in [33]. The scheduling algorithm in [80] provides a utility that is inversely proportional to the delay guarantee for single-hop traffic. A novel virtual queue is established in [80] and a quadratic Lyapunov function of this virtual queue is employed to prove the utility optimality. However, a packet dropping mechanism is required in [80] with no packet drop ratio guarantees, which makes the algorithm not suitable for implementation. Via a Lyapunov drift analysis, authors of [55] have obtained constant delay upper-bounds for two classes of scheduling policies (maximal weight matching and randomized independent set scheduling) that are dependent on the chromatic number of the underlying interference graph model. This upper bound is proved to be asymptotically tight in the sense that there exist classes of topologies where the expected delay attained by any scheduling policy is lower bounded by the same constant. Based on a token-based service discipline, the direction of link-based transmission is regulated in [106] to avoid loops in multi-hop transmission, which is further shown via numerical results to have superior delay performance over the back-pressure algorithm [100]. Optimal back-pressure algorithms are proposed in combination with hop counts in [120]. While traditional back-pressure algorithm exploits all paths (including very long ones) even when the traffic load is light, the proposed algorithm in [120] adaptively selects a set of routes
according to the traffic load so that long paths are used only when necessary, thus resulting in much smaller end-to-end packet delays as compared to the traditional back-pressure algorithm. A back-pressure type cluster-based scheduling is proposed in [88] to reduce delays in intermittently connected networks. In [88], there are two levels of routing: intra-cluster (traditional back-pressure algorithm within a cluster) and inter-cluster (a specific back-pressure-based algorithm used to determine the optimal load-balancing over an overlay network consisting of the gateway nodes of various clusters). The end-to-end buffer usage has been shown to be independent of the cluster size, reducing delays compared to the traditional back-pressure algorithm. Maximal weight matching with weight being a virtual queue is utilized in [13] to approximate order-optimal delays in multi-hop scheduling. In [13], per-neighbor queues are maintained at each node instead of per-flow queues required by the back-pressure algorithm to further reduce implementation complexity. A random access algorithm is proposed in [72] for lattice and torus interference graphs, which is shown to achieve order-optimal delay in a distributed manner with optimal throughput by periodically resetting its scheduling pattern. For a single server model, Andrews in [4][3] derive the techniques for bounding the delay violation probability for Earliest-Deadline-First (EDF) scheduling and Largest Weighted Delay First (LWDF) scheduling [97]. In [91][96], via a large deviation approach, it is shown that the workload is asymptotic minimized under two types of back-pressure-based algorithm under a heavy-traffic regime in a wireless switch. While explicit delay bounds are not provided in [91][96], the asymptotically tight bounds on steady-state queue lengths are further obtained under heavy traffic via a Lyapunov drift analysis in [21]. Under a multiple channel scenario, via a large-deviation approach, optimal scheduling for OFDM-based wireless networks has been proposed with asymptotic queue/delay-overflow probability guarantees in [104][10][8][9][7][47]. Specifically, Order-optimal queue-overflow probability
is guaranteed in [10][9] for ON/OFF channels and multi-state channels, respectively. A strictly positive value of the rate function (i.e., order-sub-optimal queue-overflow probability) is guaranteed in [8][7] for OFDM networks without and with single-carrier constraint,\(^1\) respectively. Order-optimal end-to-end-buffer-overflow probability and delay-overflow probability are considered in [104] and [47], respectively. However, the above-mentioned works assume the availability of infinite buffers and do not provide explicit queue backlog or delay guarantees.

There are several works via a centralized approach aiming to address end-to-end delay or buffer occupancy guarantees in multi-hop wireless networks. Worst-case delay is guaranteed in [81] with a packet dropping mechanism. However, dropped packets are not compensated or retransmitted with the algorithm of [81], which may lead to restrictions in its practical implementations. A low-complexity cross-layer fixed-routing algorithm is developed in [43] to guarantee order-optimal average end-to-end delay, but only for half of the capacity region. A scheduling algorithm for finite-buffer multi-hop wireless networks with fixed routing is proposed in [30] and is extended to adaptive-routing with congestion controller in [62]. Specifically, with a constantly backlogged transport layer, the algorithm in [62] guarantees \(O\left(\frac{1}{\epsilon}\right)\)-scaling in buffer size with an \(\epsilon\)-loss in throughput-utility, but this is achieved at the expense of the buffer occupancy of the source nodes, where an infinite buffer size in the network layer is assumed in each source node. The algorithm in [62] employs the product of the source node backlog and internal queue backlog as weight in the maximal weight matching. This leads to large average end-to-end delay since the network stability is achieved based on queue backlogs at these source nodes.

\(^1\)Under a single-carrier constraint, a given user can be allocated only consecutive frequency sub-bands.
Compared to the above works, the algorithms presented in [110][109][112] develop and incorporate novel virtual queue structures. Different from traditional back-pressure-based algorithms, where the network stability is achieved at the expense of large packet queue backlogs, in our algorithm, “the burden” of actual packet queue backlogs is shared by our proposed virtual queues, in an attempt to guarantee specific delay performances and finite buffer sizes. Specifically, inspired by the structure of maximal weight matching proposed in [62] and with a constantly backlogged packet generator as in [62][79], we design a congestion controller for a virtual input rate and assign weights in the scheduling policy as a product of actual packet queue backlog and the weighted backlog of a designed virtual queue. As such, the network stabilization is achieved with the help of virtual queue structures that do not contribute to delay in the network. Since all packet queues in the network, including those in source nodes, have finite sizes, all average end-to-end delays are bounded independent of length or multiplicity of paths.

While finite buffer sizes are guaranteed in [110][109][112], these centralized back-pressure-based algorithms usually incur high complexity (NP-hard in general [53]). In the literature, low-complexity works have been proposed to achieve a guaranteed fraction of the optimality dependent on the network interference model in [105][66][14] [65][89][5][38][52][67]. Specifically, in the area of topology, centralized greedy maximal matching algorithms have been studied in [89] that are proved to achieve at least a fraction $\frac{1}{K}$ of the maximal weight matching, where $K$ is the interference degree, i.e., the maximum number of non-interfering links in any interference set. The distributed implementation to find suboptimal maximal weight independent set has been studied in [5][38] with $O(N)$ complexity, where $N$ denotes the number of nodes. In interference-graph-based wireless networks, rate stability and queue stability are established, respectively, in maximal-matching-based distributed greedy scheduling
algorithms [14] and [105] for the fraction \( \frac{1}{K} \) of the capacity region with complexity \( O(\log N) \). A multi-channel multi-radio extension of [105] is studied in [66]. A distributed joint power allocation and scheduling algorithm is proposed in [67] under the node-exclusive model that achieves at least half of the capacity region. Essential to the distributed implementation in [67] is maximal matching (a matching such that no more links can be added without violating the interference constraint) and queue-length-based rate scheduling. In [66], a suboptimal random access algorithm is proposed under node-exclusive and two-hop interference models where channel access probability is a function of queue length. While the above algorithms all assume non-fading channel (i.e., channel capacity is constant and always available), a maximal matching based distributed algorithm is proposed in [52] to achieve at least a \( \frac{1}{K} \) fraction of the capacity region under an ON/OFF fading channel.

There have been two main stream of throughput-optimal distributed algorithms in the literature, Pick-and-Compare type of algorithms and random access algorithms. Pick-and-Compare type of algorithms have been proposed in [12][99][19][76] to achieve throughput optimality, but the complexity grows with the network size. The main idea behind the centralized algorithm in [99] is to sample a new candidate schedule uniformly from the independent scheduling set and switches to this new schedule if and only if it represents a larger weight. Inspired by [99], a distributed implementation is proposed in [76] via randomized gossip algorithms. In [19], a generic cross-layer mechanism is considered under the general interference model including the randomized gossip algorithm, a routing component, and a congestion controller. While the overhead of the algorithms [99][19] grows with the network size, Bui et al. in [12] proposed a class of distributed algorithms under primary interference model with constant overheads. However, the complexity of the algorithms [12][99][19][76] grow polynomially with network size.
It has recently been shown that random access algorithms employing the Glauber dynamics [83, 85, 28] can achieve optimal network throughput with $O(1)$ complexity, and polynomial delay upper-bounds have been proved in [48] for a fraction of capacity region in networks with single-hop transmissions. Specifically, denote by $Q_l$ the actual packet queue backlog for a communication link $l$ in the network. The link weights in the Glauber dynamics are selected in the form of $Q_l$ in [48], $\log Q_l$ in [83], $\log \log Q_l$ in [85], $\log Q_l y(\cdot)$ in [28], where $y(\cdot)$ is a function that increases arbitrarily slowly. The queue-length-based weight functions are studied and categorized in [64]. The resulting schedulers of the Glauber dynamics react to changes in $Q_l$. However, the queue lengths must be large enough to approach maximal weight matching [83][28], or to ensure a negative Lyapunov drift [85][48]. Instead, in our algorithm proposed in [117], the expense of large queue lengths is partially shifted to the virtual queues, which results in significantly more favorable delay performance.

An underlying reason for large delays under the above-mentioned Glauber-dynamics-based random access algorithms is temporal starvation [61]. In other words, links under these algorithms usually undergo prolonged periods of inactivity followed by a prolonged period of activity. There are a limited number of works analyzing the temporal starvation problems in the literature. The effect of number of channels on temporal starvation is analyzed in [61]. In [44], the temporal starvation problem is avoided via a virtual multi-channel approach and a congestion controller with complexity growing logarithmically with the network size, while regular throughput-optimal CSMA algorithms achieve distributed implementation with $O(1)$ complexity. Compared to regular throughput-optimal CSMA algorithms, under $v(t)$-regulated C-SMA algorithm proposed in our work [111], only links with weights above a certain threshold qualify to be scheduled. Since link weights are increasing functions of packet queue lengths, resources are scheduled to links with sufficiently large queue lengths.
only. This approach potentially alleviates the “persistent” scheduling problem of regular throughput-optimal CSMA algorithms. By favoring longer queues over shorter ones, delays in longer queues are potentially reduced, and this reduction outweighs the increase in the delay of packets in the shorter, unserved queues. Thus, the average delay is potentially reduced. In addition, under $v(t)$-regulated CSMA algorithm, the change in link schedules is more frequent, mitigating the temporal starvation problem.

Average queue backlog/delay performance for Glauber-dynamics-based random access algorithms has been studied in [48, 72, 11]. Specifically, through a mixing-time analysis, it is shown in [72] that an upper-bound for the time-averaged data queue backlog is of exponential order in the number of communication links in the network, and an upper-bound of polynomial order in the number of links is provided in [48] for a fraction of the capacity region. A lower-bound of expected data queue backlogs has been derived in [11] depending on the weight functions. However, these (average) delay bounds are not tight in general. In fact, no closed-form analysis/estimation of the queue backlog/delay performance of these random access algorithms is available in the literature, since interactions between links lead to complicated queuing behaviors and make such characterization intractable. Not surprisingly, queuing behavior and delay analysis of multi-channel extensions of these distributed algorithms is non-trivial and remains an open research problem, since the state space of schedules grows exponentially with the number of channels. In [107], we first generalize the random access algorithms from a single-channel setting [51, 83, 85, 28, 48] to a multi-channel one, denoted as multi-channel random access algorithm. By studying the equivalent deterministic single-queue system for the multi-channel random access algorithm, we find a closed-form approximation for the queue backlog in its steady state in the many-channel regime. While the delay bounds derived in [48, 72, 11] for
a single-channel setting are generally not tight, we show that the closed-form steady state queue backlog/delay approximation becomes accurate as the number of channel increases. Furthermore, in resonance with the findings in a single channel scenario [24, 11, 27], we show in [107] for a fully connected network that the more aggressive the weight function\(^2\) is, the smaller the asymptotic queue length becomes in the many-channel regime. We also note that the technical approach in our work [107] is different from the large deviation technique in [104][10][8][9][7][47] where centralized scheduling is proposed to achieve throughput optimality with asymptotically guaranteed queue/delay overflow probability in the many channel regime. Specifically, we develop Law of large Numbers (LLNs) results via the construction of an equivalent queue system. While the closed-form delay is not derived in [104][10][8][9][7][47], we provide a closed-form approximation for steady-state delay in [107].

\subsection{2.2 Preliminary Work}

In this section, we introduce three preliminary works of ours that employ virtual queues or achieve finite buffer sizes in scheduling algorithms designed for Cognitive Radio Networks (CRNs). We note that similar techniques have been employed to develop the algorithms proposed in the remaining chapters of the dissertation. In a typical CRN, licensed users are referred to as primary users (PUs) and secondary users (SUs) denote the users dynamically utilizing spectrum opportunities. The concept of CRN is simple, but the design of CRNs imposes challenges that are not present in conventional wireless networks [37]. Specific strategies should be developed for SUs to exploit channels in such a way that collisions between PUs and SUs transmission can be avoided.

\(^2\)Weight functions are employed to determine the transmission probability in the random access algorithms.
Specifically, we propose an opportunistic scheduling algorithm for multi-hop CRNs without cooperation [108] in Section 2.2.1 and with cooperation [113] in Section 2.2.3. Finite buffer sizes are guaranteed for SU packet queues and PU packet queues, respectively. A correlation-based cooperative spectrum sensing algorithm [115] is proposed utilizing a novel sensing deficiency virtual queue concept to meet sensing quality requirements in Section 2.2.2.

### 2.2.1 Opportunistic Scheduling in Multi-Hop Cognitive Radio Networks (CRNs) with Collision Rate Guarantees

The traditional back-pressure algorithm and its extensions ([100][26] and references therein) have been widely employed in developing throughput guaranteed dynamic resource allocation and scheduling algorithms for wireless systems. However, these works usually assume a single-channel setting with channel states known to nodes a priori, whereas in CRNs, SUs may not have knowledge of the exact current primary channel states. In addition, algorithms that work efficiently in a general single-channel network may not perform well in multi-channel CRNs, since PUs in different channels may have different requirements imposed on SUs such as collision rates and power interference constraints: these requirements are not addressed by a scheduling algorithm in the general single-channel network. Admission control and scheduling policies in multi-channel single-hop CRNs have been proposed in [102][71] to maximize throughput/utility subject to PU constraints. However, in the multi-hop scenario, one has to consider not only the exogenous arrivals to a secondary node but also the arrivals from neighboring nodes. Furthermore, the multi-hop setting forces a feasible scheduling algorithm to perform flow routing.

In our work [108], we consider a multi-channel multi-hop CRN overlayed with a PU network, e.g., Figure 2.1. With the constraints of collision rates observed by
PUs, we propose scheduling algorithms that can provably and asymptotically achieve guaranteed throughput in the CRN with finite buffer guarantees for each SU queue. Other contributions of our work include: (1) To the best of our knowledge, our work is the first of its kinds to develop congestion controller and scheduling policy for multi-hop CRNs with provable throughput guarantees. We also analyze lower-bound and upper-bound for average end-to-end delay under our algorithm; (2) our algorithm does not require full knowledge of PU current channel states and can guarantee that the collision rates observed by PUs are below a given threshold.

**Basic Idea of the Algorithm**

We introduce the base idea of the optimal collision-rate-guaranteed multi-hop scheduling algorithm. For any channel $l$, we impose a maximum collision rate constraint $\rho_l$ for the considered CRN. To model this collision rate constraint, we construct a virtual interference queue $X_l(t)$ for any given channel $l$ as follows:

$$X_l(t + 1) = [X_l(t) - \rho_l]^+ + C_l(t),$$

where $C_l(t) = 1$ if there is a collision in channel $l$ in the time slot $t$ and $C_l(t) = 0$ otherwise. Considering $C_l(t)$ as the arrival rate and $\rho_l$ as the service rate, and according to queueing theory, the time-averaged collision rate is upper-bounded by $\rho_l$ if queue $X_l(t)$ is stable, i.e.,

$$\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{X_l(\tau)\} < \infty, \quad \forall l.$$  

We denote by $U^c_n(t)$ the packet queue backlog at SU $n$ for commodity $c$ and by $(m, n)$ a link from SU node $m$ to SU node $m$. We also define $P_l(t)$ as the probability of
channel $l$ being idle in time slot $t$ given the previous channel state history. Then, the scheduling policy can be developed as a maximal weight matching (MWM) problem with weight $w_{mn}(t)P_l(t) - (1 - P_l(t))X_l(t)$ for each link $(m, n)$ and channel $l$, where

$$w_{mn}(t) \triangleq [U^c_{mn} - U^c_{mn}]^+,\)$$

$$c^*_{mn}(t) \triangleq \arg \max_c [U^c_m(t) - U^c_n(t)], \quad \forall (m, n) \in \mathcal{L}.$$ Specifically, for each link $(m, n)$, the commodity $c^*_{mn}(t)$ is fixed as the candidate for transmission. Note that $c^*_{mn}(t)$ and $w_{mn}(t)$ can be computed locally for each link. In the maximal weight matching, we can consider $w_{mn}(t)P_l(t)$ and $(1 - P_l(t))X_l(t)$ as the reward and the cost, respectively, of scheduling link $(m, n)$ and channel $l$.

Simulation Results

In the following, we present the simulation results for the proposed optimal algorithm and one of the extended suboptimal algorithms, GMM [108]. Simulations are run in Matlab 2009A.

In the network topology illustrated in Figure 2.1, we allow multiple PU activities to interfere with multiple secondary links. There are two source-destination pairs $(A, H)$ and $(D, E)$ with identical Poisson arrival rates. The secondary links $(B, F)$ and $(B, C)$ coexist in a single channel that belongs to primary user PU1. Similarly, the secondary links $(F, G)$ and $(C, G)$ employ a single channel designated to primary users PU2. The other links use mutually non-interfering channels that are free of PU activity. The transition behavior of both PUs from one slot to the next is illustrated in Figure 2.2, where state 0 denotes a busy state and state 1 denotes an idle state. Thus, we ensure a statistical channel availability of 0.5 for secondary links that share the resource with PUs. The collision rate threshold is set to 0.1.
We present the simulation results of the optimal algorithm and GMM algorithm in Figure 2.3, where the illustrated collision rate is the larger one of the collision rates experienced by PU1 and PU2. We observe that the collision rate threshold is not exceeded by either proposed algorithm. The throughput of the suboptimal GMM algorithm is close to the optimal algorithm. In addition, for both algorithms, the throughput is increased at the expense of a higher congestion level measured as the time-averaged total number of packets in the secondary network.

2.2.2 Correlation-Based Cooperative Spectrum Sensing in Cognitive Radio Networks

In [115], we developed (CORN)\(^2\), a correlation-based, optimal spectrum sensing algorithm for cognitive radio networks to minimize energy consumption.

To assess the spectral availability while maintaining efficient operation of CRNs,
effective spectrum sensing solutions are required [2]. The existing studies on spectrum sensing solutions indicate that collaboration among SUs improves the efficiency of spectrum utilization, and allows relaxation of the constraints at individual SUs [17]. However, network-wide effects of spatio-temporal sensing have not been formally analyzed except for heuristics in [17]. While collaboration is shown to improve sensing efficiency at the physical layer, two major tradeoffs exist in terms of network-wide considerations: (1) Cooperative spectrum sensing introduces communication overhead for the dissemination of local observations between SUs. Consequently, energy consumption associated with such communication overhead increases with increasing cooperation. (2) Spectrum utilization observed by closely located SUs is highly correlated due to the inherent spatial correlation in the received PU signals and correlated shadow fading. In addition to the spatial correlation between SUs, the observed information is also correlated in the time domain. The existing work, however, considers
local sensing is performed all the time. Instead, we argue that spectrum sensing information at a given space and time can represent spectrum information at a different point in the space-time space. Accordingly, an SU can improve its sensing quality at a particular time by using spectrum information observed by a different SU at a different time instead of local sensing.

Identifying the main objective of sensing as maintaining a given minimum sensing quality, in [115], we explore cooperative methods that will minimize the cost of sensing. The cost can flexibly be defined as a means to represent the resources spent (such as energy) or opportunities sacrificed for sensing (such as sensing duration). Since cooperation requires information exchange, these costs will also explicitly incorporate communication activities. Accordingly, we develop a provably optimal cooperative algorithm through a novel sensing deficiency virtual queue concept and exploit the correlation between SUs. The optimal algorithm further leads to a distributed solution when correlation weights are appropriately upper-bounded, which holds especially in low SNR environments with a high level of temporal correlation of spectrum sensing information.

In [116], a distributed Selective-(CORN)$^2$ (S-(CORN)$^2$) is introduced based on (CORN)$^2$ by extending the distributed algorithm to allow secondary users to select collaboration neighbors in densely populated cognitive radio networks.

**Basic Idea behind the Algorithm**

For sensing accuracy, we associate each pair of SU $i$ and channel $c$ with a minimum rate of information quality $R_D$ that needs to be maintained at all times. This minimum level can be achieved by sensing the channel locally and/or by exchanging spectrum sensing reports between other SUs in the vicinity. Specifically, When an SU $i$ senses a channel $c$ at a discrete time $t$, this event contributes to its sensing quality
by $\mu_{i,c}(t)$. The sensing quality is assumed to decay in time at a constant rate, and needs to be supplemented with additional sensing data. In addition to cooperation, each SU $i$ must also sense a channel $c$ locally at a rate of $R_S$. This requirement forces each SU $i$ to participate in sensing above a minimum rate and not rely solely on other nodes' observations.

To solve the minimum-energy-consumption sensing scheduling problem [115], we present a novel virtual deficiency queue concept, where the sensing dynamics are represented with virtual queuing structures, operating in discrete time domain. We define two types of per-node, per-channel virtual queues that track the dynamics of sensing quality as shown in Figure 2.4. The first virtual local sensing queue $Q^S_{i,c}(t)$ represents the local sensing events for channel $c$ in SU $i$ with a periodic arrival of “packets” at rate $R_S$ and an instantaneous service rate of $\mu_{i,c}(t)$. The evolution of the local sensing queue follows

$$Q^S_{i,c}(t + 1) = \left[Q^S_{i,c}(t) + R_S - \mu_{i,c}(t)\right]^+,\$$

where $[a]^+ \triangleq \max\{a, 0\}$. A “packet” arrival to the local sensing queue represents an increase in need for local sensing, which is satisfied when the packet is “served” and departs from the queue, i.e., node performs local sensing.

The second virtual queue is the total sensing deficiency queue $Q^D_{i,c}(t)$ for channel $c$ at SU $i$ with a periodic arrival of “deficiency packets” at rate $R_D$ and instantaneous service rate of $M_{i,c}(t)$. Each deficiency packet arrival to $Q^D_{i,c}(t)$ represents a decay in sensing quality. The decay in sensing quality is countered by the departure of “deficiency packets”, which corresponds to the improvement of the sensing information quality. In general, a large value of $Q^D_{i,c}(t)$ corresponds to a large sensing
deficiency, i.e., a low sensing information quality. With local sensing events and sensing reports gathered from neighbors, deficiency packets are “served”, which causes the $Q_{i,c}^D$ to shrink. This, in turn, corresponds to an improved sensing quality for the given SU-channel pair. The evolution of the total sensing deficiency queue follows

$$Q_{i,c}^D(t + 1) = [Q_{i,c}^D(t) + R_D - M_{i,c}(t)]^+. \tag{2.2.1}$$

When a sensing scheduling algorithm stabilizes the virtual queues $Q_{i,c}^S(t)$ and $Q_{i,c}^D(t)$, then this means that their respective arrival rates are smaller than or equal to their average service rates, i.e.,

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mu_{i,c}(t) \geq R_S, \quad \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} M_{i,c}(t) \geq R_D,$$

which are the two sensing constraints. Therefore, when the proposed algorithm stabilizes the system, the constraints of the optimal sensing scheduling problem are automatically satisfied.
Simulation Results

In this section, using numerical analyses via Matlab, we evaluate \((\text{CORN})^2\) and compare the performance of the distributed version of \((\text{CORN})^2\) [115] and the distributed S-(\(\text{CORN}\))^2 [116]. For the evaluations, a CRN of \(N\) SU nodes operating on a single channel is considered, where each node tries to estimate the spectrum occupancy on the channel using either non-cooperative or cooperative sensing employing \((\text{CORN})^2\).

Non-cooperative sensing is modeled as a special case of \((\text{CORN})^2\). The sensing deficiency queue, \(Q_{i,c}^D(t)\), is not considered and the spectrum sensing is scheduled according to only the local sensing queue, \(Q_{i,c}^S(t)\). Moreover, cooperation overhead is not considered. In our numerical evaluations, the cost represents energy consumption associated with sensing. We denote by \(w_{j,i}^c(t)\) the correlation weight, i.e.: when the sensing information of another SU \(j\) is used, its contribution is scaled by a factor of \(w_{j,i}^c(t)\), which captures the representativeness of \(j\)'s sensing data about channel \(c\) at \(i\). For notation simplicity, \(w_{j,i}^c(t)\) is replaced by \(w_{i,j}(t)\) in the following analysis. We use the following parameters: energy consumption of sensing \(P_S = 3.5\) mJ, energy consumption of broadcasting sensing data/receiving broadcasted sensing data \(P_T = P_R = 0.1125\) mJ, which are consistent with the values reported in [77]. Each data point represents the average values observed over 10,000 simulated time slots.

In Figure 2.5(a), the energy consumption per node per time slot is shown as a function of the number of nodes in the network under \((\text{CORN})^2\) and the non-cooperative sensing case. The energy consumption performance is investigated for different rate of sensing quality, \(R_D\), values and \(R_S = 0.55\). It can be observed that cooperative sensing with \((\text{CORN})^2\) improves energy consumption compared to non-cooperative sensing when the number of nodes exceeds three, mainly owing to the fact that cost of cooperation is offset by its benefits for larger networks. For small clusters, \(R_D\) is observed to have a negative impact on the energy efficiency, where the energy
Figure 2.5: Energy consumption vs. number of nodes for (a) different sensing quality rates and (b) different correlation weights among neighbors.

The energy consumption of cooperative sensing can be as much as 75% higher than that of non-cooperative sensing. On the other hand, as the number of cooperating nodes increases, $R_s$ dominates scheduling decisions since satisfying local sensing queue constraints becomes sufficient to satisfy any sensing deficiency queue constraints.

We next investigate the effect of the correlation weights among neighboring nodes on the energy optimality. The local correlation values are assumed to be unity, i.e., $w_{i,i}(t) = 1$. In Figure 2.5(b), the energy consumption per node per time slot is shown as a function of the number of nodes for different correlation levels. The non-cooperative case is also shown. The results indicate that $(CORN)^2$ is able to save energy compared to the non-cooperative case and the savings increase with higher correlation levels. Perfect correlation results in the highest cost saving. When the correlation levels are sufficiently low, the cost of cooperation dominates benefits reaped. In the pathological case of no correlation, the cost increases with the number of nodes due to the increased cost of sensing information reception. However, this
information does not benefit the receivers in terms of sensing information accuracy and results in a larger energy consumption than the non-cooperative case. These results emphasize the fact that cooperation is not beneficial in every case, especially when the correlation between nodes is very limited and the energy consumption for communication is higher than that for sensing.

We compare the three distributed sensing algorithms: the distributed cooperative solution of (CORN)$^2$, the cooperative S-(CORN)$^2$, and the non-cooperative sensing case. Specifically, their energy consumption performance against $P_{Tx} = P_{Rx}$ (the energy consumption for broadcasting ($P_{Tx}$) and receiving ($P_{Rx}$) sensing information) is shown in Figure 2.6(a), where the sensing cost is fixed as $P_S = 3.5$ mJ, the number of nodes is set as $N = 9$. It can be observed that S-(CORN)$^2$ always outperforms (CORN)$^2$, since SUs can decide whether or not to receive sensing information of neighboring nodes (and hence save energy consumption) under S-(CORN)$^2$. We can also observe in Figure 2.6(a) that the saved energy consumption becomes much larger
under S-(CORN)$^2$ when the cost of communication increases. In cases where communication is cheaper than sensing (i.e., $P_{Tx}$ is comparatively smaller than $P_S$), cooperative sensing outperforms non-cooperative sensing with diminishing returns. For higher values of $P_{Tx}$ (i.e., from energy consumption perspective, when communication is expensive), the energy consumption of non-cooperative sensing can be smaller than cooperative sensing. Hence, through Figure 2.6(a), we have shown that the cost for communication with respect to sensing is an important factor trading off between local sensing and cooperation. Compared to (CORN)$^2$, S-(CORN)$^2$ can accommodate larger communication energy levels, while still outperforming non-cooperative communication.

We also investigate the effect of the number of users on energy consumption in Figure 2.6(b). As the number of users initially increases, a full cooperation is encouraged as the energy consumption decreases significantly. That is why the distributed solution of (CORN)$^2$ and S-(CORN)$^2$ coincide in energy consumption at $N = 2, 3$. As the number of users further increases, the communication cost increases, leading to an increase in energy consumption. We also note that when the number of users increases, some users may decide not to receive the broadcasted sensing information from neighbors under S-(CORN)$^2$ (i.e., excessive cooperation is avoided). This results in a lower energy consumption than the original (CORN)$^2$. Thus, as the number of users increases (i.e., as the SU network becomes populated), the increase in energy consumption under S-(CORN)$^2$ is not as sharp as that under (CORN)$^2$.

### 2.2.3 Cross-Layer Scheduling for Cooperative Multi-Hop Cognitive Radio Networks

Recall that a multi-hop CRN scheduling algorithm has been proposed in our previous work [108] with estimated end-to-end delay, but there is no cooperation between
PUs and SUs. In [59], a cooperative CRN is considered to optimize PU and SU utility, where SUs assist PU transmission in a two-hop relay scenario, which is not readily extendable to generic multi-hop CRNs and does not involve delay analysis. To the best of our knowledge, no throughput/utility-optimal scheduling algorithms have been proposed in the literature for cooperative multi-hop CRNs with investigations on distributed implementation. Furthermore, characterizing delay upper-bounds for PUs in a CRN is challenging, especially with the opportunistic access of SUs to the same channel.

In our work [113], we propose a throughput-optimal cross-layer scheduling algorithm for a multi-hop cooperative CRN under a property-rights model [59], where SUs relay data between PU pairs to gain access to the licensed spectrum. An illustrative example is shown in Figure 2.7, where the cooperative CRN is composed of an SU subnetwork and a PU subnetwork. The SU subnetwork consists of SUs communicating with a secondary base station over a single hop as assumed for IEEE 802.22. In the PU subnetwork, we consider a case where the channel condition is not desirable for the direct transmission between the PU and the primary base station due to physical separation. Thus, the PU is willing to “lease” a portion of the spectrum access to SUs in return for some form of service. Specifically, PU data is relayed by SUs from the source PU to PU base station, and SUs in return gain an time-share of the channel proportional to their assistance to the PU. The model illustrated in Figure 2.7 can be considered as a generalization of the overlay CRNs with two-hop relay [6, 31, 36]. The proposed algorithm solves the throughput maximization problem using pre-determined routes with a reward mechanism: the SUs are guaranteed a throughput proportional to the PU data they relay.

An optimal opportunistic scheduling scheme has been proposed in [69] to guarantee each user a proportional share of the network resource for a non-cognitive setting,
which is extended to a scenario of two-hop relay CRNs in [59]. Different from the approach employed in [69] which is not readily applicable to a general multi-hop setting, we employ Lyapunov optimization tools to develop the throughput-optimal scheduling algorithm in a multi-hop cooperative CRN. Salient contributions of our work with respect to the literature can be listed as follows: (1) The algorithm can achieve a PU throughput arbitrarily close to the optimal values, with a tradeoff between throughput and PU/SU packet queue length. Specifically, the PU achieves a throughput “$(\epsilon + \frac{1}{V_2})$-close” to the optimal value at a tradeoff of $O(\frac{V_2}{\epsilon})$ in average SU queue length and $O(V_2)$ in the deterministic PU buffer size. (2) The algorithm guarantees deterministically upper-bounded finite buffer sizes for PU queues in the CRN. Derived from the previous two features, we show that the algorithm achieves order optimal delay [43] for PU traffic, i.e., the delay is upper-bounded by the first order of the number of hops in a route. (3) Distributed implementation and an extended algorithm with a general long-term reward mechanism are discussed. (4) The immediate and long-term reward mechanisms introduced in this work, along with the proposed algorithms, provide a novel approach to guarantee SUs’ access to the opportunistic channel while avoiding unintentional collisions between PUs and SUs.

We consider an immediate reward mechanism for SUs in the CRN model, i.e., when a PU packet is admitted to the network, an appropriate number of SU packets are also
admitted in all SUs along the path relaying the admitted PU packet. Specifically, we let $\rho_k$ be the rate of reward for SUs when a PU packet is admitted to a fixed route $k$. The algorithm [113] is composed of two parts, namely, a congestion controller and a hop/link scheduler. The congestion controller generates and admits PU packets into the PU relay subnetwork, and a corresponding fraction of SU packets are admitted to their sources according to the immediate reward mechanism. The hop/link scheduler regulates the link transmission rates of the cooperative CRN.

**Simulation Results**

In the following, we present a simulation-based performance evaluation for our proposed throughput-optimal scheduling algorithm [113]. Simulation results are obtained using the topology shown in Figure 2.8, which consists of a PU source ($s_P$) and a PU destination ($d_P$). The primary traffic is relayed by SUs $A$, $B$, $C$ and $D$ with their own one-hop secondary traffic destined to the secondary destination ($d_S$). We employ the node-exclusive model as the underlying interference model for the cooperative CRN and consider two predetermined routes $P_1 = (s_P, A, B, d_P)$ and $P_2 = (s_P, C, D, d_P)$ for PU data relay. Both PU and SU traffics are assumed to be constantly backlogged at the sources. The results reflect averages obtained over 50000 time slots for each run.

![Cooperative CRN topology for simulation](image-url)
In Figure 2.9, we illustrate the throughput and congestion level performance of the algorithm against the route-specific reward parameters $\rho_1 = \rho_2 = \rho$, where the number of admitted secondary packets for each SU is $\rho$ times the admitted PU packets and note that SU throughput is the sum for all SUs. According to the topology and the immediate reward mechanism, we must have the following relation between PU and SU throughput:

$$\text{SU throughput} = 2\rho \times (\text{PU throughput}),$$  
(2.2.2)

noting that there are 2 SUs along each pre-determined route.

![Figure 2.9: Algorithm performance with varying $\rho$](image)

We observe initially that the PU throughput decreases and SU throughput increases linearly when $\rho$ increases, satisfying (2.2.2). When $\rho$ further increases to around 0.75, SU throughput reaches and stays at its allowed maximum (1 packet/slot with
the node-exclusive interference model), hence the LHS of (2.2.2) becomes constant, which leads to a faster decrease in PU throughput linearly with respect to $\rho$ to guarantee (2.2.2).

With an increasing $\rho$, the allocated share of SU increases, which results in a corresponding small linear increase in SU congestion levels. When $\rho$ further increases to a level that allows SUs to reach their capacity (which is 1 packet/slot), the SU congestion level increases significantly, which is necessary for the SUs to approach their allowed maximum throughput.

An increasing $\rho$ reduces the throughput of the PU as more capacity is allocated to SUs. A decreased PU throughput level requires smaller queue sizes, which is reflected in the initial linear decrease observed in the PU congestion level. When $\rho$ further increases to around 0.75 (where SU throughput reaches its allowed maximum), PU congestion level drops significantly.  

---

3This observed drop in PU congestion level can be interpreted as follows. SU traffic becomes congested when SU throughput reaches the capacity, which results in a low level of PU queue backlogs according to the congestion controller in [113]. The succeeding PU queue backlogs along the routes will be shaped accordingly by the hop-back-pressure-based hop/link scheduler [113], which leads to a significant decrease in the PU congestion level.
CHAPTER 3
CROSS-LAYER CENTRALIZED SCHEDULING WITH
FINITE BUFFER

3.1 Delay-Guaranteed Cross-Layer Scheduling in Multi-Hop
Wireless Networks

3.1.1 Introduction

In this section, we propose a cross-layer algorithm to achieve guaranteed throughput
while satisfying network QoS requirements and guaranteeing that all actual queue
backlogs are deterministically upper-bounded. Specifically, we construct two virtual
queues, i.e., a virtual queue at transport layer and a virtual delay queue, to guarantee
average end-to-end delay bounds. Moreover, we construct a virtual service queue to
guarantee the minimum data rate required by individual network flows. Our cross-
layer design includes a congestion controller for the input rate to the virtual queue at
transport layer, as well as a joint policy for packet admission, routing, and resource
scheduling. We show that our algorithm can achieve a throughput arbitrarily close
to the optimal. In addition, the algorithm exhibits a tradeoff of $O\left(\frac{1}{\epsilon}\right)$ in the delay
bound, where $\epsilon$ denotes the distance from the optimal throughput.

Our main algorithm is further extended: (1) to a set of low-complexity suboptimal
algorithms; (2) from a model with constantly-backlogged sources to a model with
sources of arbitrary input rates at transport layer; (3) to an algorithm employing delayed queue information; and (4) from a node-exclusive model with constant link capacities to a model with arbitrary link capacities and interference models over fading channels. These extensions add to the versatility of the algorithm and show the applicability of the proposed virtual-queue-based algorithm to more general models.

Different from traditional back-pressure-based algorithms, where the network stability is achieved at the expense of large packet queue backlogs, in our algorithm, “the burden” of actual packet queue backlogs is shared by our proposed virtual queues, in an attempt to guarantee specific delay performances and finite buffer sizes. The most related work to our algorithm is the finite-buffer algorithm proposed in [62]. Specifically, with a constantly backlogged transport layer, the algorithm in [62] guarantees $O(\frac{1}{\epsilon})$-scaling in buffer size with an $\epsilon$-loss in throughput-utility, but this is achieved at the expense of the buffer occupancy of the source nodes, where an infinite buffer size in the network layer is assumed in each source node. The algorithm in [62] employs the product of the source node backlog and internal queue backlog as weight in the maximal weight matching. This leads to large average end-to-end delay since the network stability is achieved based on queue backlogs at these source nodes. Inspired by the structure of maximal weight matching proposed in [62], we design a congestion controller for a virtual input rate and assign weights in the scheduling policy as a product of actual packet queue backlog and the weighted backlog of a designed virtual queue. As such, the network stabilization is achieved with the help of virtual queue structures that do not contribute to delay in the network. Since all packet queues in the network, including those in source nodes, have finite sizes, all average end-to-end delays are bounded independent of length or multiplicity of paths.

The rest of the section is organized as follows: In Section 3.1.2, the network model is presented, followed by the introduction of queue dynamics for packet queues
and virtual queues. In Section 3.1.3, the optimal cross-layer control and scheduling algorithm is described, and its performance analyzed. In Section 3.1.4, we provide a class of feasible suboptimal algorithms, consider sources with arbitrary arrival rates at transport layer, employ delayed queue information in the scheduling algorithm, and extend the model to arbitrary link capacities and interference models over fading channels. We present numerical results in Section 3.1.5. Finally, we conclude this work in Section 3.1.6.

### 3.1.2 Network Model

#### Network Elements

We consider a time-slotted multi-hop wireless network consisting of \( N \) nodes and \( K \) flows. Denote by \((m, n) \in \mathcal{L}\) a link from node \( m \) to node \( n \), where \( \mathcal{L} \) is the set of directed links in the network. Denoting the set of flows by \( \mathcal{F} \) and the set of nodes by \( \mathcal{N} \), we formulate the network topology \( G = (\mathcal{N}, \mathcal{L}) \). Note that we consider adaptive routing scenario, i.e., the routes of each flow are not determined \( a \ priori \), which is more general than fixed-routing scenario. In addition, we denote the source node and the destination node of a flow \( c \in \mathcal{F} \) as \( b(c) \) and \( d(c) \), respectively.

We assume that the source node for flow \( c \) is always backlogged at the transport layer\(^1\). Let the scheduling parameter \( \mu_{mn}^c(t) \) denote the link rate assignment of flow \( c \) for link \((m, n)\) at time slot \( t \) according to scheduling decisions and let \( \mu_{s(c)b(c)}^c(t) \) denote the admitted rate of flow \( c \) from the packet generator of flow \( c \) to the source node, where \( s(c) \) denotes the packet generator of flow \( c \). It is clear that in any time slot \( t \),

\[
\mu_{s(c)n}^c(t) = 0 \quad \forall n \neq b(c).
\]

For simplicity of analysis, we assume only one packet can

---

\(^1\)Note that the constantly backlogged source is not necessarily a physical queue. It can be an application waiting for packet generation and admission, e.g., a variable rate multimedia encoder, and its end-to-end delay is the packet end-to-end delay in the network/data link layer. Thus, we can consider the backlogged source as a constantly backlogged packet generator.
be transmitted over a link in one slot, so \((\mu_{mn}^c(t))\) takes values in \(\{0, 1\}\) \(\forall (m,n) \in \mathcal{L}\). We also assume that \(\mu_{s(c)b(c)}^c(t)\) is bounded above by a constant \(\mu_M \geq 1\):

\[
0 \leq \mu_{s(c)b(c)}^c(t) \leq \mu_M, \ \forall c \in \mathcal{F}, \forall t,
\]

(3.1.1)
i.e., a source node can receive at most \(\mu_M\) packets from the transport layer in any time slot. To simplify the analysis, we prevent looping back to the source, i.e., we impose the following constraints

\[
\sum_{m \in \mathcal{N}} \mu_{mb(c)}^c(t) = 0 \ \forall c \in \mathcal{F}, \forall t.
\]

(3.1.2)

We employ the node-exclusive model in our analysis, i.e., each node can communicate with at most one other node in a time slot. Note that our model is extended to arbitrary interference models with arbitrary link capacities and fading channels in Section 3.1.4.

We now specify the QoS requirements associated with each flow. The network imposes an average end-to-end delay threshold \(\rho_c\) for each flow \(c\). The end-to-end delay period of a packet starts when the packet is admitted to the source node from the transport layer and ends when it reaches its destination. Note that the delay threshold is a time-averaged upper-bound, not a deterministic one. In addition, each flow \(c\) requires a minimum data rate of \(a_c\) packets per time slot.

**Network Constraints and Approaches**

For convenience of analysis, we define \(\mathcal{L}^c \triangleq \mathcal{L} \cup \{(s(c), b(c))\}\), where the pair \((s(c), b(c))\) can be considered as a virtual link from the backlogged packet generator to the source node. We now model queue dynamics and network constraints in the multi-hop network. Let \(U_n^c(t)\) be the backlog of the total amount of flow \(c\) packets waiting for
transmission at node $n$. For a flow $c$, if $n = d(c)$ then $U^c_n(t) = 0 \ \forall t$; Otherwise, the queue dynamics is as follows:

$$U^c_n(t+1) \leq [U^c_n(t) - \sum_{i: (n,i) \in L} \mu^c_{ni}(t)]^+ + \sum_{j: (j,n) \in L^c} \mu^c_{jn}(t), \text{ if } n \in \mathcal{N} \setminus d(c),$$

(3.1.3)

where the operator $[x]^+$ is defined as $[x]^+ = \max\{x, 0\}$. Note that in (3.1.3), we ensure that the actual number of packets transmitted for flow $c$ from node $n$ does not exceed its queue backlog, since a feasible scheduling algorithm may not depend on the information on queue backlogs. The terms $\sum_{i: (n,i) \in L} \mu^c_{ni}(t)$ and $\sum_{j: (j,n) \in L^c} \mu^c_{jn}(t)$ represent, respectively, the scheduled departure rate from node $n$ and the scheduled arrival rate into node $n$ by the scheduling algorithm with respect to flow $c$. Note that (3.1.3) is an inequality since the arrival rates from neighbor nodes may be less than $\sum_j \mu^c_{jn}(t)$ if some neighbor node does not have sufficient number of packets to transmit. Since we employ the node-exclusive model, we have

$$0 \leq \sum_{c \in \mathcal{F}} [\sum_{i: (n,i) \in L} \mu^c_{ni}(t) + \sum_{j: (j,n) \in L^c} \mu^c_{jn}(t)] \leq 1, \forall n \in \mathcal{N}.$$

(3.1.4)

From (3.1.1)(3.1.2), we also have

$$\sum_{j: (j,n) \in L^c} \mu^c_{jn}(t) \leq \mu_M, \text{ if } n = b(c),$$

(3.1.5)

if it is ensured that no packets will be looped back to the source.

Now we construct three kinds of virtual queues, namely, virtual queue $U^c_n(t)$ at transport layer, virtual service queue $Z_c(t)$ at sources, and virtual delay queue $X_c(t)$, to assist the development of our algorithm. We will show later that the minimum
Table 3.1: Update rules of each type of queue, where \( R_c(t) \) is determined by a congestion controller and \( (\mu_{mn}^c(t))_{(m,n)\in L^c} \) is determined by a scheduling policy.

<table>
<thead>
<tr>
<th>Instantaneous arrival rate</th>
<th>Instantaneous departure rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>( U_n^c(t) )</td>
<td>( \sum_{j:(j,n)\in L^c} \mu_{jn}^c(t) )</td>
</tr>
<tr>
<td>( U_{s(c)}^c(t) )</td>
<td>( R_c(t) )</td>
</tr>
<tr>
<td>( Z_c(t) )</td>
<td>( a_c )</td>
</tr>
<tr>
<td>( X_c(t) )</td>
<td>( \sum_{n\in\mathcal{N}} U_n^c(t) )</td>
</tr>
</tbody>
</table>

Figure 3.1: Queue relationship diagram where for clarity we only show the three types of virtual queues and the actual queue at the source node.
data rate requirement is achieved when $U_{s(c)}^c(t)$ and $Z_c(t)$ are stable; the average end-to-end delay constraint is satisfied if $U_{s(c)}^c(t)$ and $X_c(t)$ are stable. With update rules of each type of queue summarized in Table 3.1 and a general queue relationship illustrated in Figure 3.1, we formally introduce the three virtual queue structures in the following. 

a) To maintain the functionality of the traditional back-pressure algorithms while ensuring finite buffers, we construct a virtual queue $U_{s(c)}^c(t)$ for each flow $c$ at transport layer that acts as a weight to the “back-pressure” of actual queues, which will be further explained in the next section. We denote the virtual input rate to the queue as $R_c(t)$ at the end of time slot $t$ and we upper-bound $R_c(t)$ by $\mu_M$. Let $r_c$ denote the time-average of $R_c(t)$. We update the virtual queue as follows:

$$U_{s(c)}^c(t + 1) = [U_{s(c)}^c(t) - \mu_{s(c)b(c)}^c(t)]^+ + R_c(t), \quad (3.1.6)$$

where the initial $U_{s(c)}^c(0) = 0$. Considering the admitted rate $\mu_{s(c)b(c)}^c(t)$ as the service rate, if the virtual queue $U_{s(c)}^c(t)$ is stable, then the time-average admitted rate $\mu_c$ of flow $c$ satisfies:

$$\mu_c \triangleq \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mu_{s(c)b(c)}^c(\tau) \geq r_c \triangleq \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} R_c(\tau). \quad (3.1.7)$$

b) To satisfy the minimum data rate constraints, we construct a virtual queue $Z_c(t)$ associated with flow $c$ as follows:

$$Z_c(t + 1) = [Z_c(t) - R_c(t)]^+ + a_c, \quad (3.1.8)$$

where the initial $Z_c(0) = 0$. Considering $a_c$ as the arrival rate and $R_c(t)$ as the service rate, if queue $Z_c(t)$ is stable, we have: $r_c \geq a_c$. Additionally, if $U_{s(c)}^c(t)$ is stable, then according to (3.1.7), the minimum data rate for flow $c$ is achieved.

c) To satisfy the
end-to-end delay constraints, we construct a virtual delay queue $X_c(t)$ for any given flow $c$ as follows:

$$X_c(t + 1) = [X_c(t) - \rho_c R_c(t)]^+ + \sum_{n \in \mathcal{N}} U^c_n(t)$$  \hspace{1cm} (3.1.9)

where the initial $X_c(0) = 0$. Considering the packets kept in the network in time slot $t$, i.e., $\sum_{n \in \mathcal{N}} U^c_n(t)$, as the arrival rate and $\rho_c R_c(t)$ as the service rate, and according to queueing theory, if queue $X_c(t)$ is stable, we have

$$\lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{n \in \mathcal{N}} U^c_n(\tau) \leq \rho_c \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} R_c(\tau) = \rho_c r_c.$$  

Furthermore, if $U^c_n(t)$ is stable, then according to (3.1.7), we have:

$$\frac{1}{\mu_c} \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{n \in \mathcal{N}} U^c_n(\tau) \leq \rho_c. \hspace{1cm} (3.1.10)$$

In addition, by Little’s Theorem, (3.1.10) ensures that the average end-to-end delay of flow $c$ is less than or equal to the threshold $\rho_c$ with probability (w.p.) 1.

If queues $U^c_n(t)$ and the three virtual queues are stable for all nodes and flows, we know that the network is stable (i.e., all queues at all nodes are stable) and the average end-to-end delay constraint and minimum data rate requirement are achieved. Specifically, the network is stable and constraints are met, if

$$\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{X_c(\tau)\} < \infty, \ \forall c;$$

$$\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{U^c_n(\tau)\} < \infty, \ \forall n \in \mathcal{N} \cup \{s(c) : c \in \mathcal{F}\};$$
We note that the decision variable $R_c(t)$ represents the arrival rate to the virtual queue $U_{s(c)}^c(t)$ and serves as the service rate in virtual queues $Z_c(t)$ and $X_c(t)$. The algorithm proposed in the next section provides an $R_c(t)$ congestion controller that determines the value of $R_c(t)$ for each time slot $t$. The decision variable $\mu_{s(c)b(c)}^c(t)$ represents both the physical packet admission rate (i.e., the arrival rate to the actual source queue $U_{b(c)}^c(t)$) and the virtual service rate of $U_{s(c)}^c(t)$. Our proposed algorithm provides a scheduling policy that determines the value of $\mu_{s(c)b(c)}^c(t)$ joint with the assignment of the link service rate $\mu_{mn}^c(t)$, $(m, n) \in \mathcal{L}$. The average end-to-end packet delay in the network layer is only affected by the number of packets in the actual packet queues. Even if a virtual queue has a large number of virtual packets, they are not propagated in the network through an actual packet queue.

Now we define the capacity region of the considered multi-hop network. An arrival rate vector $(z_c)$ is called admissible if there exists some scheduling algorithm (without congestion control) under which the node queue backlogs (not including virtual queues) are stable. We denote $\Lambda$ to be the capacity region consisting of all admissible $(z_c)$, i.e., $\Lambda$ consists of all feasible rates stabilizable by some scheduling algorithm without considering QoS requirements (i.e., delay constraints and minimum data rate constraints). To assist the analysis in the following sections, we let $(r^*_c, \epsilon, c)$ denote a solution to the following optimization problem:

$$
\max_{(r_c): (r_c + \epsilon) \in \Lambda} \sum_{c \in \mathcal{F}} r_c
$$

s.t. $r_c \geq a_c$, $\forall c \in \mathcal{F}$.

where $\epsilon$ is a positive number which can be chosen arbitrarily small. For simplicity of
analysis, we assume that \((a_c)\) is in the interior of \(\Lambda\) and without loss of generality, we assume that there exists \(\epsilon' > 0\) such that \(r_{c, \epsilon}^* \geq a_c + \epsilon' \forall c \in \mathcal{F}\). According to [78], we have

\[
\lim_{\epsilon \to 0} \sum_{c \in \mathcal{F}} r_{c, \epsilon}^* = \sum_{c \in \mathcal{F}} r_c^*,
\]

where \((r_c^*)\) is a solution to the following optimization:

\[
\max_{(r_c): (r_c) \in \Lambda} \sum_{c \in \mathcal{F}} r_c \\
\text{s.t. } r_c \geq a_c, \forall c \in \mathcal{F}.
\]

### 3.1.3 Control Scheduling Algorithm for Multi-Hop Wireless Networks

Now we propose a control and scheduling algorithm \(ALG\) for the introduced multi-hop model so that \(ALG\) stabilizes the network and satisfies the delay constraint and minimum data rate constraint. Given \(\epsilon\), the proposed \(ALG\) can achieve a throughput arbitrarily close to \(\sum_{c \in \mathcal{F}} r_{c, \epsilon}^*\) under certain conditions related to delay constraints which will be later given in Theorem 3.1.2.

The optimal algorithm \(ALG\) consists of two parts: a congestion controller of \(R_c(t)\), and a joint packet admission, routing and scheduling policy. We propose and analyze the algorithm in the following.

**Algorithm Description and Analysis**

Let \(q_M \geq \mu_M\) be a control parameter of the algorithm, which we will later show to be a deterministic upper-bound on the actual queues. We first propose a congestion controller for the input rate of virtual queues at transport layer:
1) Congestion Controller of $R_c(t)$:

$$
\min_{0 \leq R_c(t) \leq \mu_M} R_c(t) \left( \frac{(q_M - \mu_M) U_{s(c)}^c(t)}{q_M} - X_c(t) \rho_c - Z_c(t) - V \right)
$$

(3.1.11)

where $V > 0$ is a control parameter. Specifically, when $\frac{q_M - \mu_M}{q_M} U_{s(c)}^c(t) - X_c(t) \rho_c - Z_c(t) - V > 0$, $R_c(t)$ is set to zero; Otherwise, $R_c(t) = \mu_M$. It will be later shown in Theorem 3.1.2 that when $V$ is large, $\textit{ALG}$ approaches optimal throughput. We will further analyse the effect of parameter $V$ on the convergence behavior of the virtual queues with numerical results provided in Section 3.1.5.

After performing the congestion control, we perform the following joint policy for packet admission, routing and scheduling (abbreviated as scheduling policy):

2) Scheduling Policy: In each time slot, with the constraints of the underlying interference model as described in Section 3.1.2 including (3.1.1)(3.1.2)(3.1.4), the network solves the following optimization problem:

$$
\max_{(\mu_{mn}^c(t))} \sum_{m,n} \mu_{mn}^c(t) w_{mn}(t)
$$

(3.1.12)

s.t. $\mu_{mn}^c(t) = 0$ $\forall c \neq c_{mn}^*(t)$, $\forall (m, n) \in \mathcal{L}^c$,

$\mu_{mn}^c(t) = 0$ if $n = s(c)$, $\forall c \in \mathcal{F}$,

where $c_{mn}^*(t)$ and $w_{mn}(t)$ are defined as follows:

$$
c_{mn}^*(t) = \arg \max_{c \in \mathcal{F}} w_{mn}^c(t),
$$

$$
w_{mn}(t) = \left[ \max_{c \in \mathcal{F}} w_{mn}^c(t) \right]^+,
$$
with weight assignment as follows

\[
w_{mn}^c(t) = \begin{cases} 
\frac{U^c_{s(c)}(t)}{q_M} [U^c_m(t) - U^c_n(t)], & \text{if } (m,n) \in \mathcal{L} \text{ and } n \neq b(c), \\
\frac{U^c_{s(c)}(t)}{q_M} [q_M - \mu_M - U^c_{b(c)}(t)], & \text{if } (m,n) = (s(c), b(c)), \\
0, & \text{otherwise.}
\end{cases}
\] (3.1.13)

In addition, when \(w_{mn}(t) = 0\), without loss of optimality, we set \(\mu^c_{mn}(t) = 0\) \(\forall c \in \mathcal{F}\) to maximize (3.1.12).

Note that \(\mathcal{L} \cup \{(s(c), b(c)) : c \in \mathcal{F}\}\) forms the \((m,n)\) pairs in \((\mu^c_{mn}(t))\) over which the optimization (3.1.12) is performed. Thus, the optimization is a typical Maximum Weight Matching (MWM) problem. We first decouple flow scheduling from the MWM. Specifically, for each pair \((m,n)\), the flow \(c^*_{mn}(t)\) is fixed as the candidate for transmission. We then assign the weight as \(w_{mn}(t)\). Note also that although similar product form of the weight assignment (3.1.13) have been utilized in [30, 62], no virtual queues are involved there. Whereas in ALG, we assign weights as a product of weighted virtual queue backlog \(\frac{U^c_{s(c)}(t)}{q_M}\) and the actual back-pressure, in an aim to shift the burden of the actual queue backlog to the virtual backlog.

To analyze the performance of the algorithm, we first introduce the following proposition.

**Proposition 3.1.1.** Employing ALG, each queue backlog in the network has a deterministic worst-case bound:

\[
U^c_n(t) \leq q_M, \hspace{1cm} \forall t, \forall n \in \mathcal{N}, \forall c \in \mathcal{F}.
\] (3.1.14)
Proof. We use mathematical induction on time slot in the proof. When \( t = 0 \), \( U^c_n(0) = 0 \leq q_M \forall n, c \). In the induction hypothesis, we suppose in time slot \( t \) we have \( U^c_n(t) \leq q_M \forall n, c \). In the induction step, for any given \( n \in \mathcal{N} \) and \( c \in \mathcal{F} \), we consider two cases as follows: (1) We first consider the case when \( n = b(c) \), i.e., when \( n \) is the source node of flow \( c \). Since \( U^c_n(t) \leq q_M \) from the induction hypothesis, we further consider two subcases:

- In the first subcase, \( U^c_{b(c)}(t) \leq q_M - \mu_M \). Then according to the queue dynamics (3.1.3) and the inequality (3.1.5), \( U^c_{b(c)}(t + 1) \leq U^c_{b(c)}(t) + \mu_M \leq q_M \);

- In the second subcase, \( q_M - \mu_M < U^c_{b(c)}(t) \leq q_M \). According to the weight assignment (3.1.13), we have \( w^c_{s(c)b(c)}(t) < 0 \) which leads to \( \mu^c_{s(c)b(c)}(t) = 0 \). Hence, \( U^c_{b(c)}(t + 1) \leq U^c_{b(c)}(t) \leq q_M \) by (3.1.2), (3.1.3).

(2) In the second case, \( n \neq b(c) \), i.e., \( n \) is not the source node of flow \( c \). Similar to the first case, we further consider the following two subcases:

- In the first subcase, \( U^c_n(t) < q_M \). Then, since we employ node-exclusive model, \( U^c_n(t + 1) \leq U^c_n(t) + 1 \leq q_M \) by (3.1.3), (3.1.4).

- In the second subcase, \( U^c_n(t) = q_M \). According to the weight assignment (3.1.13) we have \( w^c_{mn}(t) \leq 0 \ \forall m : (m, n) \in \mathcal{L} \). Now, for any given node \( m : (m, n) \in \mathcal{L} \), we have: (i) If \( c \neq c^*_{mn}(t) \), then by (3.1.12), \( \mu^c_{mn}(t) = 0 \); (ii) Otherwise, \( c = c^*_{mn}(t) \), which induces \( w_{mn}(t) = [w^c_{mn}(t)]^+ = 0 \) and by the scheduling policy, \( \mu^c_{mn}(t) = 0 \). Hence \( \mu^c_{mn}(t) = 0 \ \forall m : (m, n) \in \mathcal{L} \), and \( U^c_n(t + 1) \leq U^c_n(t) = q_M \) by the queue dynamics (3.1.3).

The above analysis holds for any given \( n \in \mathcal{N} \) and \( c \in \mathcal{F} \). Therefore the induction step holds, i.e., \( U^c_n(t + 1) \leq q_M \ \forall n, c \), which completes the proof. \( \square \)

Now we present our main results in Theorem 3.1.2.
Theorem 3.1.2. Given that
\[
q_M > \frac{2N - 1 + \mu_M^2}{2\epsilon} + \mu_M \quad \text{and} \quad \rho_c > \frac{Nq_M}{r^*_c} \forall c \in \mathcal{F},
\]
\[(3.1.15)\]

\textbf{ALG} can achieve a throughput
\[
\liminf_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{c \in \mathcal{F}} \mathbb{E}\{\mu^c_s(\hat{b}(c))(\tau)\} \geq \sum_{c \in \mathcal{F}} r^*_c - \frac{B}{V},
\]
\[(3.1.16)\]

where \(B \triangleq \frac{1}{2} NKq_M \mu_M + K \frac{q_M - \mu_M^2}{q_M} \mu_M^2 + \frac{1}{2} \mu_M^2 \sum_{c \in \mathcal{F}} \rho_c^2 + \frac{1}{2} KN^2q_M^2 + \frac{1}{2} K \mu_M^2 + \frac{1}{2} K \sum_{c \in \mathcal{F}} a_c^2.
\]

In addition, \textbf{ALG} ensures that the virtual queues have a time-averaged bound:
\[
\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{c \in \mathcal{F}} \mathbb{E}\{U^c_s(\tau) + X_c(\tau) + Z_c(\tau)\} \leq \frac{B'}{\delta},
\]
\[(3.1.17)\]

where \(B' \triangleq B + VB_R\), with \(B_R\) and \(\delta\) constant positive numbers given in the next subsection.

\textbf{Remark 3.1.3.} (Network Stability): The inequalities (3.1.14) from Proposition 3.1.1 and (3.1.17) from Theorem 3.1.2 indicate that \textbf{ALG} stabilizes the actual and virtual queues. As an immediate result, \textbf{ALG} stabilizes the network and satisfies the average end-to-end delay constraint and the minimum data rate requirement. In addition, Proposition 3.1.1 states that the actual queues are deterministically bounded by \(q_M\), which ensures finite buffer sizes for all queues in the network, including those in source nodes.

\textbf{Remark 3.1.4.} (Optimal Throughput and Delay Analysis): Since \((U^c_s(t))\) are stable, the inequality (3.1.16) gives a lower-bound on the throughput that \textbf{ALG} can achieve. Given some \(\epsilon > 0\), since \(B\) is independent of \(V\), (3.1.16) also ensures that
**ALG** can achieve a throughput arbitrarily close to \( \sum_{c \in \mathcal{F}} r^*_c \). When \( \epsilon \) tends to 0, **ALG** can achieve a throughput arbitrarily close to the optimal value \( \sum_{c \in \mathcal{F}} r^*_c \) with the tradeoff in queue backlog upper-bound \( q_M \) and the delay constraints \( (\rho_c) \), both of which are lower-bounded by the reciprocal terms of \( \epsilon \) as shown in (3.1.15) in Theorem 3.1.2. In other words, the average end-to-end delay bound is of order \( O\left(\frac{1}{\epsilon}\right) \). We note that similar tradeoff between delay and throughput (though without finite buffer and end-to-end delay guarantees) has also been observed in earlier works such as [26, 119].

In **ALG**, the control parameter \( V \), which is typically chosen to be large, does not affect the actual queue backlog upper-bound or the average end-to-end delay bound. However, a larger \( V \) increases the upper-bound of the virtual queue backlogs (from (3.1.17)) and results in a slower convergence time of the virtual queues. Note that with a finite \( V \), all virtual queues are stable nevertheless from (3.1.17). More details on the effect of \( V \) are presented in Section 3.1.5 with numerical results. In comparison, in the algorithm proposed in [62], the authors show that the internal buffer size is deterministically bounded with order \( O\left(\frac{1}{\epsilon}\right) \), but at the expense of the buffer occupancy at source nodes which is of order \( O(V) \), where \( V \) has to be large enough for their algorithm to approach \( \sum_{c \in \mathcal{F}} r^*_c \). This design assumes an infinite buffer size at source nodes and typically results in congestion at the source nodes as shown in the simulation results in [62], which further indues an unguaranteed and large average end-to-end delay. Moreover, one can expect that there are no buffer-size guarantees for single-hop flows by employing the algorithm in [62]. In contrast, in our proposed **ALG**, we shift “the burden of \( V \)” from actual queues to virtual queues and ensure that the average end-to-end delay constraints are satisfied with finite buffer sizes for all actual packet queues.
Remark 3.1.5. (Implementation Issues): To update the virtual queue $X_c(t)$ and perform the $R_c(t)$ congestion controller at the transport layer, the queue backlog information of flow $c$ is crucial. This information can be collected back to the source node by piggy-backing it on ACK from each node. In order to account for such delay of queue backlog information, the $R_c(t)$ congestion controller (3.1.11) of the algorithm can employ delayed queue backlog of $X_c(t)$. Similarly, delayed queue backlog information of $U_{s(c)}^c(t)$ can be employed at the weight assignment (3.1.13) of the scheduling policy.

The modified algorithm and its validity are further discussed in Section 3.1.4. By employing delayed queue backlog information, we can extend the algorithm to distributed implementation in much the same way as in [105, 65] to achieve a fraction of the optimal throughput. In order to achieve a throughput arbitrarily close to the optimal value with distributed implementation, we can employ random access techniques [83, 85] in the scheduling policy with fugacities [48] chosen as $\exp\left\{\frac{\alpha U_{s(c)}^c(t)[U_{s(c)}^c(t) - U_{n}^c(t)]^+}{qM}\right\}$ for each link $(m,n) \in L$, where $\bar{U}_{s(c)}^c(t)$ is a local estimate (e.g., delayed information) of $U_{s(c)}^c(t)$ and $\alpha$ a positive weight. It can be shown that the distributed algorithm can still achieve an average end-to-end delay of order $O(\frac{1}{\epsilon})$ with the time-scale separation assumption [51, 83].

We prove Theorem 3.1.2 in the following.

Proof of Theorem 3.1.2

Before we proceed, we present the following lemma which will assist us in proving Theorem 3.1.2.

---

2Fugacity [48] determines the transmission probability of a link in the CSMA framework (also referred to as Glauber dynamics) introduced in [83, 85].

3Note that the random access works cited above either do not provide delay guarantees or are not readily extended to multi-hop settings.
Lemma 3.1.6. For any feasible rate vector \((\theta_c) \in \Lambda\) with \(\theta_c \geq a_c \forall c \in \mathcal{F}\), there exists a stationary randomized algorithm \textsc{Stat} that stabilizes the network with input rate vector \((\mu_{st(c)b(c)}^{\text{Stat}}(t))\) and scheduling parameters \((\mu_{mn}^{\text{Stat}}(t))\) independent of queue backlogs, such that the expected admitted rates are:

\[
\mathbb{E}\{\mu_{st(c)b(c)}^{\text{Stat}}(t)\} = \theta_c, \forall t, \forall c \in \mathcal{F}.
\]

In addition, \(\forall t, \forall n \in \mathcal{N}, \forall c\), the flow constraint is satisfied:

\[
\mathbb{E}\left\{\sum_{i:(n,i) \in \mathcal{L}} \mu_{ni}^{\text{Stat}}(t) - \sum_{j:(j,n) \in \mathcal{L}^c} \mu_{jn}^{\text{Stat}}(t)\right\} = 0.
\]

Note that it is not necessary for the randomized algorithm \textsc{Stat} to satisfy the average end-to-end delay constraints. Similar formulations of \textsc{Stat} and their proofs have been given in [26] and [79], so we omit the proof of Lemma 3.1.6 for brevity.

Remark 3.1.7. Note that \((\theta_c)\) can take values as \((r_{c,c}^* + \epsilon)\) or \((r_{c,c}^* - \frac{1}{2}\epsilon')\), where we recall \((r_{c,c}^* + \epsilon) \in \Lambda\) and \(r_{c,c}^* \geq a_c + \epsilon' \forall c \in \mathcal{F}\). According to the \textsc{Stat} algorithm in Lemma 3.1.6, we assign the input rates of the virtual queues at transport layer as \(R_{c}^{\text{Stat}}(t) = \mu_{st(c)b(c)}^{\text{Stat}}(t)\). Thus, we also have \(\mathbb{E}\{R_{c}^{\text{Stat}}(t)\} = \theta_c\), with the time-average of \(R_{c}^{\text{Stat}}(t)\) satisfying: \(r_{c}^{\text{Stat}} = \theta_c\). In addition, according to the aforementioned assignment, the virtual queues \(U_{s(c)}^c(t)\) under \textsc{Stat} are bounded above by \(\mu_M\).

To prove Theorem 3.1.2, we first let \(Q(t) = ((U_n^c(t)), (U_{s(c)}^c(t)), (X_c(t)), (Z_c(t)))\) and define the Lyapunov function \(L(Q(t))\) as follows:

\[
L(Q(t)) = \frac{1}{2} \left\{ \sum_{c \in \mathcal{F}} \frac{q_M - \mu_M}{q_M} U_{s(c)}^c(t)^2 + \sum_{c \in \mathcal{F}} X_c(t)^2 \right\} + \sum_{c \in \mathcal{F}} Z_c(t)^2 + \sum_{c \in \mathcal{F} n \in \mathcal{N}} \frac{1}{q_M} U_{n}^c(t)^2 U_{s(c)}^c(t) \right\}.
\] (3.1.18)
We note that the structure of the Lyapunov function is different from the standard quadratic one (e.g., [79]), in that there is a virtual queue backlog multiplied to the quadratic term of the actual queue backlog, i.e., $\frac{U_{c}^{e}(t)}{q_{M}}U_{n}^{e}(t)^{2}$. Such structure leads to the structure of the scheduler (3.1.13) in $ALG$, as we shall prove in the following.

We denote the Lyapunov drift by

$$\Delta(t) = \mathbb{E}\{L(Q(t+1)) - L(Q(t))|Q(t)\}. \quad (3.1.19)$$

From the queue dynamics (3.1.3)(3.1.6), we have:

$$\sum_{c \in F} \sum_{n \in N} \frac{1}{q_{M}}U_{n}^{c}(t+1)^{2}U_{s(c)}^{e}(t+1)$$

$$\leq \sum_{c \in F} \frac{1}{q_{M}} (R_{c}(t) + U_{s(c)}^{e}(t)) \sum_{n \in N} U_{n}^{c}(t+1)^{2}$$

$$\leq \mu_{M}q_{M}NK + \sum_{c \in F} \frac{1}{q_{M}}U_{s(c)}^{e}(t) \sum_{n \in N} \{U_{n}^{c}(t)^{2}\} \quad (3.1.20)$$

$$+ (\sum_{i:(n,i) \in \mathcal{L}} \mu_{n}^{e}(t))^{2} + (\sum_{j:(j,n) \in \mathcal{L}^{e}} \mu_{jn}^{e}(t))^{2}$$

$$- 2U_{n}^{c}(t)(\sum_{i} \mu_{ni}^{e}(t) - \sum_{j} \mu_{jn}^{e}(t)),$$

where we recall that $R_{c}(t) \leq \mu_{M}$. 

50
From (3.1.20), we have

\[
\frac{1}{2} \left( \sum_{c \in \mathcal{F}} \sum_{n \in \mathbb{N}} \frac{1}{q_M} (U_n^c(t+1)^2 U_{s(c)}^c(t+1) - U_n^c(t) U_{s(c)}^c(t)) \right)
\leq \frac{1}{2} \sum_{c \in \mathcal{F}} \frac{(2N - 1 + \mu_M^2) U_{s(c)}^c(t)}{q_M} + \frac{1}{2} N K q_M \mu_M
\]

(3.1.21)

where we employ the fact deduced from (3.1.4)(3.1.5) that \( \sum \mu_{ci}^c(t) \leq 1 \) and \( \sum \mu_{cj}^c(t) \leq 1 \) when \( n \neq b(c) \) and \( \sum \mu_{jn}^c(t) \leq \mu_M \) when \( n = b(c) \). Note that we use the summation index \( i \) and \( j \) interchangeably for convenience of analysis.
By squaring both sides of the queue dynamics (3.1.6)(3.1.8)(3.1.9) and employing (3.1.21), we obtain the following inequality on the Lyapunov drift (3.1.19):

\[
\Delta(t) - V \sum_{c \in F} \mathbb{E}\{ R_c(t) | Q(t) \} \leq B + \sum_{c \in F} \mathbb{E}\{ R_c(t) \left( \frac{(q_{M} - \mu_{M})U_{s(c)}^{c}(t)}{q_{M}} - X_c(t)\rho_c - Z_c(t) - V \right) | Q(t) \}
\]

\[
+ Nq_{M} \sum_{c \in F} X_c(t) + \sum_{c \in F} a_c Z_c(t)
\]

\[
+ \frac{1}{2} \sum_{c \in F} \left( 2N - 1 + \mu_{M}^2 \right) U_{s(c)}^{c}(t)
\]

\[
- \mathbb{E}\{ \frac{q_{M} - \mu_{M}}{q_{M}} \sum_{c \in F} U_{s(c)}^{c}(t) \mu_{s(c)|b(c)}(t) \}
\]

\[
+ \sum_{c \in F} \sum_{n \in \mathcal{N}} \frac{U_{n}^{c}(t)U_{s(c)}^{c}(t)}{q_{M}} \left( \sum_{j:(n,j) \in \mathcal{L}} \mu_{nj}^{c}(t) - \sum_{i:(i,n) \in \mathcal{L}} \mu_{in}^{c}(t) \right) | Q(t) \}.
\]

(3.1.22)

We can rewrite the last term of the right-hand side (RHS) of (3.1.22) by simple algebra as

\[
- \mathbb{E}\{ \sum_{c \in F} \sum_{(m,n) \in \mathcal{L}} \mu_{mn}^{c}(t) \frac{U_{s(c)}^{c}(t)}{q_{M}} \left( U_{m}^{c}(t) - U_{n}^{c}(t) \right) \}
\]

\[
+ \sum_{c \in F} \mu_{s(c)|b(c)}^{c}(t) \frac{U_{s(c)}^{c}(t)}{q_{M}} \left( q_{M} - \mu_{M} - U_{b(c)}^{c}(t) \right) | Q(t) \}.
\]

(3.1.23)

Then, the second term and the last term of the RHS of (3.1.22) are minimized by the congestion controller (3.1.11) and the scheduling policy (3.1.12), respectively, over a set of feasible algorithms including the stationary randomized algorithm STAT introduced in Lemma 3.1.6 and Remark 3.1.7. We can substitute into the second term of RHS of (3.1.22) a stationary randomized algorithm with admitted arrival.
rate vector \((r^*_{c,c})\) and into the last term with a stationary randomized algorithm with admitted arrival rate vector \((r^*_{c,c} + \epsilon)\). Thus, we have:

\[
\Delta(t) - V \sum_{c \in F} \mathbb{E}\{R_c(t)|Q(t)\} \\
\leq B - V \sum_{c \in F} r^*_{c,c} \\
- \sum_{c \in F} \frac{U^c_s(c)(t)}{q_M}(\epsilon(q_M - \mu_M) - \frac{2N - 1 + \mu^2_M}{2}) \\
- \sum_{c \in F} (r^*_{c,c} - a_c)Z_c(t) - \sum_{c \in F} (\rho_c r^*_{c,c} - Nq_M)X_c(t). \\
\]

(3.1.24)

When (3.1.15) holds, we can find \(\epsilon_1 > 0\) such that \(\epsilon_1 \leq \rho_c r^*_{c,c} - Nq_M \forall c \in F\) and \(\epsilon_1 \leq \frac{\epsilon(q_M - \mu_M) - \frac{2N - 1 + \mu^2_M}{2}}{q_M}\). Recall that \(\epsilon'\) is defined such that \(r^*_{c,c} \geq a_c + \epsilon' \forall c \in F\). Thus, we have:

\[
\Delta(t) - V \sum_{c \in F} \mathbb{E}\{R_c(t)|Q(t)\} \\
\leq B - \delta \sum_{c \in F} (X_c(t) + U^c_s(c)(t) + Z_c(t)) - V \sum_{c \in F} r^*_{c,c}, \\
\]

(3.1.25)

where \(\delta \equiv \min\{\epsilon_1, \epsilon'\}\).

We take the expectation with respect to the distribution of \(Q\) on both sides of (3.1.25) and take the time average on \(\tau = 0, ..., t - 1\), which leads to

\[
\frac{1}{t} \mathbb{E}\{L(Q(t))\} - \frac{V}{t} \sum_{\tau=0}^{t-1} \sum_{c \in F} \mathbb{E}\{R_c(\tau)\} \\
\leq B - V \sum_{c \in F} r^*_{c,c} \\
- \frac{\delta}{t} \sum_{\tau=0}^{t-1} \sum_{c \in F} \mathbb{E}\{X_c(\tau) + U^c_s(c)(\tau) + Z_c(\tau)\}. \\
\]

(3.1.26)
Since \( \limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{c} \mathbb{E}\{R_c(\tau)\} \) is bounded above (say, by a constant \( B_R \) with \( B_R \leq K\mu_M \)) and \( \mathbb{E}\{L(Q(t))\} \) is nonnegative, by taking limsup of \( t \) on both sides of (3.1.26), we have:

\[
\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{c \in F} \mathbb{E}\{X_c(\tau) + U^c_{s(c)}(\tau) + Z_c(\tau)\} \\
\leq \frac{B}{\delta} + \frac{V}{\delta} \left[ \limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{c \in F} \mathbb{E}\{R_c(\tau)\} - \sum_{c \in F} r^*_{c}\right] \\
\leq B' \delta,
\]

which proves (3.1.17). Similarly, by taking liminf of \( t \) on both sides of (3.1.26) and employing (3.1.7), we can prove (3.1.16).

### 3.1.4 Further Discussions

#### Suboptimal Algorithms

Solving MWM optimization problem can be NP-hard depending on the underlying interference model as indicated in [92]. In the following, we introduce a group of suboptimal algorithms that aim to achieve at least a \( \gamma \) fraction of the optimal throughput.

We denote the scheduling parameters of suboptimal algorithms by \( (\mu^c_{mn}(t)) \). For convenience, we also denote the scheduling parameters of \( \text{ALG} \) by \( (\mu^c_{mn}(t)) \). Algorithms are called suboptimal if the scheduling parameters \( (\mu^c_{mn}(t)) \) satisfy the following:

\[
\sum_{m,n} \mu^c_{mn}(t),\text{SUB}(t)w_{mn}(t) \geq \gamma \sum_{m,n} \mu^c_{mn}(t),\text{OPT}(t)w_{mn}(t), \tag{3.1.28}
\]
where $\gamma \in (0, 1)$ is constant and we recall that $e_{mn}^*(t)$ and $w_{mn}(t)$ are defined in Section 3.1.3. In addition, the congestion controller of suboptimal algorithms is the same as that of $\text{ALG}$ (3.1.11).

Following the same analysis of $\text{ALG}$, Proposition 3.1.1 holds for suboptimal algorithms, i.e., the queue backlogs are bounded above by $q_M$, and we derive the following theorem:

**Theorem 3.1.8.** Given that

$$q_M > \frac{2N - 1 + \mu_M^2}{2\gamma\epsilon} + \mu_M \text{ and } \rho_c > \frac{Nq_M}{\gamma r_{c,c}^*} \forall c \in \mathcal{F},$$

(3.1.29)

$$\exists \epsilon_2 > 0 \text{ s.t. } \gamma r_{c,c}^* \geq a_c + \epsilon_2 \forall c \in \mathcal{F},$$

a suboptimal algorithm ensures that the virtual queues have a time-averaged bound:

$$\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{c \in \mathcal{F}} \mathbb{E}\{U^c_{s_c}(\tau) + X_c(\tau) + Z_c(\tau)\} \leq \frac{\bar{B}}{\delta},$$

(3.1.30)

where $\bar{B} \triangleq B + \gamma V B_R$ and $\delta$ is a positive constant associated with condition (3.1.29).

In addition, a suboptimal algorithm can achieve a throughput

$$\liminf_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{c \in \mathcal{F}} \mathbb{E}\{\mu^c_{s_{c,b_c}}(\tau)\} \geq \gamma \sum_{c \in \mathcal{F}} r_{c,c}^* - \frac{B}{\nu}.$$

(3.1.31)

The proof of Theorem 3.1.8 follows that of Theorem 3.1.2, where we have employed (3.1.28) to (3.1.22)(3.1.23), and is omitted for brevity.

**Remark 3.1.9.** Remark 5: From Theorem 3.1.8, given an arbitrarily small $\epsilon$, we show that a suboptimal algorithm can at least achieve a throughput arbitrarily close to a fraction $\gamma$ of the optimal results $\sum_{c \in \mathcal{F}} r_{c,c}^*$. Suboptimal algorithms include the well-known Greedy Maximal Matching (GMM) algorithm [53] with $\gamma = \frac{1}{2}$ as well as
the solutions to the maximum weighted independent set (MWIS) optimization problem such as GWMAX and GWMIN proposed in [89] with $\gamma = \frac{1}{\Delta}$, where $\Delta$ is the maximum node degree of the network topology $G$. The delay bound and throughput tradeoff in Theorem 3.1.2 still hold in Theorem 3.1.8.

**Arbitrary Arrival Rates at Transport Layer**

Note that in the previous model description, we assumed that the flow sources are constantly backlogged, that is, the congestion controller (3.1.11) can always guarantee $R_c(t) = \mu_M$ when $\frac{q_M - \mu_M}{q_M}U_{s(c)}(t) - X_c(t)\rho_c - Z_c(t) - V \leq 0$. In the following, we present an optimal algorithm when the flows have arbitrary arrival rates at the transport layer.

Let $A_c(t)$ denote the arrival rate of flow $c$ packets at the beginning of the time slot $t$ at the transport layer. We assume that $A_c(t)$ is i.i.d. with respect to $t$ with mean $\lambda_c$. For simplicity of analysis, we assume $(\lambda_c)$ to be in the exterior of the capacity region $\Lambda$ so that a congestion controller is needed and we assume that $A_c(t)$ is bounded above by $\mu_M \forall c \in \mathcal{F}$.

Let $L_c(t)$ denote the backlog of flow $c$ data at the transport layer which is updated as follows:

$$L_c(t+1) = \min\{[L_c(t) + A_c(t) - \mu_{s(c)b(c)}(t)]^+, L_M\}, \quad (3.1.32)$$

where $L_M \geq 0$ is the buffer size for flow $c$ at the transport layer. Note that we have $L_M = 0$ and $L_c(t) = 0$ if there is no buffer for flow $c$ at the transport layer.

Following the idea introduced in [26], we construct a virtual queue $Y_c(t)$ and an

---

Note that our analysis also works for the case when $A_c(t)$ is bounded above by some constant $A_M \forall c \in \mathcal{F}$, where $A_M \geq \mu_M$. 

---

56
auxiliary variable $v_c(t)$ for each virtual input rate $R_c(t)$, with queue dynamics for $Y_c(t)$ as follows

$$Y_c(t + 1) = [Y_c(t) - R_c(t)]^+ + v_c(t),$$  \hspace{1cm} (3.1.33)

where initially we have $Y_c(0) = 0$. The intuition is that $v_c(t)$ serves as the function of $R_c(t)$ in congestion controller (3.1.11) and we note that when $Y_c(t)$ is stable, we have $r_c \geq v_c$, where $v_c$ is the time average rate for $v_c(t)$, recalling that $r_c$ is the time average rate for $R_c(t)$. Thus, when $Y_c(t)$ and $U^c_{s(c)}(t)$ are stable, if we can ensure the value $\sum_c v_c$ is arbitrarily close to the optimal value $\sum_c r^*_c$, then so is the throughput $\sum_c \mu_c$ since $\mu_c \geq r_c \geq v_c$.

Now we provide the optimal algorithm for arbitrary arrival rates at the transport layer:

1) **Congestion Controller:**

\[
\min_{0 \leq v_c(t) \leq \mu_M} \quad v_c(t)(\eta Y_c(t) - V),
\]  \hspace{1cm} (3.1.34)

\[
\min_{R_c(t)} R_c(t)\left(\frac{qM - \mu_M}{qM}U^c_{s(c)}(t) - \eta Y_c(t) - X_c(t)\rho_c - Z_c(t)\right)
\]  \hspace{1cm} (3.1.35)

s.t. \hspace{1cm} 0 \leq R_c(t) \leq \min\{L_c(t) + A_c(t), \mu_M\}

where $\eta > 0$ is a weight associated with the virtual queue $Y_c(t)$. Note that (3.1.34) and (3.1.35) can be solved independently. Specifically, when $\eta Y_c(t) - V \geq 0$, $v_c(t)$ is set to zero; Otherwise, $v_c(t) = \mu_M$. When $\frac{qM - \mu_M}{qM}U^c_{s(c)}(t) - \eta Y_c(t) - X_c(t)\rho_c - Z_c(t) \geq 0$, $R_c(t)$ is set to zero; Otherwise, $R_c(t) = \min\{L_c(t) + A_c(t), \mu_M\}$.  

57
2) Scheduling Policy: The scheduling algorithm is the same as that of ALG provided in Section 3.1.3, except for the updated constraints: 

\[ 0 \leq \mu_{s(c)b(c)}^c(t) \leq \min\{L_c(t) + A_c(t), \mu_M\} \]

Since the scheduling policy is not changed, Proposition 3.1.1 still holds. And we present a theorem below for the performance of the algorithm:

**Theorem 3.1.10.** Given that

\[ q_M > \frac{2N - 1}{2\epsilon} + \mu_M \text{ and } \rho_c > \frac{Nq_M}{r_{e,c}^*} \quad \forall c \in \mathcal{F}, \]

the algorithm ensures that the virtual queues have a time-averaged bound:

\[ \limsup_{t \to \infty} \frac{1}{t} \sum_{t=0}^{t-1} \sum_{c \in \mathcal{F}} \mathbb{E}\{U_{s(c)}^c(\tau) + X_c(\tau) + Z_c(\tau) + Y_c(\tau)\} \leq \frac{B_2}{\delta'}, \]

where \( B_2 \triangleq B + K\eta\mu_M^2 + VB_R \) and \( \delta' \) is constant positive number. In addition, the algorithm can achieve a throughput

\[ \liminf_{t \to \infty} \frac{1}{t} \sum_{t=0}^{t-1} \sum_{c \in \mathcal{F}} \mathbb{E}\{\mu_{s(c)b(c)}^c(\tau)\} \geq \sum_{c \in \mathcal{F}} r_{e,c}^* - \frac{B_1}{V}, \]

where \( B_1 \triangleq B + K\eta\mu_M^2 \).

**Proof:** Before we proceed to the proof, we extend the stationary randomized algorithm STAT introduced in Lemma 3.1.6 and Remark 3.1.7. Given \((\theta_c)\) introduced in Lemma 3.1.6 and given flow \( c \) at node \( n \), recall that \((A_c(t))\) is i.i.d. with mean \((\lambda_c)\) and \((\lambda_c) > (\theta_c)\) element-wise. The flow control for STAT can be given as: Admit \( \mu_{s(c)b(c)}^{c,\text{STAT}}(t) = A_c(t) \) w.p. \( \frac{\theta_c}{\lambda_c} \); otherwise, \( \mu_{s(c)b(c)}^{c,\text{STAT}}(t) = 0 \). Then \( \mathbb{E}\{\mu_{s(c)b(c)}^{c,\text{STAT}}(t)\} = \theta_c \), \( \forall t \). Now take \( v_{c}^{\text{STAT}}(t) = R_{c}^{\text{STAT}}(t) = \mu_{s(c)b(c)}^{c,\text{STAT}}(t) \forall c \in \mathcal{F} \). Then we also have
\[ \mathbb{E}\{v_{c}^{\text{STAT}}(t)\} = \mathbb{E}\{R_{c}^{\text{STAT}}(t)\} = \theta_{c}. \] Note that \( R_{c}^{\text{STAT}}(t) \leq A_{c}(t) \leq \min\{L_{c}(t) + A_{c}(t), \mu_{M}\} \) and \( v_{c}^{\text{STAT}}(t) \leq \mu_{M}. \)

Now we present the proof.

**Proof.** We define the Lyapunov function as
\[ L(Q'(t)) = L(Q(t)) + \frac{\eta}{2} \sum_{c \in F} Y_{c}^{2}(t) \]
and the Lyapunov drift as
\[ \Delta'(t) = \mathbb{E}\{L(Q'(t + 1)) - L(Q'(t))|Q'(t)\}, \]
where \( Q'(t) = (Q(t), (Y_{c}(t)). \) Following the steps in deriving (3.1.22)(3.1.23) and squaring both sides of the virtual queue dynamics (3.1.33), we have

\[ \Delta'(t) - V \sum_{c \in F} \mathbb{E}\{v_{c}(t)|Q'(t)\} \leq B_{1} + \sum_{c \in F} \mathbb{E}\{v_{c}(t)(\eta Y_{c}(t) - V)|Q'(t)\} \]
\[ + \sum_{c \in F} \mathbb{E}\{R_{c}(t)(\frac{q_{M} - \mu_{M})U_{s(c)}^{c}(t)}{q_{M}} \]
\[ - \eta Y_{c}(t) - X_{c}(t) \rho_{c} - Z_{c}(t)|Q'(t)\} \]
\[ + N q_{M} \sum_{c \in F} X_{c}(t) + \sum_{c \in F} a_{c} Z_{c}(t) \]
\[ + \frac{1}{2} \sum_{c \in F} (2N - 1 + \mu_{M})U_{s(c)}^{c}(t) \]
\[ - \frac{1}{2} \sum_{c \in F} \sum_{(m,n) \in L} \mu_{mn}^{c}(t) \frac{U_{s(c)}^{c}(t)}{q_{M}}(U_{m}^{c}(t) - U_{n}^{c}(t)) \]
\[ + \sum_{c \in F} \mu_{s(c)b(c)}^{c}(t) \frac{U_{s(c)}^{c}(t)}{q_{M}}(q_{M} - \mu_{M} - U_{b(c)}^{c}(t)))Q'(t)\}, \]

The second term, third term and the last term of the RHS of (3.1.36) are minimized by the congestion controller (3.1.34), (3.1.35) and the scheduling policy (3.1.12), respectively, over a set of feasible algorithms including the stationary randomized algorithm STAT. Substitute into the second term of RHS of (3.1.36) a stationary randomized algorithm with admitted arrival rate vector \((r_{c,e}^{*} - \frac{1}{2} \epsilon'),\) the third term a
stationary randomized algorithm with admitted arrival rate vector \( (r^*_{e,c}) \) and the last term a stationary randomized algorithm with admitted arrival rate vector \( (r^*_{e,c} + \epsilon) \). Then, following the steps in proving Theorem 3.1.2, we can prove Theorem 3.1.10.

Theorem 3.1.10 shows that optimality is preserved and \( O\left(\frac{1}{\epsilon}\right) \) delay scaling is kept.

**Employing Delayed Queue Backlog Information**

Recall that in \( ALG \), congestion controller (3.1.11) is performed at the transport layer and link weight assignment in (3.1.13) is performed locally at each link. Thus, in order to account for the propagation delay of queue information, we employ delayed queue backlog of \( (X_c(t)) \) in (3.1.11) and employ delayed queue backlog of \( (U^c_{s(c)}(t)) \) for links in \( L \) in (3.1.13). Specifically, we rewrite (3.1.11) in \( ALG \) as:

\[
\min R_c(t) \left( \frac{(q_M - \mu_M)U^c_{s(c)}(t)}{q_M} - X_c(t-T)\rho_c - Z_c(t) - V \right),
\]

where \( T \) is an integer number that is larger than the maximum propagation delay from a source to a node, and we rewrite (3.1.13) as:

\[
w^c_{mn}(t) = \begin{cases} 
\frac{U^c_{s(c)}(t-T)}{q_M} [U^c_{m}(t) - U^c_{n}(t)], & \text{if } (m, n) \in \mathcal{L}, \\
\frac{U^c_{s(c)}(t)}{q_M} [q_M - \mu_M - U^c_{b(c)}(t)], & \text{if } (m, n) = (s(c), b(c)), \\
0, & \text{otherwise}.
\end{cases}
\]

Proposition 3.1.1 still holds and we present a theorem for the scheduling algorithm using delayed queue backlog information, which maintains the throughput optimality and \( O\left(\frac{1}{\epsilon}\right) \) scaling in delay bound:
Theorem 3.1.11. Given that

\[ q_M > \frac{2N - 1 + \mu^2_M}{2\epsilon} + \mu_M \text{ and } \rho_c > \frac{Nq_M}{r^*_c, c} \quad \forall c \in \mathcal{F}, \]

the algorithm ensures that the virtual queues have a time-averaged bound:

\[
\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{c \in \mathcal{F}} \mathbb{E}\{U^c_{s(c)}(\tau) + X_c(\tau) + Z_c(\tau)\} \leq \frac{B_4}{\delta},
\]

where \( B_4 \triangleq B_3 + VB_R \) and \( B_3 \triangleq B + KN\mu_MT + Nq_MT\mu_M\rho_c + K\rho^2_c\mu^2_M T \). In addition, the algorithm can achieve a throughput

\[
\liminf_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{c \in \mathcal{F}} \mathbb{E}\{\mu^c_{s(c)b(c)}(\tau)\} \geq \sum_{c \in \mathcal{F}} r^*_c, c - \frac{B_3}{V}.
\]

The proof of Theorem 3.1.11 follows the analysis in Section 3.1.3 and is omitted for brevity. On employing delayed queue backlogs, we can extend the centralized optimization problem (3.1.12) to distributed implementations with methods introduced in Remark 3.1.5.

**Arbitrary Link Capacities and Arbitrary Interference Models with Fading Channels**

Recall that in the model description in Section 3.1.2, the link capacity is assumed constant (one packet per slot) and node-exclusive model is employed. In this subsection, we extend the model to arbitrary link capacities and arbitrary interference models with fading channels of finite channel states. Thus, instead of (3.1.4), we have \((\mu^c_{mn}(t))_{(m,n) \in \mathcal{L}} \in I(t)\), where \( I(t) \) is the feasible activation set for time slot \( t \) determined by the underlying interference model and current channel states, with link capacity constraints \( \sum_{c \in \mathcal{F}} \mu^c_{mn}(t) \leq l_{mn} \), where \( l_{mn} \) is the arbitrarily chosen link
capacity for a link \((m, n) \in \mathcal{L}\). We define 
\[ l_n \equiv \max_{(\mu_{mn}(t)) \in I(t)} \sum_{c \in \mathcal{F}} \sum_{m: (m,n) \in \mathcal{L}} \mu_{mn}^c(t) \]. Note that it is clear that 
\[ l_n \leq \sum_{m: (m,n) \in \mathcal{L}} l_{mn} \]. Then we can update the optimization (3.1.12) and weight assignment (3.1.13), respectively, as follows:

\[
\max_{(\mu_{mn}(t)) \in I(t)} \sum_{m,n} \mu_{mn}^c(t) w_{mn}(t)
\]

s.t. \((\mu_{mn}(t))_{(m,n) \in \mathcal{L}} \in I(t)\) and \(\mu_{s(c)b(c)}(t) \leq \mu_M \quad \forall c \in \mathcal{L}\).

\[ w_{mn}(t) = \begin{cases} 
U_{s(c)}^c(t) \frac{U_m^c(t) - U_n^c(t) - l_n}{l_n}, & \text{if } (m, n) \in \mathcal{L}, \\
U_{s(c)}^c(t) \frac{U_m^c(t) - \mu_M - U_{b(c)}^c(t)}{\mu_M}, & \text{if } (m, n) = (s(c), b(c)), \\
0, & \text{otherwise}.
\end{cases} \]

It is not difficult to check that Proposition 3.1.1 still holds with \(q_M \geq \max\{\max_{n \in \mathcal{N}} l_n, \mu_M\}\) and Theorem 3.1.2 holds with a different definition of constant \(B\). The above modified algorithm can be further extended to solve power allocation problems, which is discussed in Section 3.2.

3.1.5 Numerical Results

In this section, we present the simulation results for the optimal algorithm \textit{ALG} proposed in Section 3.1.3. Simulations are run in Matlab 2009A for \(3 \times 10^5\) time slots with results averaged over the last \(10^5\) time slots. In the network topology illustrated in Figure 3.2, there are three source-destination pairs \((A, G)\), \((D, E)\) and \((F, H)\) with same Poisson arrival rates and \(\mu_M = 2\). The required minimum data rate for the three flows are all set to 0.1.
Figure 3.2: Network topology for simulations

Figure 3.3: Virtual queue evolutions with backlogged sources

Table 3.2: Throughput (sum over three flows) and delay (averaged over three flows) performance of $ALG$ with different $V$s for backlogged sources

<table>
<thead>
<tr>
<th>$V$</th>
<th>200</th>
<th>1000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.9202</td>
<td>0.9368</td>
<td>0.9379</td>
</tr>
<tr>
<td>End-to-end delay</td>
<td>46.01</td>
<td>45.76</td>
<td>46.51</td>
</tr>
</tbody>
</table>
Effects of Parameter $V$

We show the algorithm performance with backlogged sources in Table 3.2 by varying the control parameter $V$, where we set $q_M = 5$ and $\rho_c = 30q_M$ for each flow $c$. With an increasing $V$, we observe that the throughput is slightly increased while the average end-to-end delay is not affected.

However, a large value of $V$ increases the upper-bound of the virtual queue backlogs (as stated in Remark 3.1.4 in Section 3.1.3) and has a negative effect on the convergence rates of virtual queues, as is illustrated in Figure 3.3. Specifically, for $V = 10000$, it takes more than $2 \times 10^5$ time slots for $U_{s(c)}^c(t)$ and $Z_c(t)$ to converge. Thus, in the following subsection, we choose $V = 1000$ since a larger $V$ does not lead to a noticeable increase in the throughput while resulting in a smaller convergence rate of virtual queues.

Throughput Optimality and Delay tradeoff

In the following, we illustrate the throughput and delay performance of the proposed algorithm. For comparison, we denote by $BP$ the back-pressure scheduling algorithm in [100] with a congestion controller in [26], and denote by $Finite Buffer$ the cross-layer algorithm developed in [62] with buffer size equal to the queue length limit $q_M$. Note that it is shown in simulation results in [62] that Finite Buffer algorithm ensures much smaller internal queue length (of nodes excluding the source node) than BP algorithm (and queue length is related to delay performance).

We first illustrate in Table 3.3 the throughput optimality of $ALG$ when the sources are constantly backlogged. We loosen the delay constraint as $\rho_c = 30q_M$. With fixed $V = 1000$, as we increase the control parameter $q_M$, the $ALG$ achieves a throughput approaching the throughput of BP algorithm which is known to be
Table 3.3: Throughput (sum for three flows) and delay (averaged over three flows) performance of $ALG$ when sources are backlogged at the transport layer

<table>
<thead>
<tr>
<th></th>
<th>$ALG$ ($\rho_c = 150$)</th>
<th>$ALG$ ($\rho_c = 300$)</th>
<th>$ALG$ ($\rho_c = 3000$)</th>
<th>$ALG$ ($\rho_c = 30000$)</th>
<th>$BP$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>0.9368</td>
<td>1.1834</td>
<td>1.2007</td>
<td>1.2305</td>
<td>1.2315</td>
</tr>
<tr>
<td>End-to-end delay</td>
<td>45.76</td>
<td>131.47</td>
<td>$1.514 \times 10^3$</td>
<td>$1.3687 \times 10^4$</td>
<td>$3.753 \times 10^4$</td>
</tr>
</tbody>
</table>

optimal. We also note that this approximation in throughput results in worse average end-to-end delay performance, which complies with Remark 3.1.3.

We then illustrate the throughput and delay tradeoff for both the $ALG$ and its corresponding suboptimal GMM algorithm (discussed in Remark 3.1.9 in Section 3.1.4) in Figure 3.4 for the case of arbitrary arrival rates at transport layer with $L_M = 0$, where we set $q_M = 5$ and $\rho_c = 50$ for each flow $c$. Note that this pair of $q_M$ and $\rho_c$ shows that the bound in (3.1.15) is actually quite loose, and thus our algorithm can achieve better delay performance than stated in (3.1.15). Figure 3.4 shows that the average end-to-end delay under $ALG$ is well below the constraint ($\rho_c = 50$) and lower than that under BP and Finite Buffer algorithms. The throughput of $ALG$ is close to (although lower than) that of the optimal BP algorithm when arrival rates are small ($\leq 0.3$). Specifically, when the arrival rate is 0.3, $ALG$ achieves a throughput 10% more than the GMM algorithm and 9.0% less than BP algorithm, with an average end-to-end delay 35.2% less than the BP algorithm. In the large-input-rate-region ($> 0.3$), we also observe that the delay in both the BP and Finite Buffer algorithm violates the delay constraints. In addition, in the above illustrated scenarios with backlogged and arbitrary arrival rates, the minimum arrival rates and average end-to-end delay requirements are satisfied for individual flows under $ALG$.

As a side note, the average end-to-end delay in all of the four algorithms in Figure 3.4 first decreases, which can be explained by the intuition that all the algorithms are
based on back-pressure of links (i.e., queue backlog difference of links) and the queue backlog difference tends to be larger for each hop with a larger arrival rate. When arrival rate further increases, congestion level becomes higher since more packets are admitted into the network.

Figure 3.4: Throughput and delay tradeoff under Alg. with performances compared to Finite Buffer algorithm and BP algorithm, with varying arrival rates at the transport layer.

### 3.1.6 Conclusions

In this section, we proposed a cross-layer framework that approaches the optimal throughput arbitrarily close for a general multi-hop wireless network. We show a tradeoff between the throughput and average end-to-end delay bound while satisfying the minimum data rate requirements for individual flows.

Our work aims at a better understanding of the fundamental properties and performance limits of QoS-constrained multi-hop wireless networks. While we show an
$O(\frac{1}{\epsilon})$ delay bound with $\epsilon$-loss in throughput, how small the actual delay can become still remains elusive, which is dependent on specific network topologies. In our future work, we will investigate the capacity region under end-to-end delay constraints applied to different network topologies.

3.2 Power Optimal Control with Finite Buffers

3.2.1 Introduction

With expanding wireless applications and increasing demand for wireless data rates, it is significant to develop power control algorithms that take maximum advantage of available capacity while satisfying certain Quality of Service (QoS) requirements such as minimum data rate and end-to-end delay constraints. Inspired by the back-pressure algorithm [100], optimal power allocation algorithm is further analyzed in [79], with additional congestion controllers considered in [82]. The above referenced works do not deal with delay-related issues or work with finite buffer networks. On the other hand, our throughput-optimal algorithm in [110] guarantees finite buffer size but the link capacity is assumed to be fixed and power allocation is not considered.

In [112], an extension work of [109], we propose two cross-layer algorithms, namely, Power-optimal Scheduling Algorithm (PSA) and Throughput-optimal Scheduling Algorithm (TSA), to minimize energy consumption and to maximize throughput, respectively, in multi-hop wireless networks. Since the objectives of energy consumption minimization and throughput optimization may cause unfairness to individual flows in the network, we place additional per-flow minimum data rate constraints. In addition, we impose maximum link energy consumption constraints in the TSA algorithm to limit energy expenditure. Our algorithms jointly integrate congestion control, power allocation, routing and link rate scheduling. Different from traditional
algorithms which assume infinite buffers, the proposed algorithms deterministically upper-bound the flow-based packet queue length and thus can be employed in multi-hop networks with finite buffers. In addition, the algorithms achieve a power expenditure/throughput “ε-close” to the optimal value, with a tradeoff of order $O\left(\frac{1}{\varepsilon}\right)$ in the buffer size. The average end-to-end delay upper-bound can also be derived from the finite buffer property.

Specifically, we introduce virtual queues to guarantee the data rate constraints and the energy consumption constraints. Both PSA and TSA are composed of a regulator, a congestion controller, a power allocator, and a link rate scheduler. The regulator regulates the virtual queue dynamics, the congestion controller is employed to admit packets from transport layer, while the power allocator determines the power allocation for links in the network, and the link rate scheduler schedules transmission rates for individual flows. Furthermore, we consider adaptive routing scenario, i.e., the routes of each flow are not determined a priori, which is more general than fixed-routing scenario.

To the best of our knowledge, our power allocation algorithms are the first of their kind to achieve an energy consumption/throughput performance at least ε-close to the optimal, with a tradeoff of in the buffer size for individual flows at nodes. The buffer size upper-bound is deterministic, which leads to bounded average end-to-end delay by Little’s Law. Numerical results are also presented to show the performance of the two proposed algorithms with different system parameters.

The rest of the work is organized as follows. Section 3.2.2 provides the network model for multi-hop wireless networks. In Section 3.2.3 and 3.2.4, we propose the optimal power allocation and scheduling algorithms PSA and TSA, respectively, and analyze their performances. The numerical results of the proposed algorithm are provided in Section 3.2.5. Finally, we conclude our work in Section 3.2.6.
3.2.2 Network Model

Network Elements

We consider a multi-hop wireless network consisting of $N$ nodes and $K$ flows. In the network topology, we denote by $\mathcal{F}$ the set of flows, $\mathcal{N}$ the set of nodes, $\mathcal{L}$ the set of directed links in the network, and $(m, n) \in \mathcal{L}$ a link from node $m$ to node $n$. In the network, flows follow routes that are determined adaptively. Additionally, we denote the source node and the destination node of a flow $c \in \mathcal{F}$ as $b(c)$ and $d(c)$, respectively.

A generic cross-layer power control algorithm consists of a congestion controller, a power allocator and link rate scheduler across transport layer and network layer. Packets are generated by specific applications at the transport layer, admitted to the network layer at the source node (by a congestion controller)\(^5\) and transferred from source node to destination node in the network layer (by a power allocator and link rate scheduler). We consider a centralized schedule-based channel, where the channel access is characterized by a power allocation vector and determined by the power allocator.

We consider a time-slotted system with integer-valued time $t \in \{0, 1, 2, \ldots \}$.\(^6\) The centralized power scheduling over the channel is characterized by $\mathbf{P}(t)$, where $\mathbf{P}(t) = (P_{mn}(t))_{(m,n) \in \mathcal{L}}$ represents the power allocation vector for time slot $t$ according to a generic power allocator. We constrain $\mathbf{P}(t)$ to be in $\Pi$, i.e., $\mathbf{P}(t) \in \Pi$, where $\Pi$ is the compact and convex set of feasible power vectors. We also assume that

---

\(^5\)We note that the backlogged source of a flow can be considered an application waiting for packet generation and admission (e.g., a variable rate multimedia encoder). The term “congestion controller” is different than in various TCP versions and corresponds to the packet admission rate to the network as in [79].

\(^6\)Note also that a number of time synchronization methods have been proposed in the literature, e.g., [18, 101, 87], which can be utilized to ensure synchronized operation in the time-slotted system we use in our work.
\( P_{mn}(t) \leq P_M, \forall (m, n) \in \mathcal{L}, \forall \mathbf{P}(t) \in \Pi \), where \( P_M \) is the power upper-bound. In addition, we denote \( \mu_{mn}(\mathbf{P}(t)) \) as the link rate function for link \((m, n)\) corresponding to the power assignment \( \mathbf{P}(t) \), and denote \( \mu(\mathbf{P}(t)) = (\mu_{mn}(\mathbf{P}(t)))_{(m,n)\in\mathcal{L}} \). A wide range of underlying interference models can be characterized by the link rate functions, where the interference caused by simultaneous relaying or transmission in shared wireless channel environment is addressed. In the following, we take the node-exclusive interference model and SNIR model for instance. The node-exclusive interference model [92], which models wireless networks such as Bluetooth networks [75] and FH-CDMA networks [34], can be characterized by \( \mu(\mathbf{P}(t)) \) satisfying, \( \forall (m_1, n_1) \neq (m_2, n_2) \in \mathcal{L} \) such that \( \{m_1, n_1\} \cap \{m_2, n_2\} \neq \emptyset \),

\[
\mu_{m_1n_1}(\mathbf{P}(t))\mu_{m_2n_2}(\mathbf{P}(t)) = 0, \forall \mathbf{P}(t). \tag{3.2.1}
\]

In the node-exclusive interference model [92], a link transmission is only interfered by other simultaneous transmission within a one-hop distance, and equation (1) constrains that a node can involve in at most one communication in one time slot, i.e., simultaneous transmission or relaying within any one-hop distance is avoided. The Signal to Noise and Interference Ratio (SNIR) model, where the link transmission rates are dependent on the interference heard from surrounding simultaneous transmission, can be characterized by \( \mu(\mathbf{P}(t)) \) satisfying:

\[
\mu_{mn}(\mathbf{P}(t)) = f_{mn} \left( \frac{G_{mn}(t)P_{mn}(t)}{B_n + \sum_{(i,j) \neq (m,n)} G_{in}(t)P_{ij}(t)} \right), \tag{3.2.2}
\]

where \( B_n \) is the base noise at node \( n \in \mathcal{N} \), \( G_{mn}(t) \) the propagation gain from the transmitter to the receiver of link \((m, n)\) \( \in \mathcal{L} \) in time slot \( t \), and \( f_{mn}(\cdot) \) the function of SNIR characterizing the link rates associated with the underlying interference model.
We assume the propagation gain process \((G_{mn}(t))\) is ergodic and takes values over a finite state space.

For convenience of analysis, we assume that the link rate functions are upper semi-continuous, and define:

\[
\begin{align*}
l_n &\triangleq \max_{P \in \Pi} \sum_{j : (j,n) \in L} \mu_{jn}(P), \\
f_M &\triangleq \max_{n \in N} l_n, \\
l_M &\triangleq \max_{n \in N} \max_{P \in \Pi} \sum_{i : (n,i) \in L} \mu_{ni}(P),
\end{align*}
\]

(3.2.3)

i.e., \(l_M\) and \(f_M\) are the maximum departure rate from a node and the maximum endogenous arrival rate into a node, respectively.

For a feasible link rate scheduler in time slot \(t\), we let the scheduling parameter \(\mu_{mn}(t)\) be the link rate assignment for flow \(c\) for link \((m,n)\). Thus, given \(P(t)\), we must have \(\sum_{c \in F} \mu_{mn}^c(t) \leq \mu_{mn}(P(t)), \forall (m,n) \in L\).

We assume that the source node for flow \(c\) is always backlogged at the transport layer. For a congestion controller, let \(\mu_{s(c)b(c)}^c(t)\) be the admitted rate of flow \(c\) from the transport layer of flow to the source node, where we can regard \(s(c)\) as the source at the transport layer of flow \(c\). It is clear that in any time slot \(t\), \(\mu_{s(c)b(c)}^c(t) = 0 \forall n \neq b(c)\). We also assume that \(\mu_{s(c)b(c)}^c(t)\) is upper-bounded by a constant \(\mu_M > 0\):

\[
\mu_{s(c)b(c)}^c(t) \leq \mu_M, \forall c \in F, \forall t,
\]

(3.2.4)

i.e., at most \(\mu_M\) packets can be admitted into a source node in any time slot. To simplify the analysis, we prevent looping back to the source, i.e., we impose the
following constraints

\[ \sum_{m \in \mathcal{N}} (\mu_{m, \text{b}(c)}(t)) = 0 \quad \forall c \in \mathcal{F}, \forall t. \quad (3.2.5) \]

In addition, we assume that the network requires each flow \( c \) should transmit at a minimum data rate of \( a_c \) packets per time slot.

**Network Constraints and Approaches**

Network stability and optimality are two necessary goals for the algorithm designs. We first introduce the notion of network stability in this subsection and note that power optimality and throughput optimality will be defined in Section 3.2.3 and Section 3.2.4 where we propose **PSA** and **TSA**, respectively. A given power control algorithm is said to stabilize the network if it stabilizes all actual packet queues. Hence, to represent network stability, we begin with a definition of queue stability with respect to a generic queue backlog \( A(t) \). The queue is *stable* if

\[
\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{A(\tau)\} < \infty.
\]

In addition, we present the following lemma on the stability of queues:

**Lemma 3.2.1.** If a queue \( A(t) \) is stable with some generic service process \( \mu(t) \) and some generic arrival process \( R(t) \) such that the queue dynamics is of the form \( A(t+1) = [A(t) - \mu(t)]^+ + R(t) \), where we define the operator \([x]^+ = \max\{x, 0\}\), then:

\[
\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\mu(\tau)\} \geq \limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{R(\tau)\} \quad (3.2.6)
\]
\[
\liminf_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\mu(\tau)\} \geq \liminf_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{R(\tau)\}
\]

(3.2.7)

**Proof.** From the queue dynamics, we have:

\[
A(t) \geq A(0) - \sum_{\tau=0}^{t-1} \mu(\tau) + \sum_{\tau=0}^{t-1} R(\tau),
\]

which yields:

\[
\frac{\mathbb{E}\{A(t)\}}{t} + \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\mu(\tau)\} \geq \frac{\mathbb{E}\{A(0)\}}{t} + \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{R(\tau)\},
\]

(3.2.8)

\[
\frac{\mathbb{E}\{A(t)\}}{t} - \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{R(\tau)\} \geq \frac{\mathbb{E}\{A(0)\}}{t} - \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\mu(\tau)\}.
\]

(3.2.9)

By taking the limsup of \(t\) on both sides of (3.2.8) and the fact that \(\limsup_{t \to \infty} \frac{\mathbb{E}\{A(t)\}}{t} = 0\) [79], we can prove (3.2.6). Similarly, we can prove (3.2.7) by taking liminf of \(t\) on both sides of (3.2.9).

Now, let \(U^c_n(t)\) be the actual queue backlog of flow \(c\) packets at node \(n\). Then, the network is *stable* if queues \(U^c_n(t)\) are stable, \(\forall n \in \mathcal{N}, \forall c \in \mathcal{F}\).

For convenience of analysis, we define \(\mathcal{L}^c \triangleq \mathcal{L} \cup \{(s(c), b(c))\}\), where the pair \((s(c), b(c))\) can be considered as a virtual link from transport layer to the source node. We now model queue dynamics and network constraints in the multi-hop network. For flow \(c\), if \(n = d(c)\), i.e., \(n\) is the destination node of flow \(c\), then we have \(U^c_n(t) = 0\) \(\forall t\); Otherwise, the queue dynamics are:

\[
U^c_n(t + 1) \leq [U^c_n(t) - \sum_{i: (n,i) \in \mathcal{L}} \mu^c_{ni}(t)]^+ + \sum_{j: (j,n) \in \mathcal{L}^c} \mu^c_{jn}(t), \text{ if } n \in \mathcal{N} \setminus d(c).
\]

(3.2.10)
Note that in (3.2.10), we ensure that the number of packets transmitted for flow \( c \) from node \( n \) does not exceed its corresponding queue backlog, since a feasible scheduling algorithm may be independent of the information on queue backlogs. The terms \( \sum_{i: (n,i) \in \mathcal{L}} \mu_{ni}^c(t) \) and \( \sum_{j: (j,n) \in \mathcal{L}} \mu_{jn}^c(t) \) represent, respectively, the scheduled departure rate from node \( n \) and the scheduled arrival rate into node \( n \) for flow \( c \). Note that (3.2.10) is an inequality since the arrival rates from corresponding neighbor nodes may be less than \( \sum_{j} \mu_{jn}^c(t) \) if some neighbor node does not have enough packets to transmit.

From (3.2.4)(3.2.5), we also have

\[
\sum_{j: (j,b(c)) \in \mathcal{L}} \mu_{b(c)}^j(t) \leq \mu_M, \quad (3.2.11)
\]

if it is guaranteed that no packets will be looped back to the source.

We utilize several types of virtual queues in our two proposed algorithms introduced in Section 3.2.3 and Section 3.2.4. For each flow \( c \), we construct a virtual queue \( U_{s(c)}^c(t) \) at transport layer. We denote by \( R_c(t) \) the virtual input rate to the queue at the end of time slot \( t \), and denote by \( r_c \) the time-average of \( R_c(t) \). We place an upper-bound \( \mu_M \) on \( R_c(t) \) and update the virtual queue as follows:

\[
U_{s(c)}^c(t + 1) = [U_{s(c)}^c(t) - \mu_{s(c)b(c)}^c(t)]^+ + R_c(t), \quad (3.2.12)
\]

where we set \( U_{s(c)}^c(0) = 0 \). Considering the admitted rate \( \mu_{s(c)b(c)}^c(t) \) as the service rate, if the virtual queue \( U_{s(c)}^c(t) \) is stable, then by Lemma 3.2.1 the time-average admitted rate \( \mu_c \) of flow \( c \) satisfies:

\[
\mu_c \triangleq \liminf_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\mu_{s(c)b(c)}^c(\tau)\} \geq r_c \triangleq \liminf_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{R_c(\tau)\}. \quad (3.2.13)
\]
To satisfy the minimum data rate constraints, we construct a virtual queue \( Z_c(t) \) for flow \( c \) with queue dynamics:

\[
Z_c(t + 1) = [Z_c(t) - R_c(t)]^+ + a_c,
\]

where we set \( Z_c(0) = 0 \). If queue \( Z_c(t) \) is stable, we have \( r_c \geq a_c \). Additionally, if \( U_{s(c)}^c(t) \) is stable, then according to (3.2.13), we have \( \mu_c \geq a_c \), i.e., the minimum data rate for flow \( c \) is achieved: 

\[
\liminf_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\mu_{s(c)b(c)}^c(\tau)\} \geq a_c.
\]

The queue evolutions and relationships are illustrated in Figure 3.5. In the figure, for simplicity we do not represent the actual packet queue evolutions for nodes other than source nodes, since the dynamics for actual queues (3.2.10) are dependent on specific network topologies. The decision variable \( R_c(t) \), to be determined by the \( R_c(t) \) regulator in the proposed algorithms, is both the input rate to the virtual queue \( U_{s(c)}^c(t) \) and the service rate of the virtual queue \( Z_c(t) \). The decision variable \( \mu_{s(c)b(c)}^c(t) \), to be determined by the congestion controller in the proposed algorithms, is both the service rate of the virtual queue \( Z_c(t) \) and the input rate to the actual queue at the source \( U_{b(c)}^c(t) \). Thus, the decision of \( R_c(t) \), together with that of \( \mu_{s(c)b(c)}^c(t) \), regulates the queue evolutions and stability of the virtual queues (\( Z_c(t) \) and \( U_{s(c)}^c(t) \)). In the proposed algorithms, after determining the power assignment \( P(t) \) (by a power allocator), we determine (through a link rate scheduler) the decision variables (\( \mu_{mn}^c(t) \)), which, together with the decision of \( \mu_{s(c)b(c)}^c(t) \), regulate the queue evolution and stability of the actual packet queues. We also note that physical packets are only involved in the actual packet queues and the corresponding queue evolutions. Thus, in the following sections, the finite buffer properties refer only to the actual packet queues. By imposing a finite buffer size to actual queues, we can monitor the average flow-based end-to-end delay upper-bounds for the multi-hop network simply
by employing Little’s Law. In the proposed algorithms, the virtual queues $Z_c(t)$ will be used to satisfy the minimum data rate requirements; the virtual queues $U_{s(c)}^c(t)$ are employed as a weight for the differential backlogs across each link, in an attempt to guarantee the finite buffer property and the optimality; the virtual queues $X_{mn}(t)$ will be defined in Section 3.2.4 and are specially employed for TSA algorithms to meet average link energy consumption constraints.

Figure 3.5: Queue relationship diagram with decision variables $R_c(t)$, $\mu_{s(c)b(c)}^c(t)$, $(\mu_{mn}^c(t))$ and $P(t)$, where virtual queues $X_{mn}(t)$ are specially employed for TSA.

In Section 3.2.3 and Section 3.2.4, we introduce PSA and TSA, respectively. We note that the objectives of PSA (minimizing energy expenditure) and TSA (maximizing throughput) cannot be achieved by a single solution, since energy expenditure and throughput tradeoff between each other cannot be optimized at the same time.
While the two algorithms have different objectives and solutions, they share some common features:

- Both algorithms employ virtual queues $Z_c(t)$ to guarantee the minimum data rate constraints and $U^{c}_{s(c)}(t)$ as weight for the differential backlogs when schedule link rates.

- Finite buffer property is satisfied in both algorithms with an $\epsilon$-versus-$\frac{1}{\epsilon}$ tradeoff. Specifically, **PSA/TSA** achieves an energy expenditure/throughput “$\epsilon$-close” to the optimality with a tradeoff in the uniform finite buffer size of order $O(\frac{1}{\epsilon})$.

To assist the development of the following sections, we can define the capacity region $\Lambda$ of the multi-hop network, similar as in [82][79], as the closure of all stabilizable rate vectors considering all power control algorithms choosing $P(t) \in \Pi$. Without loss of generality, we assume that the minimum data rate vector $(a_c)_{c \in \mathcal{C}}$ is within $\Lambda$.

### 3.2.3 Power-Optimal Scheduling Algorithm (PSA) for Multi-Hop Wireless Networks

In this section, we propose a power-optimal scheduling algorithm **PSA** for the introduced multi-hop wireless network so that **PSA** stabilizes the network and satisfies the minimum data rate constraint.

We let $P^*_\epsilon$ denote the minimum sum power for stabilizing rates $(a_c + \epsilon)$, where $\epsilon$ is a positive number which can be chosen arbitrarily small. According to [78][79] we have $\lim_{\epsilon \to 0^+} P^*_\epsilon = P^*$, where $P^*$ is the minimum sum power for stabilizing rates $(a_c)$. Note that $P^*_\epsilon$ can be considered as the $\epsilon$-optimal sum power for the multi-hop wireless network.

Given $\epsilon > 0$, **PSA** is designed to achieve a sum power arbitrarily close to $P^*_\epsilon$. 

77
with a tradeoff with buffer size which will be later given in Theorem 3.2.3 and further explained in Remark 3.2.4.

Let \( q_M \geq \max \{ f_M, \mu_M \} \) be a control parameter standing for buffer size. The optimal algorithm \( PSA \) operates on a time-slot basis consisting of four parts: \( R_c(t) \) regulator, a congestion controller, a power allocator, and a link rate scheduler.

1) \( R_c(t) \) Regulator:

\[
\min_{0 \leq R_c(t) \leq \mu_M} R_c(t)\left( \frac{q_M - \mu_M}{q_M} U_c^{s(c)}(t) - Z_c(t) \right).
\] (3.2.15)

The \( R_c(t) \) regulator controls the virtual queue evolutions of \( U_c^{s(c)}(t) \) and \( Z_c(t) \). Since \( R_c(t) \) is the arrival process for virtual queue \( U_c^{s(c)}(t) \) and the service process for virtual queue \( Z_c(t) \), we assign \( R_c(t) = 0 \) when \( U_c^{s(c)}(t) \) is more congested than \( Z_c(t) \) and assign \( R_c(t) = \mu_M \) otherwise. Specifically, when \( \frac{q_M - \mu_M}{q_M} U_c^{s(c)}(t) - Z_c(t) > 0 \), \( R_c(t) \) is set to zero; Otherwise, \( R_c(t) = \mu_M \).

2) Congestion Controller:

\[
\max_{0 \leq \mu_{s(c)b(c)}(t) \leq \mu_M} \mu_{s(c)b(c)}(t)\left( q_M - \mu_M - U_c^{b(c)}(t) \right).
\] (3.2.16)

The congestion controller aims to upper-bound by \( q_M \) the actual packet queue at source node. Specifically, when \( q_M - \mu_M - U_c^{b(c)}(t) \leq 0 \), \( \mu_{s(c)b(c)}(t) \) is set to zero; Otherwise, \( \mu_{s(c)b(c)}(t) = \mu_M \), where we recall that \( \mu_{s(c)b(c)}(t) \) is the admitted number of packets from transport layer into the source node in time slot \( t \).

3) Power Allocator:

\[
\max_{\mathbf{P}(t) \in \Pi} \sum_{(m,n) \in \mathcal{L}} (\mu_{mn}(\mathbf{P}(t))w_{mn}(t) - VP_{mn}(t)),
\] (3.2.17)
where \( V > 0 \) is a control parameter and \( w_{mn}(t) \) is defined as follows:

\[
w_{mn}(t) \triangleq [\max_{c \in \mathcal{F}} \frac{U^c_c(t)}{q_M}(U^c_m(t) - U^c_n(t) - l_n)]^+.
\]  

(3.2.18)

Note that when \( w_{mn}(t) = 0 \), without loss of optimality we allocate \( \mathbf{P}(t) \) such that \( \mu_{mn}(\mathbf{P}(t)) = 0 \) to maximize (3.2.17). Different from the traditional back-pressure algorithm, for link \((m, n) \in \mathcal{L} \) and flow \( c \in \mathcal{F} \), we add a weight of \( l_n \) to the differential backlog \((U^c_m(t) - U^c_n(t))\) which is further multiplied by \( \frac{U^c_c(t)}{q_M} \). This new type of back-pressure ensures the finite buffer property and the optimality, proven later in Proposition 3.2.2 and Theorem 3.2.3, respectively. In (3.2.17), we can consider \( \mu_{mn}(\mathbf{P}(t))w_{mn}(t) \) as the reward and \( P_{mn}(t) \) as the cost weighted by \( V \), induced from link \((m, n) \) by allocating \( \mathbf{P}(t) \).

Under the node-exclusive interference model (3.2.1), the power allocator is equivalent to the well-known maximal weight matching optimization problem [100], which can be solved in a centralized way. Under the SNIR interference model (3.2.2), with a high SNIR assumption where \( f_{mn}(\cdot) \) can be approximated as log(\cdot) in equation (3.2.2), the optimization can be converted to a nonlinear convex optimization via a log transform [15].

4) Link Rate Scheduler:

\[
\mu^c_{mn}(t) = \begin{cases} 
\mu_{mn}(\mathbf{P}(t)), & \text{if } c = c^*_{mn}(t), \\
0, & \text{otherwise},
\end{cases}
\]  

(3.2.19)

where \( \mathbf{P}(t) \) is determined by the power allocator and \( c^*_{mn}(t) \) is defined as follows:

\[
c^*_{mn}(t) \triangleq \arg \max_{c \in \mathcal{F}} \frac{U^c_c(t)}{q_M}(U^c_m(t) - U^c_n(t) - l_n).
\]

79
For a link \((m, n) \in \mathcal{L}\), the link rate scheduler greedily schedules the available link resource \(\mu_{mn}(P(t))\) to the flow \(c^*_mn(t)\) with the largest back-pressure (3.2.18).

Note that in the power allocator and the link rate scheduler, we constrain that there is no looping back to source.

To analyze the performance of the algorithm, we first introduce the following proposition of finite buffer property.

**Proposition 3.2.2.** Employing PSA, if \(q_M \geq \max\{f_M, \mu_M\}\), then each queue backlog in the network has a deterministic worst-case bound:

\[
U^c_n(t) \leq q_M, \quad \forall t, \forall n \in \mathcal{N}, \forall c \in \mathcal{F}.
\]

**Proof.** We prove Proposition 3.2.2 by induction on time slot. When \(t = 0\), we have \(U^c_n(0) = 0\) \(\forall n, c\). Now suppose in time slot \(t\) we have \(U^c_n(t) \leq q_M\) \(\forall n, c\). In the induction step, for any given node \(n\) and flow \(c\), we consider two cases as follows: (1) We first consider the case when \(n\) is the source node, i.e., when \(n = b(c)\). If \(U^c_{b(c)}(t) \leq q_M - \mu_M\), then according to the queue dynamics (3.2.10)(3.2.11), \(U^c_{b(c)}(t + 1) \leq q_M\); Otherwise, \(U^c_{b(c)}(t) > q_M - \mu_M\) and according to the congestion controller (3.2.16), we have \(\mu^c_{b(c)b(c)}(t) = 0\), so \(U^c_{b(c)}(t + 1) \leq U^c_{b(c)}(t) \leq q_M\) by (3.2.5)(3.2.10). (2) In the second case, \(n\) is not the source node of flow \(c\). If \(U^c_n(t) \leq q_M - l_n\), then \(U^c_n(t) \leq q_M\) by (3.2.10); Otherwise, \(U^c_n(t) > q_M - l_n\), and according to the link rate scheduler (3.2.19) we have \(\mu^c_{mn}(t) = 0\) \(\forall m \in \mathcal{N}\), so \(U^c_n(t + 1) \leq U^c_n(t) \leq q_M\) by the queue dynamics (3.2.10).

Since the above analysis holds for any given \(n\) and \(c\), the induction step holds, i.e., \(U^c_n(t + 1) \leq q_M\) \(\forall n, c\), which completes the proof. \(\square\)

Now we present our main results of the PSA algorithm in Theorem 3.2.3 which is further explained with Remark 3.2.4 and Remark 3.2.7.

80
Theorem 3.2.3. Given $\epsilon > 0$, if

$$q_M > \frac{NL_M^2 + (N - 1)f_M^2 + \mu_M^2 + Nl_Mf_M}{\epsilon} + \mu_M, \quad (3.2.21)$$

then PSA can achieve a time-average power

$$\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau = 0}^{t-1} \sum_{(m,n) \in \mathcal{L}} \mathbb{E}\{P_{mn}(\tau)\} \leq P^*_\epsilon + \frac{B_1}{V}, \quad (3.2.22)$$

where $B_1 \triangleq \frac{1}{2}\mu_M q_M NK + \frac{3q_M - 2\mu_M}{2q_M} K \mu_M^2 + \frac{1}{2} \sum_{c \in F} a_c^2$.

In addition, PSA ensures that the virtual queues have a time-averaged upper-bound:

$$\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau = 0}^{t-1} \sum_{c \in F} \mathbb{E}\{U^c_c(\tau) + Z_c(\tau)\} \leq \frac{B_1 + VP^*_\epsilon}{\delta}, \quad (3.2.23)$$

where $\delta$ is a positive constant satisfying

$$\delta \leq \frac{\epsilon(q_M - \mu_M)}{2q_M} - \frac{NL_M^2 + (N - 1)f_M^2 + \mu_M^2 + Nl_Mf_M}{2q_M}.$$

Proof. The proof for Theorem 3.2.3 is provided in the next subsection. \(\square\)

Remark 3.2.4. The results (3.2.20) and (3.2.23) indicate that PSA stabilizes the network and satisfies the minimum data rate requirement. Specifically, $q_M$ in (3.2.20) can be employed as the uniform buffer size at each node for a single flow. The inequality (3.2.22) gives the upper-bound of the power PSA can achieve. Since the constant $B_1$ is independent of $V$, (3.2.22) also ensures that PSA can achieve a power arbitrarily close to $P^*$. When $\epsilon$ tends to 0, PSA can achieve a power arbitrarily close to the optimal value $P^*$ with the tradeoff in buffer size $q_M$ which is of order $O\left(\frac{1}{\epsilon}\right)$ as shown in (3.2.21). In comparison, in [79], the tradeoff in the average buffer occupancy is
of order $O\left(\frac{V}{\epsilon}\right)$, where a large value of $V$ is required to achieve close to the optimal value. In [62] which aims to achieve optimal throughput-utility, the internal buffer size is of order $O\left(\frac{1}{\epsilon}\right)$, but the buffer size at source nodes is assumed infinitely large, which will result in a large average end-to-end delay. Note also that given buffer size $q_M$, the average end-to-end delay for flow $c \in \mathcal{F}$ can be upper-bounded by $\frac{N q_M}{a_c}$ by Little’s Law.

**Proof of Theorem 3.2.3**

Before we proceed, we present the following lemma which will assist us in proving Theorem 3.2.3.

**Lemma 3.2.5.** For any feasible rate vector $(\theta_c) \in \Lambda$ with $\theta_c \geq a_c \ \forall c \in \mathcal{F}$, there exists a stationary randomized power allocation and scheduling algorithm STAT that stabilizes the network with input rate vector $(\mu^{c,\text{STAT}}_{s(c)b(c)}(t))$, power allocations $(P_{mn}^{\text{STAT}}(t))$ and scheduling parameters $(\mu^{c,\text{STAT}}_{mn}(t))$ independent of queue backlogs, such that the expected admitted rates are:

$$\mathbb{E}\{\mu^{c,\text{STAT}}_{s(c)b(c)}(t)\} = \theta_c, \ \forall t, \ \forall c \in \mathcal{F}.$$ 

In addition, $\forall t, \ \forall n \in \mathcal{N}, \forall c \in \mathcal{F}$, the flow balance constraints are satisfied:

$$\mathbb{E}\left\{\sum_{i: (n,i) \in \mathcal{L}} \mu^{c,\text{STAT}}_{ni}(t) - \sum_{j: (j,n) \in \mathcal{L}} \mu^{c,\text{STAT}}_{jn}(t)\right\} = 0.$$ 

Further, if there exists $\epsilon > 0$ such that $\theta_c = a_c + \epsilon \ \forall c \in \mathcal{F}$, then STAT can be developed to satisfy $\sum_{(m,n) \in \mathcal{L}} \mathbb{E}\{P_{mn}^{\text{STAT}}(t)\} = P^{*}_\epsilon$.

Note that it is not necessary for the randomized algorithm STAT to satisfy the
buffer size constraint (3.2.21). Similar formulations of STAT and their proofs have been given in [82, 79, 26], so we omit the proof of Lemma 3.2.5 for brevity.

**Remark 3.2.6.** Given STAT algorithm in Lemma 3.2.5, we assign the input rates of the virtual queues $U_c^s(t)$ at transport layer as $R_c^{STAT}(t) = \mu_{s(c)\circ (c)}^{c,STAT}(t)$. Thus, we also have $E\{R_c^{STAT}(t)\} = \theta_c$. According to queue dynamics (3.2.12), it is easy to show that the virtual queues under STAT are upper-bounded by $\mu_M$ and the time-average of $R_c^{STAT}(t)$ satisfies: $r_c^{STAT} = \theta_c$. Note that $(\theta_c)$ can take values as $(a_c + \frac{\epsilon}{2})$ or $(a_c + \epsilon)$, where $\epsilon > 0$ such that $(a_c + \epsilon)$ is strictly inside $\Lambda$.

To prove Theorem 3.2.3, we let $Q_1(t) = ((U_n^c(t)), (U_s^c(c)(t)), (Z_c(t)))$ and define the Lyapunov function $L_1(Q_1(t))$ as follows:

$$L_1(Q_1(t)) \triangleq \frac{1}{2} \left\{ \frac{q_M}{q_M} - \mu_M \right\} \sum_{c \in F} U_c^s(c)(t)^2$$

$$+ \sum_{c \in F} Z_c(t)^2 + \sum_{c \in F} \sum_{n \in N} \frac{1}{q_M} U_n^c(t)^2 U_s^c(c)(t) \right\},$$

with $L_1(Q_1(0)) = 0$. We denote the Lyapunov drift by

$$\Delta_1(t) = E\{L_1(Q_1(t+1)) - L_1(Q_1(t))|Q_1(t)\}. \quad (3.2.24)$$

Note that the last term of the Lyapunov function $L_1(Q_1(t))$ takes the same form as
that in [30][62]. From the queue dynamics (3.2.10)(3.2.12), we have:

\[
\sum_{c \in F} \sum_{n \in N} \frac{1}{q_M} U_n^c(t + 1)^2 U_{s(c)}^c(t + 1)
\]

\[
\leq \sum_{c \in F} \frac{1}{q_M} (R_c(t) + U_{s(c)}^c(t)) \sum_{n \in N} U_n^c(t + 1)^2
\]

\[
\leq \mu_M q_M NK + \sum_{c \in F} \frac{1}{q_M} U_{s(c)}^c(t) \sum_{n \in N} \{U_n^c(t)^2
\]

\[
+ \bigg( \sum_{i : (n,i) \in \mathcal{L}} \mu_{ni}^c(t) \bigg)^2 + \bigg( \sum_{j : (j,n) \in \mathcal{L}^c} \mu_{jn}^c(t) \bigg)^2
\]

\[
- 2U_n^c(t)\bigg( \sum_{i} \mu_{ni}^c(t) - \sum_{j} \mu_{jn}^c(t) \bigg) \bigg),
\]

where we recall that \( R_c(t) \leq \mu_M \) and we square both sides of (3.2.10) to deduce the second inequality.

From (3.2.25), we have

\[
\frac{1}{2} \left( \sum_{c \in F} \sum_{n \in N} \frac{1}{q_M} \left( U_n^c(t + 1)^2 U_{s(c)}^c(t + 1) - U_n^c(t)^2 U_{s(c)}^c(t) \right) \right)
\]

\[
\leq \frac{1}{2} \sum_{c \in F} \frac{(Nl_M^2 + (N - 1)f_M + \mu_M^2)U_{s(c)}^c(t)}{q_M}
\]

\[
+ \frac{1}{2} NK q_M \mu_M - \sum_{c \in F} \sum_{n \in N} \frac{U_n^c(t)U_{s(c)}^c(t)}{q_M} \times
\]

\[
\bigg( \sum_{i : (n,i) \in \mathcal{L}} \mu_{ni}^c(t) - \sum_{j : (j,n) \in \mathcal{L}^c} \mu_{jn}^c(t) \bigg),
\]

where we employ (3.2.3)(3.2.11).

By squaring both sides of the queue dynamics (3.2.12)(3.2.14) and employing
(3.2.26), we obtain from the Lyapunov drift (3.2.24):

\[
\Delta_1(t) + V \sum_{(m,n) \in \mathcal{L}} \mathbb{E}\{P_{mn}(t)|Q_1(t)\} 
\leq B_1 + \sum_{c \in \mathcal{F}} a_c Z_c(t) 
\]

\[
+ \sum_{c \in \mathcal{F}} \mathbb{E}\{R_c(t)(\frac{q_M - \mu_M U_{sc}(t) - Z_c(t))}{q_M})|Q_1(t)\} 
\]

\[
+ \frac{1}{2} \sum_{c \in \mathcal{F}} \frac{(Nf_M^2 + (N - 1)f_M^2 + \mu_M^2)U_{sc}(t)}{q_M} 
\]

\[
- \mathbb{E}\{\sum_{c \in \mathcal{F}} U_{sc}(t)\mu_{s(c)b(c)}(t)\frac{q_M - \mu_M}{q_M} 
\]

\[
- \frac{1}{2} \sum_{c \in \mathcal{F}} \sum_{(m,n) \in \mathcal{L}} \mu_{mn}^c(t) \frac{U_{sc}(t)}{q_M} l_n - V \sum_{(m,n) \in \mathcal{L}} P_{mn}(t) 
\]

\[
+ \sum_{c \in \mathcal{F}} \sum_{n \in \mathcal{N}} \left( \sum_{i: (n,i) \in \mathcal{L}} \mu_{ni}^c(t) - \sum_{j: (j,n) \in \mathcal{L}^c} \mu_{jn}^c(t) \right) |Q_1(t)\}.
\]

We can rewrite the last term of RHS of (3.2.27) (the last four lines of (3.2.27)) by simple algebra as

\[
- \sum_{c \in \mathcal{F}} \mathbb{E}\{\mu_{s(c)b(c)}(t)\frac{U_{sc}(t)}{q_M}(q_M - \mu_M - U_{bc}(t))|Q_1(t)\} 
\]

\[
- \mathbb{E}\{\sum_{(m,n) \in \mathcal{L}} \sum_{c \in \mathcal{F}} \mu_{mn}^c(t) \frac{U_{sc}(t)}{q_M} 
\]

\[
\times (U_{m}^c(t) - U_{n}^c(t) - l_n) - VP_{mn}(t)|Q_1(t)\}.
\]

Note that the third term of the RHS of (3.2.27) is minimized by the \(R_c(t)\) regulator (3.2.15), and the last term of the RHS of (3.2.27) is minimized by the combined policy of congestion controller (3.2.16), power allocator (3.2.17) and link rate scheduler.
(3.2.19), over a set of feasible algorithms including the stationary randomized algorithm STAT introduced in Lemma 3.2.5 and Remark 3.2.6. We can substitute into the third term of RHS of (3.2.27) a stationary randomized algorithm with admitted rate vector \((a_c + \frac{\epsilon}{2})\) and into the last term with a stationary randomized algorithm with admitted rate vector \((a_c + \epsilon)\). Thus, we have:

\[
\Delta_1(t) + V \sum_{(m,n) \in \mathcal{L}} \mathbb{E}\{P_{mn}(t)|Q_1(t)\} \\
\leq B_1 + VP^*_\epsilon - \frac{\epsilon}{2} \sum_{c \in \mathcal{F}} Z_c(t) - \sum_{c \in \mathcal{F}} U^c_{s(c)}(t) \times \\
\left( \frac{\epsilon(q_M - \mu_M)}{2q_M} \right) - \frac{Nf^2_M + (N - 1)f^2_M + \mu^2_M + Nl_M f_M}{2q_M},
\]

where we employ the fact \(\sum_{(m,n) \in \mathcal{L}} \mu^c_{mn}(t)l_n \leq Nl_M f_M, \forall c \in \mathcal{F}\). When (3.2.21) holds, we can find \(\delta > 0\) such that \(\delta \leq \frac{\epsilon(q_M - \mu_M)}{2q_M} - \frac{Nf^2_M + (N - 1)f^2_M + \mu^2_M + Nl_M f_M}{2q_M}\). Thus, we have:

\[
\Delta_1(t) + V \sum_{(m,n) \in \mathcal{L}} \mathbb{E}\{P_{mn}(t)|Q_1(t)\} \\
\leq B_1 - \delta \sum_{c \in \mathcal{F}} (U^c_{s(c)}(t) + Z_c(t)) + VP^*_\epsilon.
\]

(3.2.28)

We take the expectation with respect to the distribution of \(Q_1\) on both sides of (3.2.28) and take the time average on \(\tau = 0, \ldots, t - 1\), which leads to

\[
\frac{1}{t} \mathbb{E}\{L_1(Q_1(t))\} + \frac{V}{t} \sum_{\tau=0}^{t-1} \sum_{(m,n) \in \mathcal{L}} \mathbb{E}\{P_{mn}(\tau)\} \\
\leq B_1 + VP^*_\epsilon - \frac{\delta}{t} \sum_{\tau=0}^{t-1} \sum_{c \in \mathcal{F}} \mathbb{E}\{U^c_{s(c)}(\tau) + Z_c(\tau)\}.
\]

(3.2.29)
By taking limsup of $t$ on both sides of (3.2.29), we can prove (3.2.22) and (3.2.23), respectively.

**Remark 3.2.7.** Note that in PSA, the $R_c(t)$ regulator, the congestion controller and the link rate scheduler can operate locally at transport layer, source nodes and links, respectively. To reduce the complexity of the optimization of power allocator (3.2.17), distributed implementation can be developed in much the same way as in [82]. In addition, delayed queue backlogs can be employed similar to the analysis in [62], and our results can be extended to the case where flows have arbitrary arrival rate at transport layer as in [26].

### 3.2.4 Throughput-Optimal Algorithm (TSA) for Multi-Hop Wireless Networks

In this section, we propose a throughput-optimal scheduling algorithm $\text{TSA}$ for the introduced multi-hop wireless network so that $\text{TSA}$ maximizes the network throughput while satisfying the minimum data rate constraint. In addition, each link must meet the average link energy consumption constraint $(\eta_{mn})_{(m,n)\in \mathcal{L}}$, i.e.,

$$p_{mn} \triangleq \lim \sup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{P_{mn}(\tau)\} \leq \eta_{mn}, \forall (m, n) \in \mathcal{L}. \quad (3.2.30)$$

Given $\epsilon > 0$, we define the $\epsilon$-optimal throughput $\mu^*_\epsilon$ as follows

$$\mu^*_\epsilon = \max_{(\mu_c)\in \Lambda} \sum_{c \in \mathcal{F}} \mu_c,$$

s.t.  

$$p_{mn} \leq \eta_{mn} - \epsilon,$$

$$\mu_c \geq a_c + \epsilon.$$
Accordingly, we define the optimal throughput \( \mu^* \) satisfying both the minimum data rate constraints and the link energy consumption constraints as

\[
\mu^* = \max_{(\mu_c) \in \Lambda} \sum_{c \in F} \mu_c,
\]

subject to

\[
\begin{align*}
p_{mn} & \leq \eta_{mn}, \\
\mu_c & \geq a_c.
\end{align*}
\]

Then we have \( \lim_{\epsilon \to 0^+} \mu^*_\epsilon = \mu^* \), since

\[
\mu^*_\epsilon \frac{\epsilon}{\epsilon_M} + \mu^*(1 - \frac{\epsilon}{\epsilon_M}) \leq \mu^*_\epsilon \leq \mu^*,
\]

where

\[
\epsilon_M = \min\left\{ \min_{(m,n) \in \mathcal{L}} \eta_{mn}, \arg\max_{\epsilon > 0} \{ (a_c + \epsilon) \in \Lambda \} \right\}
\]

and the first inequality is derived from the fact that the \( \epsilon \)-optimal throughput \( \mu^*_\epsilon \) is greater than or equal to the throughput achieved by the randomized algorithm that adopts a \( \mu^*_\epsilon \)-throughput achieving algorithm with probability \( \frac{\epsilon}{\epsilon_M} \) and a \( \mu^* \)-throughput achieving algorithm with probability \( 1 - \frac{\epsilon}{\epsilon_M} \).

Similar to the virtual queue \( Z_c(t) \) defined in Section 3.2.2, for each link \((m,n) \in \mathcal{L}\), we define the virtual queue \( X_{mn}(t) \) associated with each link energy consumption constraint \((3.2.30)\), with queue dynamics as follows:

\[
X_{mn}(t + 1) = \left[ X_{mn}(t) - \eta_{mn} \right]^+ + P_{mn}(t).
\]

Thus, if the virtual queue \( X_{mn}(t) \) is stable, by Lemma 3.2.1 the energy consumption constraint associated with link \((m,n) \) is satisfied.
Given $\epsilon > 0$, \textbf{TSA} is designed to achieve a throughput arbitrarily close to $\mu^*$, with a tradeoff in buffer size which will be later shown in Theorem 3.2.8 and further explained in Remark 3.2.9.

With $q_M$ standing for the buffer size, the algorithm \textbf{TSA} also operates on a time-slot basis consisting of four parts: $R_c(t)$ regulator, a congestion controller, a power allocator, and a link rate scheduler.

1) \textbf{R}_c(t) \textbf{ Regulator}:

$$
\min_{0 \leq R_c(t) \leq \mu_M} R_c(t)(\frac{q_M - \mu_M}{q_M} U_s(t) - Z_c(t) - V),
$$

(3.2.31)

where $V$ is the same control parameter as that in \textbf{PSA} algorithm. Specifically, when $\frac{q_M - \mu_M}{q_M} U_s(t) - Z_c(t) - V > 0$, $R_c(t)$ is set to zero; Otherwise, $R_c(t) = \mu_M$.

2) \textbf{Congestion Controller}: The congestion controller is the same as (3.2.16) in the \textbf{PSA} algorithm in Section 3.2.3.

3) \textbf{Power Allocator}:

$$
\max_{P(t) \in \Pi} \sum_{(m,n) \in \mathcal{L}} (\mu_{mn}(P(t))w_{mn}(t) - X_{mn}(t)P_{mn}(t)),
$$

(3.2.32)

where we recall that $w_{mn}(t)$ is defined in (3.2.18). In (3.2.32), we can consider $\mu_{mn}(P(t))w_{mn}(t)$ as the reward and $X_{mn}(t)P_{mn}(t)$ the cost induced from link $(m,n)$ by allocating $P(t)$. Compared to the power allocator in \textbf{PSA}, now the “cost” is weighted by a time-varying virtual queue $X_{mn}(t)$ standing for the link power constraint.

4) \textbf{Link Rate Scheduler}: The link rate scheduler is the same as (3.2.19) in the \textbf{PSA} algorithm in Section 3.2.3.

It is not difficult to check that Proposition 3.2.2 on finite buffer property still holds.
under the TSA algorithm. Now we present our main results of the TSA algorithm in Theorem 3.2.8.

**Theorem 3.2.8.** Given $\epsilon > 0$, if the buffer size $q_M$ satisfies (3.2.21), TSA can achieve a throughput

$$\liminf_{t \to \infty} \frac{1}{t} \sum_{\tau = 0}^{t-1} \sum_{c \in F} \mathbb{E}\{\mu^c_{s(c) \mu(c)}(\tau)\} \geq \mu^* - \frac{B_2}{V},$$

(3.2.33)

where $B_2 \triangleq B_1 + \frac{1}{2} \sum\limits_{(m, n) \in \mathcal{L}} \eta_{mn}^2 + \frac{1}{2} |\mathcal{L}| P_M^2$. In addition, TSA ensures that the virtual queues are upper-bounded:

$$\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau = 0}^{t-1} \mathbb{E}\left\{\sum_{c \in F} (U^c_{s(c)}(\tau) + Z_c(\tau)) + \sum_{(m, n) \in \mathcal{L}} X_{mn}(t)\right\} \leq B_2 + V(\mu^* - \mu^*_\epsilon) \frac{1}{\delta},$$

(3.2.34)

where we recall $\delta$ is a positive constant defined in Theorem 3.2.3.

**Proof.** The proof for Theorem 3.2.8 is provided in the next subsection.

**Remark 3.2.9.** The inequality (3.2.34) indicates that TSA satisfies the minimum data rate requirements and the link consumption constraints. Since the constant $B_2$ is independent of $V$, (3.2.33) ensures that TSA can achieve a throughput arbitrarily close to $\mu^*$ when $\epsilon$ is small enough and $V$ is large enough, with the tradeoff in buffer size $q_M$ of order $O\left(\frac{1}{\epsilon}\right)$ as shown in (3.2.21). By Little’s Law, the average end-to-end delay for flow $c \in F$ is upper-bounded by $\frac{Nq_M}{a_c}$. 

90
Proof of Theorem 3.2.8

Similar to the proof of Theorem 3.2.3, we first construct a set of stationary randomized algorithms in the following lemma:

**Lemma 3.2.10.** Given some small enough $\epsilon > 0$, there exists a stationary randomized power allocations and scheduling algorithm $\text{STAT}^*\epsilon$ that stabilizes the network with input rate vector $(\mu_{s(c)b(c)}^\text{STAT}^*\epsilon(t))$, power allocation $(P_{mn}^\text{STAT}^*\epsilon(t))$ and scheduling parameters $(\mu_{mn}^\text{STAT}^*\epsilon(t))$ independent of queue backlogs, such that the expected admitted rates are:

$$\sum_{c \in F} \mathbb{E}\left\{ \mu_{s(c)b(c)}^\text{STAT}^*\epsilon(t) \right\} = \mu^*\epsilon, \forall t.$$ 

In addition, $\forall t, \forall n \in N, \forall c \in F$, the flow balance constraints are satisfied:

$$\mathbb{E}\left\{ \sum_{i: (n,i) \in L} \mu_{ni}^\text{STAT}^*\epsilon(t) - \sum_{j: (j,n) \in L^c} \mu_{jn}^\text{STAT}^*\epsilon(t) \right\} = 0.$$

Further, the minimum data rate constraints and the energy consumption constraints are satisfied:

$$\mathbb{E}\{ \mu_{s(c)b(c)}^\text{STAT}^*\epsilon(t) \} \geq a_c + \epsilon, \forall t, \forall c \in F,$$

$$\mathbb{E}\{ P_{mn}^\text{STAT}^*\epsilon(t) \} \leq \eta_{mn} - \epsilon, (m, n) \in L.$$

For brevity, we omit the proof of Lemma 3.2.10, and interested readers are referred to [82, 79, 26] for more details. In $\text{STAT}^*\epsilon$, we take the virtual input rate to $U_{s(c)}(t)$ as $P_{c}^\text{STAT}^*\epsilon(t) = \mu_{s(c)b(c)}^\text{STAT}^*\epsilon$. We let $Q_2(t) = ((U_n^c(t)), (U_{s(c)}^c(t)), (Z_c(t)), (X_{mn}(t)))$ and define the Lyapunov...
function \( L_2(Q_2(t)) \) as follows:

\[
L_2(Q_2(t)) \triangleq L_1(Q_1(t)) + \frac{1}{2} \sum_{(m,n) \in L} X_{mn}(t)^2,
\]

with \( L_2(Q_2(0)) = 0 \). We denote the Lyapunov drift by

\[
\Delta_2(t) = \mathbb{E}\{L_2(Q_2(t + 1)) - L_2(Q_2(t))|Q_2(t)\}.
\]

Following the analysis in deriving (3.2.27), we obtain the inequality on the Lyapunov drift as follows:

\[
\Delta_2(t) - V \sum_{c \in \mathcal{F}} \mathbb{E}\{R_c(t)|Q_2(t)\} \\
\leq B_2 + \sum_{c \in \mathcal{F}} a_c Z_c(t) - \sum_{(m,n) \in L} \eta_{mn}X_{mn}(t) \\
+ \sum_{c \in \mathcal{F}} \mathbb{E}\{R_c(t)(\frac{q_M - \mu_M}{q_M} U_{s(c)}^c(t) - Z_c(t) - V)|Q_2(t)\} \\
+ \frac{1}{2} \sum_{c \in \mathcal{F}} \frac{(Nf_M^2 + (N - 1)f_M^2 + \mu_M^2) U_{s(c)}^c(t)}{q_M} \\
- \sum_{c \in \mathcal{F}} \mathbb{E}\{\mu_{s(c)b(c)}^c(t) \frac{U_{s(c)}^c(t)}{q_M} \times (q_M - \mu_M - U_{b(c)}^c(t))|Q_2(t)\} \\
- \mathbb{E}\{ \sum_{(m,n) \in L} \sum_{c \in \mathcal{F}} \mu_{mn}^c(t) \frac{U_{s(c)}^c(t)}{q_M} \times (U_m^c(t) - U_n^c(t) - l_n) - X_{mn}(t)P_{mn}(t)|Q_2(t)\}.
\]

Note that the fourth term of the RHS of (3.2.35) is minimized by the \( R_c(t) \) regulator (3.2.31), the sixth term minimized by the congestion controller, and the last term minimized by the combined policy of power allocator (3.2.32) and link rate scheduler,
over a set of feasible algorithms including the stationary randomized algorithms introduced in Lemma 3.2.10. We can substitute into the fourth term of RHS of (3.2.35) a stationary randomized algorithm $STAT^*_\epsilon/2$ and into the last two terms a stationary randomized algorithm $STAT^*_\epsilon$. Thus, we have:

$$\Delta_2(t) - V \sum_{c \in F} \mathbb{E}\{R_c(t)|Q_2(t)\} \leq B_2 - V \mu^*_\epsilon/2 - \delta[\sum_{c \in F} (U_{s(c)}^c(t) + Z_c(t)) + \sum_{(m,n) \in \mathcal{L}} X_{mn}(t)].$$

By taking the expectation with respect to the distribution of $Q_2$ on both sides of the above inequality and taking the time average on $\tau = 0, ..., t - 1$, we can prove (3.2.34) with limsup of $t$ and (3.2.33) with liminf of $t$, respectively.

3.2.5 Numerical Results

![Network topology for simulations](image-url)
In this section, a simulation-based performance evaluation of our proposed algorithms is presented. Consider the network topology in Figure 3.6 where each link is bidirectional. For simplicity, we assume the noise for any one-hop transmission is constant and normalized to 1. In addition, we set the maximum number of admissible packets as $\mu_M = 4$.

In our simulation setup in Figure 3.6, there are two flows with source-destination pairs $(A, H)$ and $(D, E)$ and the required minimum data rate $a_c$ for the two flows are identical for simplicity. Simulations are run in Matlab 2009A over $10^5$ time slots. Specifically, we illustrate the virtual queue dynamics and the performance for **PSA** and **TSA**.

We first employ node-exclusive model (1) in our simulation. With the employment of quadrature amplitude modulation (QAM) schemes [2], the link rate function $\mu_{m_1n_1}(P(t))$, $(m_1, n_1) \in L$, is defined (in unit of packet/time slot) as follows:

$$\mu_{m_1n_1}(P(t)) = \begin{cases} 
1, & \text{when } 0.25 \leq P_{m_1n_1}(t) < 0.5, \\
2, & \text{when } 0.5 \leq P_{m_1n_1}(t) < 1.25, \\
4, & \text{when } P_{m_1n_1}(t) \geq 1.25, 
\end{cases}$$

if $P_{m_2n_2}(t) = 0, \forall (m_2, n_2) \in L$ s.t. $\{m_1, n_1\} \cap \{m_2, n_2\} \neq \emptyset$. Otherwise, $\mu_{m_1n_1}(P(t)) = 0$. In the last simulation, we present the results for **PSA** under SNIR interference model (3.2.2) with time-varying propagation gains.

**Control Parameter $V$ and Virtual Queue Evolutions**

We first present the virtual queue evolutions with different choices of parameter $V$ for the two algorithms, where we recall that $V$ is the control parameter in the power allocator (3.2.17) of **PSA** and the $R_c(t)$ regulator (3.2.31) of **TSA**, which tradeoffs between the optimality and the upper-bounds of virtual queues as shown in Theorem
3.2.3 and Theorem 3.2.8, respectively. Figure 3.7 shows the virtual queue evolutions of per-flow $Z_c(t)$ and $U^c_{i(c)}(t)$ for $PSA$ and Figure 3.8 the virtual queue evolutions of per-flow $Z_c(t)$, $U^c_{i(c)}(t)$ and per-link $X_{mn}(t)$.

By increasing $V$, we can approach the optimality, which is both illustrated in Theorem 3.2.3 and 3.2.8 and exemplified in the next subsection with $PSA$. However, the convergence time becomes large for all virtual queues with a large $V$ as shown in both Figure 3.7 and 3.8. For instance, when $V = 500$, it can take up to 10000 time slots for $Z_c(t)$ to converge for $TSA$ algorithm. However, we will show in the next subsection that $V$ does not have to be large to approach the optimal value.

![Figure 3.7: Virtual queue dynamics for PSA algorithm](image)

Performance of $PSA$ Algorithm under Node-Exclusive Interference Model

In Figure 3.9, with fixed buffer size $q_M = 100$, we illustrate the energy consumption by increasing minimum data rate requirement $a_c$ for different values of $V$. By fixing the control parameter $V$, the energy consumption increases for $PSA$ as $a_c$ increases; By fixing $a_c$, the energy consumption decreases as $V$ increases (at the expense of convergence time as explained in Section 3.2.5). Figure 3.9 also shows that $V = 100$
is sufficiently large such that almost no power gain is observed by further increasing the value of $V$. We also note that through simulation, $V$ does not have a direct effect on delay/congestion performance of the algorithm. We will show in the following that the value of $q_M$ has a large impact on the congestion level of the network.

Figure 3.9: Performance for PSA with fixed buffer size $q_M = 100$
Table 3.4: Performance of \textit{PSA} when minimum data rate is set as $a_c = 0.2$

<table>
<thead>
<tr>
<th>Sum power</th>
<th>$PSA (q_M = 20)$</th>
<th>$PSA (q_M = 50)$</th>
<th>$PSA (q_M = 100)$</th>
<th>$PSA (q_M = 200)$</th>
<th>EECA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3.5336</td>
<td>3.2915</td>
<td>3.2696</td>
<td>3.2180</td>
<td>3.2145</td>
</tr>
<tr>
<td>No. packets in queues</td>
<td>102.59</td>
<td>283.78</td>
<td>584.50</td>
<td>1.1854 × 10^3</td>
<td>1.2665 × 10^3</td>
</tr>
</tbody>
</table>

**Performance of TSA Algorithm under Node-Exclusive Interference Model**

In Figure 3.10, with fixed minimum data rate constraint $a_c = 0.6$ and fixed control parameter $V = 100$, we plot the throughput, congestion level (represented by the total number of packets in the network) and energy consumption of \textit{TSA} against buffer size $q_M$ with different values of link energy consumption constraint, where we uniformly set $\eta_{mn} = \eta, \forall (m, n) \in L$, for simplicity. By fixing buffer size $q_M$ and increasing the maximum allowed link power $\eta$, \textit{TSA} improves the throughput at the expense of power expenditure where we note the congestion level (represented by the total number of packets in the network in Figure 3.10) is hardly affected. By fixing $\eta$ and increasing buffer size $q_M$, \textit{TSA} improves the throughput at the expense of power expenditure and an almost linear increase in the congestion level. Given $\eta$, Figure 3.10 also shows that throughput and energy consumption behave similarly and converge when $q_M$ is large enough. Specifically, Figure 3.10 illustrates that $q_M = 100$ is sufficiently large such that there is hardly any gain in throughput with a larger $q_M$ while we note that the congestion level increases almost linearly with $q_M$.

**Performance of PSA Under SNIR Interference Model**

In the following, with the same system setup and the same topology in Figure 3.6, we employ \textit{PSA} under the SNIR interference model (3.2.2). We assume the distance between the two communication nodes of each link is normalized to 1. We setup the simulation parameters in (3.2.2) with reference to [82][15][86][94]. Specifically, we set
Figure 3.10: Performance for TSA with fixed minimum data rate requirement $a_c = 0.6$ and control parameter $V = 100$

$$f_{mn}(\cdot) = \log(\cdot)$$ and the i.i.d. time-varying propagation gains in (3.2.2), $\forall t$, as

$$G_{mn}(t) = \begin{cases} d_{mn}^{-4}, & \text{w.p. } \frac{2}{3}, \\ \frac{1}{3}d_{mn}^{-4}, & \text{w.p. } \frac{1}{3}, \end{cases}$$

where $d_{mn}$ denotes the Euclidean distance between any two nodes $m$ and $n$. We set the power upper-bound $P_M = 10$ with respect to the normalized base noise (i.e., $B_n = 1$), and assume the transmission power of each communication link can be tuned to a number of levels in the set $\{0, \frac{1}{4}P_M, \frac{1}{2}P_M, \frac{3}{4}P_M, P_M\}$.

In Table 3.4, we present the performance of our proposed finite-buffer algorithm \textbf{PSA}, and compare to the result of EECA algorithm proposed in [79] which is proved to achieve optimal power while stabilizing any rate vector inside the capacity region. Since we also require minimum rates for individual flows, an additional virtual queue is employed in the congestion controller in EECA according to [26]. We set the control parameter $V = 10$ in \textbf{PSA}, since a larger $V$ cannot further minimize energy consumption but may increase buffer occupancy while a smaller $V$ will lead to worse
power performance. Table 3.4 illustrates that, with the consumed energy in \textit{PSA} slightly larger than that of EECA, the queue backlog in the network is far smaller than that of EECA, which results in better end-to-end delay performance at a small energy cost. As we increase \( q_M \), \textit{PSA} approaches the optimal sum power which is the simulation value by EECA. In addition, the minimum data rate requirement and finite buffer property are achieved under \textit{PSA} in the simulation.

### 3.2.6 Conclusions

In Section 3.2, we proposed two cross-layer algorithms to minimize energy consumption and maximize throughput, respectively, for multi-hop wireless networks with finite buffers. Our work aims at a better understanding of the fundamental properties and performance limits of dynamic power allocation and scheduling in multi-hop wireless networks. We showed a tradeoff between \( O\left(\frac{1}{\epsilon}\right) \) in the finite buffer size and the \( \epsilon \)-characterized proximity to the optimal power/throughput. Our future work will involve investigation on short-lived flows and distributed implementations of back-pressure-based algorithms with random access techniques.
CHAPTER 4
DISTRIBUTED UTILITY-OPTIMAL SCHEDULING
WITH FINITE BUFFERS

4.1 Introduction

While we focus on centralized algorithms in the works [110][109][112], we propose a distributed cross-layer scheduling algorithm in [117] for networks with single-hop transmissions that can guarantee finite buffer sizes and meet minimum utility requirements. The algorithm can achieve a utility arbitrarily close to the optimal value with a tradeoff in the buffer sizes. The finite buffer property is not only important from an implementation perspective, but, along with the algorithm, also yields superior delay performance. A novel structure of Lyapunov function, first introduced in [110], is employed to prove the utility optimality of the algorithm with the introduction of novel virtual queue structures. Unlike traditional back-pressure-based optimal algorithms, our proposed algorithm does not need centralized computation and achieves fully local implementation without global message passing. Compared to other recent throughput/utility-optimal CSMA distributed algorithms, we illustrate through rigorous numerical and implementation results that our proposed algorithm achieves far better delay performance for comparable throughput/utility levels.

In the algorithm development, for each communication link $l$ in the network, we maintain an actual packet queue $U_l$ and construct two virtual queues: a virtual service
queue $Z_l$ to guarantee the minimum utility requirement and a virtual queue $Q_l$ that acts as a weight on the actual queue $U_l$ when scheduling link rates. The cross-layer algorithm is composed of three parts: a regulator that regulates the evolution of virtual queues, a congestion controller that regulates data packet admission from the packet generator, and a link rate scheduler employing the Glauber dynamics method [83, 28, 48]. Moreover, we make use of the time-scale separation assumption (the underlying Markov chain of the scheduler converges instantaneously to its steady state distribution compared to link weight adaptation rate of the scheduler) employed in [51][83], which is justified in [84][50].

On the other hand, random access algorithms employing the Glauber dynamics [83, 85, 28] only use a function of actual queue backlog $U_l$ as a weight in the link transmission probability. Specifically, the link weights in the Glauber dynamics are selected in the form of $U_l$ in [48], $\log U_l$ in [83], $\log \log U_l$ in [85], $\frac{\log U_l}{y(U_l)}$ in [28], where $y(\cdot)$ is a function that increases arbitrarily slowly. The resulting schedulers of the Glauber dynamics react to changes in $U_l$. However, the queue lengths must be large enough to approach maximal weight matching [83][28], or to ensure a negative Lyapunov drift [85][48]. In other words, the network stability is achieved at the expense of large values of $U_l$, which leads to large average packet delays. Different from those algorithms, the link rate scheduler of our algorithm assigns the random access probabilities of a link $l$ as the product of the packet queue backlog $U_l$ and the virtual queue backlog $Q_l$. Consequently, the network stability is achieved by shifting the burden from actual packet queues to our proposed virtual queues $Q_l$, while the actual queues $U_l$ are upper-bounded by a finite buffer size. As a direct result, the delay performance of the algorithm is significantly improved. The stability and utility optimality of the algorithm are proved through Lyapunov optimization techniques. Different from the traditional Lyapunov optimization techniques using quadratic Lyapunov functions,
we employ a novel type of Lyapunov function by multiplying the virtual queue backlog $Q_l$ to the quadratic term of actual queue backlogs $U_l^2$. We show that this particular structure of Lyapunov function leads to the product form of virtual and actual queue backlogs in the proposed algorithm.

Salient properties of our algorithm can be listed as follows: (1) All link-based queues in the network use finite-sized buffers, which leads to network stability; (2) Minimum utility requirements are satisfied for individual communication links; (3) A utility “$\epsilon$-close” to the optimal network utility (expressed as the sum of link utilities, given minimum data rate requirement of individual links) is achieved with a tradeoff of $O(\frac{1}{\epsilon})$ in the buffer size; (4) The link rate scheduler of our algorithm uses the CS-MA paradigm with RTS/CTS handshake, which admits distributed implementation without global message passing.

The algorithm is shown both in simulation and implementation to have close-to-optimal throughput proved theoretically and display very favorable delay performances. Our main algorithm is further extended: (1) to construct individual upper-bounds for average link-based delay; (2) to approximate the time-scale separation, which lies at the heart of our algorithm as well as those introduced in [51] [83].

The rest of the section is organized as follows: In Section 4.2, we present the network model, followed by approaches considered for wireless networks with single-hop transmissions. In Section 4.3, the distributed utility-optimal algorithm is described and its performance analyzed. We provide a modified algorithm to construct per-flow average delay upper-bounds in Section 4.4. Two methods to approach time-scale separation are introduced in Section 4.5. We present implementation and numerical results in Sections 4.6 and 4.7, respectively. Finally, we conclude our work in Section 4.8.
4.2 Network Model

4.2.1 Network Elements

We consider a time-slotted wireless network with single-hop transmissions consisting of \( N \) nodes and \( L \) directional links, with the node and the link sets denoted as \( \mathcal{N} \) and \( \mathcal{L} \), respectively. Each directional link \( l = (m, n) \in \mathcal{L} \) carries a single-hop flow from a source node \( m \) to a destination node \( n \), with \( m, n \in \mathcal{N} \).

Let \( G = (\mathcal{L}, \mathcal{E}) \) be the interference graph (also called conflict graph) associated with the network topology. There is an edge \((l, j) \in \mathcal{E}\) if simultaneous transmissions over links \( l \) and \( j \) are not possible. We denote the set of interfering links of link \( l \in \mathcal{L} \) as \( \mathcal{N}(l) \triangleq \{ j : (l, j) \in \mathcal{E} \} \cup \{ l \} \), where we let \( l \in \mathcal{N}(l) \) for analytical clarity.

For simplicity of analysis, in each time slot, each link’s packet transmission rate is normalized and can take a fractional value in \([0, 1]\). Thus, a feasible schedule at time slot \( t \) can be represented by a vector \( \mu(t) = (\mu_l(t))_{l \in \mathcal{L}} \in [0,1]^L \) such that:

\[
\mu_l(t)\mu_j(t) = 0, \quad \forall l \in \mathcal{L} \quad \text{and} \quad \forall j \in \mathcal{N}(l) \setminus \{l\},
\]

where \( \mu_l(t) \) is the scheduled link rate for link \( l \in \mathcal{L} \) at time slot \( t \). For each feasible schedule \( \mu(t) \), we can find a corresponding link set \( x \subset \mathcal{L} \), called an independent set, such that \( l \in x \text{ iff } \mu_l(t) > 0 \). Let \( \mathcal{I} \) be the set of all independent sets in the network. Note that for a link \( l = (m, n) \in \mathcal{L} \) connecting \( m \) to \( n \), with \( m, n \in \mathcal{N} \), its scheduled link rate can also be denoted as \( \mu_{mn}(t) = \mu_l(t) \).

We denote by \( f_l(\cdot) \), \( l \in \mathcal{L} \), the utility functions of the time-average data transmission rate. As convention, we assume the utility functions are positively-valued, concave, continuously differentiable and strictly increasing, with \( f_l(0) = 0, \ l \in \mathcal{L} \). Let \( d_l \geq 0 \) be the minimum utility constraint associated with link \( l \in \mathcal{L} \), i.e., the utility of the time-average data rate over link \( l \) must be greater than or equal to \( d_l \).

According to [100][26], we define the capacity region \( \Lambda \) of the considered network.
as the closure of all feasible admitted rate vectors \((a_l)_{l \in \mathcal{L}}\) each stabilizable by some scheduling algorithm. The capacity region \(\Lambda\) is convex, closed, and bounded [26]. Without loss of generality, we assume that \((f_l^{-1}(d_l))_{l \in \mathcal{L}}\) is inside \(\Lambda\), i.e., there exists \(\epsilon > 0\) with \((f_l^{-1}(d_l) + \epsilon)_{l \in \mathcal{L}} \in \Lambda\). To assist the analysis in the following sections, we let \((a^*_l)_{l \in \mathcal{L}}\) denote a solution to the following optimization problem:

\[
(a^*_l)_{l \in \mathcal{L}} \in \arg \max_{(a_l)} \sum_{l \in \mathcal{L}} f_l(a_l) \\
\text{s.t. } a_l \geq f_l^{-1}(d_l) \text{ and } (a_l + \epsilon)_{l \in \mathcal{L}} \in \Lambda.
\]

(4.2.1)

Note that \(\sum_{l \in \mathcal{L}} f_l(a^*_l)\) can be regarded as the overall utility \(\epsilon\)-close to the optimality. Then, according to [78] we have

\[
\lim_{\epsilon \to 0^+} \sum_{l \in \mathcal{L}} f_l(a^*_l, \epsilon) = \sum_{l \in \mathcal{L}} f_l(a^*_l),
\]

where \((a^*_l)_{l \in \mathcal{L}}\) is a solution to the following optimization:

\[
(a^*_l)_{l \in \mathcal{L}} \in \arg \max_{(a_l)} \sum_{l \in \mathcal{L}} f_l(a_l) \\
\text{s.t. } a_l \geq f_l^{-1}(d_l) \text{ and } (a_l)_{l \in \mathcal{L}} \in \Lambda.
\]

(4.2.2)

In most practical systems, congestion control is needed to avoid overloading the networks. Motivated by this, we consider a closed-loop system, i.e., the admission/generation of packets are determined by the scheduling algorithm. Thus, the packet delay is defined to be the period that starts when the packet gets admitted into the network and ends when it gets served. In addition, we assume the physical packet generator of each link’s source is constantly backlogged. Thus, a congestion controller is needed to determine the rate of packet generation/admission into the
network.\(^1\) Let \(A_l(t)\) denote the admitted rate of link \(l\) from the packet generator in time slot \(t\). Since the link’s packet transmission capacity is normalized to 1, we assume that the admitted rate \(A_l(t)\) is also normalized with respect to the link capacity. For the congestion controller, we also assume that \(A_l(t)\) is bounded above by some constant \(\mu_M > 0, \forall l \in \mathcal{L} \forall t\), i.e., the source node of a link can receive at most \(\mu_M\) packets from the packet generator in any time slot.\(^2\)

### 4.2.2 Network Constraints and Approaches

To present network stability, we begin with a definition of queue stability with respect to a generic queue backlog \(X(t)\): the queue is stable if 
\[
\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{X(\tau)\} < \infty.
\]

Let \(U_l(t)\) be the actual packet queue backlog of link \(l\), maintained at the source node of the link. Then, the network is stable if queues \(U_l(t)\) are stable, \(\forall l \in \mathcal{L}\). The queue dynamics have the following evolution:

\[
U_l(t+1) = [U_l(t) - \mu_l(t)]^+ + A_l(t), \forall l \in \mathcal{L},
\]

where we recall \(\mu_l(t) \in [0, 1]\) is the scheduled link rate in time slot \(t\) and we define the operator \([\cdot]^+\) as \([\cdot]^+ \triangleq \max\{\cdot, 0\}\). Note that the data transmitted for a link \(l\) cannot exceed its backlog and a feasible scheduling algorithm may be independent of queue backlog information. For simplicity of analysis, we assume the input rates \(A_l(t)\) will be added to the queue backlogs at the end of time slot \(t\).

For each link \(l\), we construct a virtual queue \(Q_l(t)\) at link \(l\)'s source node to later

---

1\(^\text{Note that the constantly backlogged packet generator is not necessarily a physical queue. It can represent an application controlling its packet generation, e.g., a variable rate multimedia encoder. After packets are generated and admitted to the source node, they are delivered from the source to the destination node via the network layer.}\)

2\(^\text{To be specific, we require } \mu_M \leq q_M \text{ to guarantee the finite buffer sizes in Proposition 4.3.1, where } q_M \text{ denotes the uniform buffer size introduced in Section 4.3.}\)
assist the development of our proposed algorithm in the next section. We denote by
$R_l(t)$ the virtual input rate to queue $Q_l(t)$ at the end of time slot $t$, and denote by $r_l$ the time-average of $R_l(t)$. We place an upper-bound $v_M$ on $R_l(t)$ with
\[
\max_{(a_l)_{l \in L} \in \Lambda} \max_{l \in L} a_l \leq v_M \leq \mu_M,
\]
and update queue $Q_l(t)$ virtually as follows:
\[
Q_l(t + 1) = [Q_l(t) - A_l(t)]^+ + R_l(t), \tag{4.2.4}
\]
with $Q_l(0) = 0$. Considering the admitted rate $A_l(t)$ as the service rate of the virtual queue, if $Q_l(t)$ is stable, then the time-average admitted rate $a_l$ of link $l$ satisfies:
\[
a_l \triangleq \liminf_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{A_l(\tau)\} \geq r_l \triangleq \liminf_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{R_l(\tau)\}. \tag{4.2.5}
\]
We construct a virtual service queue $Z_l(t)$ at the source node of each $l \in L$ with its virtual queue dynamics given by:
\[
Z_l(t + 1) = [Z_l(t) - R_l(t)]^+ + f_l^{-1}(d_l), \tag{4.2.6}
\]
with $Z_l(0) = 0$. When $Z_l(t)$ and $Q_l(t)$ are stable, we have $f_l(a_l) \geq f_l(r_l) \geq d_l$, i.e., the minimum utility constraint is satisfied.

The queue relationships between queues $Z_l(t)$, $Q_l(t)$ and $U_l(t)$ are shown in Figure 4.1, where we note that $A_l(t)$ represents both the physical packet admission rate (i.e., the arrival rate to actual queue $U_l(t)$) and the virtual service rate of $Q_l(t)$. Physical packets are only involved in the “actual part” block. We note that the average packet delay is only affected by the number of packets in the actual packet queue.
The size of virtual queues have no effect on the delay as they only determine rate of packet arrivals. Even if a virtual queue has a large number of virtual packets, they are not propagated in the network through an actual packet queue and actual packets never passed through virtual queues.

In addition, the reason we construct two virtual queues is that the utility optimality and minimum utility constraint cannot be both guaranteed with the employment of one single virtual queue. Specifically, for link \( l \in \mathcal{L} \), virtual queue \( Z_l(t) \) monitors the achievement of minimum utility constraint and virtual queue \( Q_l(t) \) acts as a weight for actual packet queue \( U_l(t) \) in the link rate scheduler (introduced in the next section) that approaches a maximal weight matching in an aim to achieve optimal utility. For instance, in Section 4.4, only one (modified) virtual queue \( Z_l(t) \) is employed in a modified algorithm to construct average delay upper-bounds at the expense of losing utility optimality.

### 4.3 The Proposed Algorithm for Single-Hop Networks

In this section, employing the queue structures developed in Section 4.2.2, we propose a distributed cross-layer scheduling algorithm \( \text{ALG} \) for the wireless network model introduced in Section 4.2 such that \( \text{ALG} \) can stabilize the network and satisfy the minimum utility constraints. Moreover, \( \text{ALG} \) achieves a utility arbitrarily close to the optimal value \( \sum_{i \in \mathcal{L}} f_i(q_i^*) \) under certain conditions related to queue buffer sizes which will be later given in Theorem 4.3.4.

Let \( q_M \geq \mu_M \) be a control parameter indicating the uniform buffer size for \( U_l(t) \) at each link \( l \in \mathcal{L} \), where we recall \( \mu_M \) limits the maximal rate of packet arrival to a source node. \( \text{ALG} \) ensures that all queue backlogs are bounded by \( q_M \), which further leads to a bounded average packet delay according to Little’s Law. We construct the \( \text{ALG} \) with the following three functions: \( R_l(t) \) regulator, a congestion
controller and a link rate scheduler, which decide on the arrival and service rates of all queues ∀t (as shown in Figure 4.1).

1) $R_l(t)$ Regulator:

$$\min_{0 \leq R_l(t) \leq v_M} g_l(R_l(t)), \quad (4.3.1)$$

where $g_l(R_l(t)) \triangleq R_l(t)(\frac{qM - \mu_M}{qM}Q_l(t) - Z_l(t)) - Vf_l(R_l(t))$ and $V > 0$ is a control parameter. Note that $g_l(\cdot)$ is convex and $\min_{R_l(t)} g_l(R_l(t)) \leq g_l(0) = 0$.

$R_l(t)$ regulator operates locally at the source node of link $l \in \mathcal{L}$ and actually regulates the virtual queue evolutions. Since $R_l(t)$ is the input rate to the virtual queue $Q_l(t)$ and the service rate to the virtual queue $Z_l(t)$, $R_l(t)$ is weighted by the (weighted) difference between $Q_l(t)$ and $Z_l(t)$ in (4.3.1). We will show later in Theorem 4.3.4 that with a large control parameter $V$, $ALG$ can approach the optimal utility. With the numerical results in Section 4.7.1, we will further analyze the effect.

Figure 4.1: Queue relationships
of $V$ on the throughput performance and the convergence rates of virtual queues $Q_l(t)$ and $Z_l(t)$.

2) **Congestion Controller:**

$$
\max_{0 \leq A_l(t) \leq \mu_M} A_l(t)(q_M - \mu_M - U_l(t)).
$$

(4.3.2)

Specifically, when $q_M - \mu_M - U_l(t) < 0$, $A_l(t)$ is set to zero; Otherwise, $A_l(t) = \mu_M$.

The congestion controller operates locally at the source node of link $l \in \mathcal{L}$, with the decision sent as feedback to the packet generator to generate $A_l(t)$ data packets to be admitted to the source node at the end of time slot $t$. The congestion controller also guarantees the finite buffer size property (see Proposition 4.3.1 in this section for detail).

3) **Link Rate Scheduler:** The link rate scheduler follows the form of parallel Glauber dynamics [83][28][48] with a probability vector $(p_l(t))_{l \in \mathcal{L}} \in [0, 1]^L$ to be defined later in Proposition 4.3.2. The scheduler operates as follows:

1. Randomly select an independent set $x(t) \in \mathcal{I}$ with probability (w.p.) $p_{x(t)}$, such that:

$$
\sum_{x(t) \in \mathcal{I}} p_{x(t)} = 1 \text{ and } \cup_{p_{x(t)}>0} x(t) = \mathcal{L}.
$$

(4.3.3)

2. $\forall l \in x(t),$

   - If $\sum_{j \in \mathcal{N}(l) \setminus \{t\}} \mu_j(t-1) = 0$: $\mu_l(t) = 1$, w.p. $p_l(t)$; $\mu_l(t) = 0$, w.p. $1 - p_l(t)$.
   - Else, $\mu_l(t) = 0$.

$\forall l \in \mathcal{L} \setminus x(t)$, $\mu_l(t) = \mu_l(t-1)$.

It is not difficult to check that the link set corresponding to the schedule $(\mu_l(t))_{t \in \mathcal{L}}$
is an independent set [83] and evolves as a discrete-time Markov chain (DTMC). The distributed implementation of the link rate scheduler is provided in Section 4.3.1, which is based on the CSMA paradigm with RTS/CTS handshake and requires no global signaling or message passing.

To analyze the performance of \textbf{ALG}, we first introduce the following two propositions. Proposition 4.3.1 shows that \textbf{ALG} has a deterministic worst-case upper-bound for all queues $U_l(t)$, $\forall l \in \mathcal{L}$. In Proposition 4.3.2, we show that the link rate scheduler finds a schedule approaching arbitrarily close to that of a maximal weight matching scheduler with high probability, when virtual queues $Q_l(t)$ are large enough. Using these two propositions, we derive the main results of utility optimality and virtual queue stability in Theorem 4.3.4.

**Proposition 4.3.1.** Employing \textbf{ALG}, if $\mu_M \leq q_M$, then each actual queue backlog in the network has a deterministic worst-case bound:

$$U_l(t) \leq q_M, \quad \forall t, \forall l \in \mathcal{L}, \quad (4.3.4)$$

where we recall that $q_M$ indicates the uniform link-based buffer size.

**Proof.** We use mathematical induction on time slot in the proof. When $t = 0$, we have $U_l(0) = 0 \forall l \in \mathcal{L}$.

Now suppose in time slot $t$, $U_l(t) \leq q_M \forall l \in \mathcal{L}$. In the induction step, for any given $l$, we consider two separate cases. In the first case, $U_l(t) \leq q_M - \mu_M$, then according to the queue dynamics (4.2.3) $U_l(t+1) \leq U_l(t) + \mu_M \leq q_M$. Otherwise, $U_l(t) > q_M - \mu_M$ and according to the congestion controller (4.3.2), we have $A_l(t) = 0$, which leads to $U_l(t+1) \leq U_l(t) \leq q_M$ by the queue dynamics (4.2.3).

Since the above analysis is true for any given $l \in \mathcal{L}$, the induction step holds, i.e., $U_l(t+1) \leq q_M \forall l \in \mathcal{L}$, which completes the proof. \hfill \blacksquare
Let \( w_l(t) = \frac{1}{q_M} U_l(t) Q_l(t) \). The Glauber dynamics structure of the link rate scheduler leads to the following proposition.

**Proposition 4.3.2.** Let \( p_l(t) = \frac{e^{w_l(t)}}{1 + e^{w_l(t)}} \), \( \forall l \in \mathcal{L} \) in the link rate scheduler. For any given \( \epsilon' \) and \( \delta' \) satisfying \( 0 < \epsilon', \delta' < 1 \), we can find a constant \( B(\epsilon', \delta') > 0 \) such that for any time slot \( t \), with probability greater than \( (1 - \delta') \), the link rate scheduler finds a schedule \( (\mu_l(t))_{l \in \mathcal{L}} \), satisfying:

\[
\sum_{l \in \mathcal{L}} w_l(t) \mu_l(t) \geq (1 - \epsilon') w^*(t), \quad \text{whenever} \quad ||q(t)|| > B,
\]

where \( q(t) \triangleq (w_l(t))_{l \in \mathcal{L}}, \ ||q(t)|| \triangleq \sum_{l \in \mathcal{L}} w_l(t), \) and \( w^*(t) \) is the result of a maximal weight matching scheduler: \( w^*(t) \triangleq \max_{x \in 2^\mathcal{L}} \sum_{l \in x} w_l(t) \).

Proposition 4.3.2 directly follows Proposition 2 in [83] with the time-scale separation assumption (the DTMC of the schedules chosen by the scheduler is in steady state in each time slot), so we omit the proof for brevity. By the time-scale separation assumption, we assume that the schedule converges much faster to its steady state distribution compared to the link weight \( (w_l(t) = \frac{1}{q_M} U_l(t) Q_l(t)) \) adaptation rate of the scheduler. This assumption has been used in previous works such as [51][83]. While the queue backlogs do not stay constant, we can update the link weights less frequently (i.e., the link weights stay constant for a sufficiently long period of time) to approach the time-scale separation, which is further discussed in Section 4.5.2.

**Remark 4.3.3.** (Implications of the Propositions): A maximal weight matching scheduler that obtains \( w^*(t) \) in each time slot \( t \) is usually centralized and computation-prohibitive but can lead to optimal throughput/utility [100][26][110]. Proposition 4.3.2 states that the proposed link rate scheduler can find a schedule using weight matching \( \epsilon' \)-close to \( w^*(t) \) with high probability \( (1 - \delta') \), when link weights \( w_l(t) \) are large enough.
By definition, $w_l(t)$ is the product of the virtual queue $Q_l(t)$ and the buffer occupancy $U_l(t)$. Since $U_l(t) \leq q_M$ holds $\forall l \in L$ by Proposition 4.3.1, (4.3.5) in Proposition 4.3.2 holds with high probability when some virtual queues $Q_l(t)$ are large enough with non-zero buffer occupancy. Since the actual queues are still upper-bounded by $q_M$, a bounded average packet delay is implied by Little’s Law. In other words, the design of the link weight $w_l(t)$ allows us to “shift the burden” of large queue backlogs from the actual queues $U_l(t)$ to the virtual queues $Q_l(t)$. Note that Proposition 4.3.2 also holds for link rate schedulers with link weights $w_l(t) = \log U_l(t)$ proposed in Q-CSMA [83], $w_l(t) = \log \log U_l(t)$ proposed in [85], $w_l(t) = \frac{\log U_l(t)}{y(U_l(t))}$ in [28], where $y(\cdot)$ is some function that increases arbitrarily slowly, and $w_l(t)$ is in the form of $U_l(t)$ in [48]. However, these schedulers approach a maximal weight matching scheduler when the actual queues are large enough, leading to large delays. On the other hand, our algorithm ALG does the same while limiting the maximum buffer size, which guarantees bounded average delay.

For notational simplicity, we define $\gamma \triangleq (1-\epsilon')(1-\delta')$, where we note that $\epsilon'$ and $\delta'$ in Proposition 4.3.2 can take values arbitrarily small. We present our main results in Theorem 4.3.4.

**Theorem 4.3.4.** Given some $\epsilon > 0$ arbitrarily small and $\gamma > \max_{l \in L} \frac{f_l^{-1}(d_l)}{a_{l,\epsilon} + \frac{1}{2\epsilon}}$, if

$$q_M > \frac{\mu_M^2 + 1}{\gamma\epsilon} + \mu_M,$$

(4.3.6)

ALG ensures that the virtual queues are stable:

$$\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{l \in L} \mathbb{E}\{Q_l(\tau) + Z_l(\tau)\} \leq \frac{B_2 + V g_M}{\delta_1},$$

(4.3.7)
where \( B_1 \triangleq \frac{1}{2} \mu_M q_M L + \mu_M^2 \frac{q_M - \mu_M}{q_M} L + \frac{1}{2} \mu_M^2 L + \frac{1}{2} \sum_{l \in L} (f_l^{-1}(d_l))^2 \), \( B_2 \triangleq B_1 + \gamma B(\max_{l \in L} a_{l, e}^* + \epsilon) \),

\[
\delta_1 \triangleq \min \{ \gamma \epsilon(q_M - \mu_M) - \mu_M^2 - 1, \\
\min_{l \in L} [\gamma(a_{l, e}^* + \frac{1}{2} \epsilon) - f_l^{-1}(d_l)] \},
\]

and

\[
g_M \triangleq \limsup_{t \to \infty} \frac{1}{t} \sum_{t=0}^{t-1} \sum_{l \in L} \mathbb{E}\{f_l(R_l(\tau))\} - \gamma \sum_{l \in L} f_l(a_{l, e}^* + \frac{1}{2} \epsilon)
\]

\[
\leq \sum_{l \in L} f_l(\mu_M) - \gamma \sum_{l \in L} f_l(a_{l, e}^* + \frac{1}{2} \epsilon).
\]

In addition, \( \text{ALG} \) can achieve

\[
\sum_{l \in L} f_l(a_l) \geq \gamma \sum_{l \in L} f_l(a_{l, e}^* + \frac{1}{2} \epsilon) - \frac{B_2}{V}.
\]

Proof. We prove Theorem 4.3.4 in Section 4.3.2, with the employment of Proposition 4.3.1 and Proposition 4.3.2. \( \square \)

Remark 4.3.5. (Algorithm Performance): 1) Network Stability and Minimum Utility Constraints: (4.3.4) in Proposition 4.3.4 indicates that \( \text{ALG} \) stabilizes the actual queue backlogs. As an immediate result, the network is stable. (4.3.7) in Theorem 4.3.4 shows that the virtual queues are stable and, hence, the minimum utility constraints are satisfied. Note that from (4.3.7), the virtual queues can grow with an increased value of parameter \( V \). In fact, a large \( V \) implies a slow convergence rate of virtual queues, which will be explained in more detail with numerical results in Section 4.7.1. 2) Utility Optimality and Buffer Size of order \( O(\frac{1}{\epsilon}) \): The inequality (4.3.8) gives the lower-bound of the utility \( \text{ALG} \) can achieve. Since the constant \( B_2 \)
is independent of $V$ and $\gamma$ can be chosen arbitrarily close to 1 by definition, (4.3.8) ensures that $\text{ALG}$ can achieve a utility arbitrarily close to $\sum_{l \in L} f_l(a^*_l + \frac{1}{2} \epsilon)$ by selecting a $V$ large enough and by choosing small $\epsilon'$, $\delta'$ such that $\gamma$ is close to 1. When $\epsilon$ tends to 0, $\text{ALG}$ can achieve a utility arbitrarily close to the optimal value $\sum_{l \in L} f_l(a^*_l)$ with the tradeoff in buffer size $q_M$, which is of order $O(\frac{1}{\epsilon})$ as shown in (4.3.6). Note that the only assumption for Proposition 4.3.2 (and hence Theorem 4.3.4) to hold is the time-scale separation assumption, which has also been employed in [51][83] and justified in [84][50]. When this assumption holds, the buffer size has an order of $O(\frac{1}{\epsilon})$ which is independent of the network size, i.e., an order-optimal delay performance [72] is achieved.

As a result of Proposition 4.3.1 and Theorem 4.3.4, we show in the following corollary that the average queuing delay for each link is upper-bounded.

**Corollary 4.3.6.** Let $M(T)$ be the number of packets admitted to any given queue $U_l(t)$ by any given time slot $T$ and let $W_j$ be the queuing delay of packet $j$ by time slot $T$, $j = 1, 2, ..., M(T)$. Under $\text{ALG}$, the average queuing delay is upper-bounded, i.e., for any given link $l \in L$ and any $0 < \epsilon_0 < f_l^{-1}(d_l)$,

$$\limsup_{T \to \infty} \frac{1}{M(T)} \sum_{j=1}^{M(T)} W_j \leq \frac{q_M}{f_l^{-1}(d_l) - \epsilon_0}. \quad (4.3.9)$$

**Proof:** According to the derivation of Little's Law, we have the following equation:

$$\sum_{t=1}^{T} U_l(t) = \sum_{j=1}^{M(T)} W_j,$$

114
from which we have:

\[ q_M \geq \frac{1}{T} \sum_{t=1}^{T} U_i(t) = \frac{1}{T} \sum_{j=1}^{M(T)} W_j. \]

Note that under \( \text{ALG} \), the time-averaged admitted rate is lower-bounded by \( f_i^{-1}(d_i) \) (since virtual queues \( Q_i(t) \) and \( Z_i(t) \) are stable from Theorem 4.3.4), i.e.,

\[ \liminf_{T \to \infty} \frac{M(T)}{T} \geq f_i^{-1}(d_i). \]

Hence, there exists \( K > 0 \) such that for all \( T > K \),

\[ \inf_{t \geq T} \frac{M(t)}{t} \geq f_i^{-1}(d_i) - \epsilon_0. \]

Thus, we have, for all \( T > K \),

\[ \frac{1}{M(T)} \sum_{j=1}^{M(T)} W_j \leq \frac{1}{T(f_i^{-1}(d_i) - \epsilon_0)} \sum_{i=1}^{M(T)} W_j \leq \frac{q_M}{f_i^{-1}(d_i) - \epsilon_0}. \]

We take the limsup of \( T \) on both sides of the above inequality, proving (4.3.9).

### 4.3.1 Distributed Link Rate Scheduler

In the following, we present a distributed implementation of the link rate scheduler. This distributed scheduler is a variation of the distributed scheduler in [83], which is based on the RTS/CTS mechanism in the 802.11 standard and is proposed for single-hop wireless networks. Note that the RTS/CTS handshake is only a tool to implement the distributed local interaction of the algorithm and other alternatives may exist. We assume each node has a single transceiver, i.e., a node can not transmit
and receive at the same time. Let the set of links with the same source node \( n \in \mathcal{N} \) be \( I(n) = \{(n, i) : (n, i) \in \mathcal{L}\} \), and let \( \mu_l(0) = 0, \forall l \in \mathcal{L} \).

The main difficulty in the distributed implementation is to randomly select an independent link set satisfying condition (4.3.3). To achieve this, we employ the RTS/CTS mechanism. Specifically, in each time slot, we assign a number of \( T_s \) control mini-slots before data transmissions. The distributed link scheduler is operated at each node \( n \in \mathcal{N} \) for each time slot \( t \), as illustrated in Figure 4.2. At the beginning of each time slot, the node \( n \) selects a link \((n, r_n)\) from \( I(n) \) uniformly at random and chooses a random backoff time from the \((T_s - 1)\) mini-slots (Step 1 in Figure 4.2).

Then under some conditions specified in Step 2 in Figure 4.2, node \( n \) initiates an RTS (Request-To-Send)-CTS (Clear-To-Send) handshake with its destination node \( r_n \) when the backoff duration is over. To facilitate the RTS/CTS handshake, each mini-slot is further split into two micro-slots (Step 2.1 and Step 2.2 in Figure 4.2), dedicated for CTS and RTS transmissions, respectively. After the RTS/CTS handshakes, the distributed link rate scheduler assigns link rates (Step 3 in Figure 4.2) following the same assignment in the scheduler with Glauber dynamics described in Section 4.3.

For narrative clarity, we let RTS\((n)\) and CTS\((n)\) denote, respectively, the RTS intended for node \( n \) and the CTS intended for node \( n \), with \( n \in \mathcal{N} \).

It is not difficult to check that under the distributed link rate scheduler, any link set \( x(t) \) whose links succeed in the RTS-CTS handshake (i.e., Step 2 in Figure 4.2) is an independent link set and \( \bigcup_{P(x(t))>0} x(t) = \mathcal{L} \), satisfying (4.3.3). Thus, the distributed link rate scheduler is equivalent to the scheduler in Section 4.3.

4.3.2 Proof of Theorem 4.3.4

Before we proceed, we present the following lemma which will assist us in proving Theorem 4.3.4.
Distributed Link Rate Scheduler at node $n$ in time slot $t$

**Step 1.** Initialization:
A node $r_n$ is selected in $I(n)$ uniformly at random; $n$ chooses a backoff time $b_n$ uniformly at random from the first $(T_s - 1)$ mini-slots.

**Step 2.** RTS-CTS handshake:
for mini-slot $t_s = 1 : T_s$ (i.e., $t_s$ is the current mini-slot)

**Step 2.1.** The first micro-slot: CTS transmission
if $n$ has not overheard in the past any RTS($m$), $\forall m \neq n$
and $n$ has not overheard any collision of RTS before
and $n$ has not sent out a CTS before
and $n$ has not received CTS($n$) before
and $n$ received only one RTS($n$) in mini-slot $(t_s - 1)$
do: node $n$ sends out a CTS responding to the RTS($n$) received in mini-slot $(t_s - 1)$;
end if

**Step 2.2.** The second micro-slot: RTS transmission
if $n$ has not overheard in the past any CTS($m$), $\forall m \neq n$
and $n$ has not overheard any collision of CTS before
and $n$ has not sent a CTS before
and $b_n = t_s$
do: node $n$ sends out an RTS($r_n$) to node $r_n$;
end if
end for

**Step 3.** Schedule for links $\{(n,i) : (n,i) \in L\}$:
$\mu_{ni}(t) = \mu_{ni}(t - 1)$, $\forall i \neq r_n : (n,i) \in L$.
if $n$ received CTS($n$) from $r_n$ in Step 2 do:
if $\sum_{j \in N((n,r_n)) \setminus \{(n,r_n)\}} \mu_j(t - 1) > 0$
do: $\mu_{nr_n}(t) = 0$;
else
do: $\mu_{nr_n}(t) = 1$ w.p. $p_{(n,r_n)}$ defined in Proposition 4.3.2;
$\mu_{nr_n}(t) = 0$ w.p. $(1 - p_{(n,r_n)})$;
end if
else (i.e., $(n,r_n)$ failed the RTS-CTS handshake)
do: $\mu_{nr_n}(t) = \mu_{nr_n}(t - 1)$;
end if

Figure 4.2: Distributed link rate scheduler for networks with single-hop transmissions.
Lemma 4.3.7. For any feasible rate vector \((\theta_l)_{l \in L} \in \Lambda\), there exists a stationary randomized algorithm \(\text{STAT}\) that stabilizes the network with input rate vector \((A_{i,\text{STAT}}^i(t))\) and scheduling parameters \((\mu_{i,\text{STAT}}^i(t))\) independent of queue backlogs, such that the admitted rates are:

\[ A_{i,\text{STAT}}^i(t) = \mathbb{E}\{\mu_{i,\text{STAT}}^i(t)\} = \theta_l, \forall t, \forall l \in L. \]

Note that it is not necessary for the randomized algorithm \(\text{STAT}\) to be distributed or finite-buffered. Similar formulations of \(\text{STAT}\) and their proofs have been given in [26][78][71], so we omit the proof of Lemma 4.3.7 for brevity.

Remark 4.3.8. According to the \(\text{STAT}\) algorithm in Lemma 4.3.7, we assign the input rates of the virtual queues \(Q_i(t)\) as \(R_{i,\text{STAT}}^i(t) = \theta_l\). Note that \((\theta_l)\) can take values such as \((a_{i,\epsilon}^* + \frac{1}{2}\epsilon)\) or \((a_{i,\epsilon}^* + \epsilon)\).

To prove Theorem 4.3.4, we first define a network state of

\[ Q_1(t) = ((U_i(t)), (Q_i(t)), (Z_i(t))) \]

and then define the Lyapunov function \(L_1(Q_1(t))\) as follows:

\[ L_1(Q_1(t)) \triangleq \frac{1}{2}\sum_{i \in L}\{q_M - \frac{\mu_M}{q_M}Q_i(t)^2 + \frac{1}{q_M}U_i(t)^2Q_i(t) + Z_i(t)^2\}, \]

where we note that different from the traditional quadratic Lyapunov function, there is the (weighted) virtual queue backlog \(\frac{Q_i(t)}{q_M}\) multiplied to the quadratic term of actual queue backlogs. We denote the Lyapunov drift by

\[ \Delta(t) \triangleq \mathbb{E}\{L_1(Q_1(t + 1)) - L_1(Q_1(t))|Q_1(t)\}. \]
By squaring both sides of the queue dynamics (4.2.3)(4.2.4)(4.2.6) and through simple algebra, we obtain

\[
\Delta(t) - V \sum_{l \in \mathcal{L}} \mathbb{E}\{f_l(R_l(t))|Q_1(t)\} \\
\leq B_1 + \frac{\mu_M^2 + 1}{2q_M} \sum_{l \in \mathcal{L}} Q_l(t) - V \sum_{l \in \mathcal{L}} \mathbb{E}\{f(R_l(t))|Q_1(t)\} \\
- \mathbb{E}\{\sum_{l \in \mathcal{L}} \frac{1}{q_M} Q_l(t) U_l(t)(\mu_l(t) - A_l(t))|Q_1(t)\} \\
- \mathbb{E}\{\sum_{l \in \mathcal{L}} \frac{q_M - \mu_M}{q_M} Q_l(t)(A_l(t) - R_l(t))|Q_1(t)\} \\
- \mathbb{E}\{\sum_{l \in \mathcal{L}} Z_l(t)(R_l(t) - f_l^{-1}(d_l))|Q_1(t)\}
\]

the equivalence of which is given as follows:

\[
\Delta(t) - V \sum_{l \in \mathcal{L}} \mathbb{E}\{f_l(R_l(t))|Q_1(t)\} \\
\leq B_1 + \frac{\mu_M^2 + 1}{2q_M} \sum_{l \in \mathcal{L}} Q_l(t) + \sum_{l \in \mathcal{L}} f_l^{-1}(d_l) Z_l(t) \\
+ \sum_{l \in \mathcal{L}} \mathbb{E}\{g_l(R_l(t))|Q_1(t)\} \\
- \sum_{l \in \mathcal{L}} \frac{1}{q_M} U_l(t) Q_l(t) \mathbb{E}\{\mu_l(t)|Q_1(t)\} \\
- \sum_{l \in \mathcal{L}} \frac{Q_l(t)}{q_M} \mathbb{E}\{A_l(t)|Q_1(t)\}(q_M - \mu_M - U_l(t)).
\]  

(4.3.10)

Denoting the RHS of (4.3.10) by \(RHS(t)\), and taking the expectation of both
sides of (4.3.10) with respect to the distribution of $Q_1(t)$, we have:

$$\mathbb{E}\{\Delta(t)\} - V \sum_{l \in \mathcal{L}} \mathbb{E}\{f_l(R_l(t))\}$$

$$\leq P(||q(t)|| > B) \mathbb{E}\{Q_1(t)||q(t)|| > B\} \{RHS(t)\}$$

$$+ P(||q(t)|| \leq B) \mathbb{E}\{Q_1(t)||q(t)|| \leq B\} \{RHS(t)\}. \hspace{1cm} (4.3.11)$$

Note that from the $R_l(t)$ regulator and the congestion controller in $\text{ALG}$, the fourth and the last terms of $RHS(t)$ are non-positive. By employing Proposition 4.3.2, the $RHS(t)$ inside the first term of the RHS of (4.3.11) satisfies:

$$RHS(t)$$

$$\leq B_1 + \frac{\mu^2 M + 1}{2q_M} \sum_{l \in \mathcal{L}} Q_l(t) + \sum_{l \in \mathcal{L}} f^{-1}_l(d_l) Z_l(t)$$

$$+ \gamma \sum_{l \in \mathcal{L}} \mathbb{E}\{g_l(R_l(t))|Q_1(t)\} - \gamma \mathbb{E}\{w^*(t)|Q_1(t)\}$$

$$- \gamma \sum_{l \in \mathcal{L}} \frac{Q_l(t)}{q_M} \mathbb{E}\{A_l(t)|Q_1(t)\}(q_M - \mu - U_l(t)). \hspace{1cm} (4.3.12)$$

Similarly, the $RHS(t)$ inside the second term of the RHS of (4.3.11) satisfies:

$$RHS(t)$$

$$\leq B_1 + \frac{\mu^2 M + 1}{2q_M} \sum_{l \in \mathcal{L}} Q_l(t) + \sum_{l \in \mathcal{L}} f^{-1}_l(d_l) Z_l(t)$$

$$+ \gamma \sum_{l \in \mathcal{L}} \mathbb{E}\{g_l(R_l(t))|Q_1(t)\}$$

$$- \gamma \sum_{l \in \mathcal{L}} \frac{Q_l(t)}{q_M} \mathbb{E}\{A_l(t)|Q_1(t)\}(q_M - \mu - U_l(t)). \hspace{1cm} (4.3.13)$$

The fourth term and the last term of the RHS of (4.3.12)(4.3.13) are minimized by the $R_l(t)$ regulator (4.3.1) and the congestion controller (4.3.2), respectively, over a set of feasible algorithms including the stationary randomized algorithm STAT introduced.
in Lemma 4.3.7. We can substitute into the fourth term of RHS of (4.3.12) a stationary randomized algorithm STAT with admitted arrival rate vector \((a^*_{l,\epsilon} + \frac{1}{2}\epsilon)\) and into the last term with an STAT with admitted arrival rate vector \((a^*_{l,\epsilon} + \epsilon)\). In addition, since \(-w^*(t)\) minimizes \(-\sum_{l \in \mathcal{L}} U_l(t)Q_l(t)\mu_l(t)\) over all feasible schedulers, we can substitute into the fifth term of RHS of (4.3.12) an STAT with admitted arrival rate vector \((a^*_{l,\epsilon} + \epsilon)\). By the above substitutions and rearranging terms, we obtain from (4.3.11):

$$
\mathbb{E}\{\Delta(t)\} - V \sum_{l \in \mathcal{L}} \mathbb{E}\{f_l(R_l(t))\}
\leq B_1 - \gamma V \sum_{l \in \mathcal{L}} f_l(a^*_{l,\epsilon} + \frac{1}{2}\epsilon)
- \sum_{l \in \mathcal{L}} [\gamma(a^*_{l,\epsilon} + \frac{1}{2}\epsilon) - f^{-1}(d_l)] \mathbb{E}\{Z_l(t)\}
- \left(\frac{q_M - \mu_M}{2q_M} - \frac{\mu_M^2 + 1}{2q_M}\right) \mathbb{E}\{\sum_{l \in \mathcal{L}} Q_l(t)\}
+ \mathbb{E}\left\{\frac{\gamma}{q_M} \sum_{l \in \mathcal{L}} U_l(t)Q_l(t)(a^*_{l,\epsilon} + \epsilon) ||\mathbf{q}(t)|| \leq B\right\},
$$

which leads to:

$$
\mathbb{E}\{\Delta(t)\} - V \sum_{l \in \mathcal{L}} \mathbb{E}\{f_l(R_l(t))\}
\leq B_2 - \delta_1 \sum_{l \in \mathcal{L}} \mathbb{E}\{Q_l(t) + Z_l(t)\} - \gamma V \sum_{l \in \mathcal{L}} f_l(a^*_{l,\epsilon} + \frac{1}{2}\epsilon).
$$

(4.3.15)

We take the time average on \(\tau = 0, ..., t - 1\) of both sides of (4.3.15), and take the limsup with respect to \(t\) to prove (4.3.7). Similarly, we can prove (4.3.8) given that the virtual queues \(Q_l(t)\) are stable.
4.4 Constructing Individual Average Delay Upper-bound

For delay-sensitive traffic, the delay performance of a network is of more significance than the utility optimality. In this section, we propose an algorithm modified from the original one that guarantees link-based minimum utility constraint. Specifically, we employ a modified version of virtual queues $Z_l(t)$ without using $Q_l(t)$ in the modified algorithm. Note that the employment of a single virtual queue $Z_l(t)$ no longer guarantees the utility optimality. However, at the expense of losing utility optimality, we can construct non-trivial link-based delay upper-bounds for the modified algorithm.

Instead of setting $q_M$ as the global buffer size, we now define and employ in this section different buffer sizes for each link, i.e., $q_l$, $l \in \mathcal{L}$. Instead of (4.2.6), we now update the virtual service queue as:

$$Z_l(t + 1) = [Z_l(t) - A_l(t)]^+ + f_i^{-1}(d_i), \forall l \in \mathcal{L},$$

where we recall that $(d_l)_{l \in \mathcal{L}}$ is the link-based minimum utility constraint and the minimum utility constraint is met when the virtual queues $Z_l(t)$ are stable. Without loss of generality, we assume $d_l > 0$, $\forall l \in \mathcal{L}$, in this section. The algorithm is composed of two functions:

1) **Congestion Controller**:

$$\max_{0 \leq A_l(t) \leq \mu_M} A_l(t)(q_l - \mu_M - U_l(t)),$$

which is modified from the congestion controller (4.3.2) introduced in Section 4.3 by replacing $q_M$ with $q_l$.

2) **Link Rate Scheduler**: Let $w_l(t) = \frac{1}{q_l}Z_l(t)U_l(t)$, $\forall l \in \mathcal{L}$. The link rate scheduler is the same as that in Section 4.3 with $p_l(t) = \frac{e^{w_l(t)}}{1 + e^{w_l(t)}}$, $l \in \mathcal{L}$. 122
It is not difficult to check that \( U_l(t) \leq q_l, \forall l \in L \forall t \), (i.e., the finite buffer sizes and network stability) and Proposition 4.3.2 still hold for the above algorithm. In addition, the minimum utility constraints are satisfied, which is formally stated in the following theorem.

**Theorem 4.4.1.** For any arbitrarily vector \((\epsilon_l)_{l \in L}\) such that

\[
\epsilon_l > 0, \forall l \in L, \text{ and } (f_l^{-1}(d_l) + \epsilon_l)_{l \in L} \in \Lambda,
\]

(4.4.1)
given \( \gamma \geq \max_{l \in L} \frac{f_l^{-1}(d_l)}{f_l^{-1}(d_l) + \epsilon_l} \), if

\[
q_l > \frac{\mu_M^2 + 1}{2[(\gamma - 1)f_l^{-1}(d_l) + \gamma \epsilon_l]} + \mu_M,
\]

(4.4.2)
then the virtual service queues \( Z_l(t) \) are stable as shown in (4.4.3) on top of the next page.

\[
\limsup_{t \to \infty} \frac{1}{t} \sum_{l=0}^{t-1} \sum_{l \in L} \mathbb{E}\{Z_l(\tau)\} \leq \sum_{l \in L} \left\{ q_l^2 f_l^{-1}(d_l) + (q_l - \mu_M)(1 + f_l^{-1}(d_l)^2) + 2q_l d_l + \epsilon_l B \right\},
\]

\[
\min_{l \in L} \left\{ 2(q_l - \mu_M)[(\gamma - 1)f_l^{-1}(d_l) + \gamma \epsilon_l] - \mu_M^2 - 1 \right\}.
\]

(4.4.3)

The proof for Theorem 4.4.1 is similar to that of Theorem 4.3.4 developed in Section 4.3.2 and is omitted for brevity. From Theorem 4.4.1, we can construct the delay upper-bounds, given in the following corollary.
Corollary 4.4.2. The modified algorithm can guarantee an upper-bound for the average delay $D_l$ for link $l \in \mathcal{L}$:

$$D_l \leq \frac{1}{f^{-1}(d_l)} \left( \mu_M + \frac{\mu_M^2 + 1}{2\epsilon_l} \right) + \eta,$$

(4.4.4)

where $\eta$ is any arbitrarily small positive constant and $(\epsilon_l)_{l \in \mathcal{L}}$ satisfies (4.4.1).

Proof. Let the buffer sizes be chosen as

$$q_l = \frac{\mu_M^2 + 1}{2\epsilon_l} + \mu_M + \eta f^{-1}(d_l), \forall l \in \mathcal{L}. \quad (4.4.5)$$

To satisfy the minimum utility constraints, from (4.4.2) in Theorem 4.4.1, we should have:

$$\frac{\mu_M^2 + 1}{2\epsilon_l} + \mu_M + \eta f^{-1}(d_l) > \frac{\mu_M^2 + 1}{2[(\gamma - 1)f^{-1}(d_l) + \gamma \epsilon_l]} + \mu_M, \forall l \in \mathcal{L},$$

which leads to

$$\gamma > \max_{l \in \mathcal{L}} \frac{f^{-1}(d_l) + \eta}{f^{-1}(d_l) + \epsilon_l}. \quad (4.4.6)$$

Since the RHS of (4.4.6) is smaller than 1, as long as $\gamma$ satisfies (4.4.6), the choices of buffer sizes (4.4.5) are feasible. Then, the delay upper-bounds (4.4.4) in Corollary 4.4.2 immediately follows from Little’s Law.

Remark 4.4.3. In the above algorithm, we employ $Z_l(t)$ instead of $Q_l(t)$ in the weight $w_l(t), \forall l \in \mathcal{L}$. From the queue dynamics, $Z_l(t)$ can be small when the link service rate is greater than $f_l^{-1}(d_l)$, which results in a small link activation probability $p_l$. Hence, the algorithm no longer guarantees utility optimality. At the expense of losing
utility optimality, nontrivial per-flow delay upper-bounds can be derived and tailored by selecting different \((\epsilon_l)_{l \in \mathcal{L}}\) values, where \((\epsilon_l)_{l \in \mathcal{L}}\) can be chosen anywhere between the minimum data rate vector \((f_l^{-1}(d_l))_{l \in \mathcal{L}}\) and some rate vector on the boundary of the capacity region \(\Lambda\). Since the average delay \(D_l\) is of order \(O\left(\frac{1}{\epsilon_l}\right)\) from (4.4.4), we can choose \((\epsilon_l)_{l \in \mathcal{L}}\) as the difference between \((f_l^{-1}(d_l))_{l \in \mathcal{L}}\) and some rate vector on the boundary (or close to the boundary) of \(\Lambda\) to obtain tighter delay upper-bounds.

### 4.5 Approaching Time-Scale Separation

We recall that Proposition 4.3.2 (and hence Theorem 4.3.4) in Section 4.3 are based on the time-scale separation assumption. This assumption requires the schedule determined by the link rate scheduler to converge to its steady-state faster than the rate at which link weights \(w_l(t)\) change over time. In this section, we discuss two approaches to approximate this time-scale separation: (i) changing the link weights at a slow rate (ii) updating the link weights less frequently.

#### 4.5.1 Small Changes in \(w_l(t)\)

In the link rate scheduler introduced for *ALG* in Section 4.3, we assign the link weights as:

\[
w_l(t) = \frac{\alpha}{q_M} U_l(t) Q_l(t),
\]

where \(\alpha\) is a small positive constant such that \(w_l(t)\), and hence the link activation probability \(p_l\), change slowly over time.

It is not difficult to show that Proposition 4.3.1, Proposition 4.3.2 and Theorem 4.3.4 hold for this choice of link weights, with a minor change in the constants \(B\) and \(B_2\). Hence, if time-scale separation holds, we can still expect an order of \(O\left(\frac{1}{\epsilon}\right)\)
in buffer sizes as in (4.3.6), which further induces that the average link delay is of order \(O(\frac{1}{\epsilon})\), i.e., order-optimal delay can be achieved. However, an \(\alpha\) too small may reduce both the link activation probabilities and the responsiveness of the scheduler to the queue length variations. An alternative method to approach the time-scale separation is provided in the next subsection.

4.5.2 Slow Updates in \(w_l(t)\)

We can make the DTMC of the schedules converge to the steady state distributions by updating the weights in the link rate scheduler periodically every \(T\) time slots:

\[
w_l(t) = \frac{1}{q_M} U_l(kT)Q_l(kT), \quad kT \leq t < (k + 1)T,
\]

while other components of the algorithm remain the same, where \(T\) is the update period and \(k\) takes integer values.

Then Proposition 4.3.1 still holds for \(\text{ALG}\) with slow updates, and we have the following theorem:

**Theorem 4.5.1.** Given some \(\epsilon > 0\) arbitrarily small and \(\gamma > \max_{l \in \mathcal{L}} \frac{f_l^{-1}(d_l)}{a_{l,\epsilon} + \frac{1}{2}\epsilon}\), if

\[
q_M > \frac{2(T - 1)(1 + \gamma \mu M) + \mu M^2 + 1}{\gamma \epsilon} + \mu M,
\]

\(\text{ALG}\) with slow updates ensures the stability of virtual queues:

\[
\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{l \in \mathcal{L}} \mathbb{E}\{Q_l(\tau) + Z_l(\tau)\} \leq \frac{B_3 + V g_M}{\delta_3},
\]

126
where $B_4 \triangleq B_2 + (1 + \gamma)\mu_M L(T - 1)$ and

$$
\delta_3 \triangleq \min\left\{ \frac{\gamma \epsilon (q_M - \mu_M) - \mu_M^2 - 1 - 2(T - 1)(1 + \gamma \mu_M)}{2q_M}, \min_{l \in \mathcal{L}} \gamma (a_{i^*}^l + \frac{1}{2} \epsilon) - f_l^{-1}(d_l) \right\}.
$$

In addition, ALG can achieve

$$
\sum_{l \in \mathcal{L}} f(a_l) \geq \sum_{l \in \mathcal{L}} f(a_{i^*}^l + \frac{1}{2} \epsilon) - \frac{B_4}{V}.
$$

Proof. Given time slot $t = kT + i$, where $0 \leq i < T$, by analyzing the queue dynamics (4.2.3)(4.2.4) we have:

$$
U_l(t)Q_l(t) \leq (U_l(kT) + i\mu_M)Q_l(t)
$$

(4.5.3)

$$
\leq U_l(kT)Q_l(kT) + i\mu_M q_M + i\mu_M Q_l(t).
$$

Similarly,

$$
U_l(kT)Q_l(kT) \leq U_l(t)Q_l(t) + i\mu_M q_M + iQ_l(t).
$$

(4.5.4)

By employing the inequalities (4.5.3)(4.5.4), following the steps in deriving (4.3.14)
in the proof of Theorem 4.3.4, we can obtain:

\[
E\{\Delta(t)\} - V \sum_{l \in L} E\{f_l(R_l(t))\} \\
\leq B_2 + (1 + \gamma)\mu_M L(T - 1) - \gamma V \sum_{l \in L} f_l(a^*_l, \epsilon) + \frac{1}{2} \epsilon \\
- \sum_{l \in L} |\gamma(a^*_l, \epsilon) + \frac{1}{2} \epsilon - f^{-1}(d_l)| E\{Z_l(t)\} \\
- E\{\sum_{l \in L} Q_l(t) \left( \frac{\gamma \epsilon(q_M - \mu_M)}{2q_M} \right) \\
- \frac{\mu_M^2 + 1 + 2(T - 1)(1 + \gamma \mu_M)}{2q_M}\}.
\]

which leads to:

\[
E\{\Delta(t)\} - V \sum_{l \in L} E\{f_l(R_l(t))\} \\
\leq B_4 - \delta_3 \sum_{l \in L} E\{Q_l(t) + Z_l(t)\} - \gamma V \sum_{l \in L} f_l(a^*_l, \epsilon) + \frac{1}{2} \epsilon.
\]

Following the steps in deriving Theorem 4.3.4, we can prove Theorem 4.5.1.

**Remark 4.5.2.** In [48], by defining the mixing time of a Markov chain as the time required for the Markov chain to get close to the stationary distribution, it has been shown that the mixing time corresponding to the parallel Glauber dynamics of the link rate scheduler is of order $O(\log N)$, where we recall $N$ is the number of nodes in the network. Thus, we can expect the buffer size (and hence the average delay bound) to be of order $O(\log N)$ from (4.5.2) by choosing $T = O(\log N)$ in (4.5.1). This still presents a significant improvement over existing solutions using the periodic update approach: The average actual queue length under the algorithm proposed in [48] is of order $O(N^2 \log N)$ (from Theorem 8 in [48]). Note that our results do not conflict with the findings in [90]. Specifically, in [90], it is shown that in open-loop systems,
the network delay grows exponentially with the network size. However, in our work, a closed-loop system is considered, i.e., the rate of packet admission/generation into the network is determined by the scheduler.

4.6 Implementation Study

To validate ALG, the proposed algorithm for systems with single-hop transmissions and constantly backlogged sources in Section 4.3, we compare ALG with an algorithm: Q-CSMA: A distributed throughput-optimal algorithm proposed in [83], with link weights in the scheduler chosen as \( w_l(t) = \log(U_l(t)) \). This has been shown in [83][28] to achieve better delay performance than the throughput-optimal algorithms with link weights of the form \( U_l(t) \) [48] and \( \log \log(U_l(t)) \) [85].

To guarantee fairness in the comparison, we construct the same congestion controller (4.3.2) for Q-CSMA and ALG. It is easy to check that queue backlogs \( U_l(t) \) are also bounded by \( q_M \) in our implementation of Q-CSMA. Therefore, \( q_M \) can be considered as the buffer size for both algorithms, and the packet delay is defined in the same way for both algorithms (i.e., the period from the time a packet gets admitted into the network to the time it gets served).

We implement the proposed algorithm and Q-CSMA in hardware on the Crossbow Telos-B platform as shown in Figure 4.3(a). Each node is equipped with a CC2420 802.15.4 wireless transceiver [45] and a programmable MSP430 processor [46]. For each protocol implementation, a time-slotted structure was used to coordinate the distributed link rate scheduler as described in Section 4.3.1. To maintain the time-slotted structure, nodes periodically exchange timing information every \( N_{sync} \) time slots to realign their internal clocks. The value of \( N_{sync} \) is dependent on the estimated clock drift of the nodes and the duration of the mini-slots and time slots. For our implementation, resynchronizing the node clocks every 100 time-slots is sufficient.
Time-synchronization is initialized based on \textit{a priori} knowledge of the network topology. If the network topology is not known, this information can be obtained via a neighbor discovery protocol [23].

We test both $\text{ALG}$ and Q-CSMA on the topology shown in Figure 4.3(b) with 5 nodes and 10 directional links. In this test, all nodes are within communication range of each other and therefore are in the same interference set, i.e., we employ the two-hop interference model. Under this model, at most 1 link can be activated in each time slot and hence the optimal throughput is 1. For simplicity we choose the utility functions as $f_l(x) = x$, $\forall l \in L$, i.e., we consider data rate constraints and throughput optimization instead of utility constraint and utility optimization. We choose the link rate weights as $w_l = \frac{0.1}{q_M} U_l(t) Q_l(t)$, $\forall l \in L$, as proposed in Section 4.5.1 where $\alpha = 0.1$. We fix the number of mini-slots (introduced in Section 4.3.1) as $T_s = \max_{l \in L} |N(l)| + 1 = 11$ and let $v_M = \mu_M = 2$.

Figure 4.3(c) shows the delay performance comparison between $\text{ALG}$ and Q-CSMA as a function of network throughput, where packet delay is measured in unit of time slots and throughput in unit of packets. We set the buffer size $q_M = 5$ in $\text{ALG}$ since we observe that $q_M = 5$ is sufficient to achieve a near optimal throughput. Each result represents the average delay and throughput over 10000 time slots. Note that in Figure 4.3(c), to ensure a fair comparison with Q-CSMA, we do not impose minimum utility/throughput constraints to individual links for both $\text{ALG}$ and Q-CSMA. By increasing $V$, the network throughput can be improved \textit{without sacrificing delay performance} under $\text{ALG}$. This might sound counter-intuitive, but can be explained by the Little’s Law. With an increasing $V$, the throughput increases, i.e., the actual packet queues are emptying faster and being serviced at an increased rate, while we observe that the actual queue backlogs are close to (and upper-bounded by) the finite buffer size ($q_M = 5$ in this experiment). Hence, the average delay can be
slightly reduced when $V$ increases. With Q-CSMA, buffer size $q_M$ must be increased significantly to achieve a throughput similar to that of $ALG$, which results in a significantly large average delays. Note that for small values of $q_M$, the average delay for Q-CSMA may be less than that of $ALG$. However, by tuning $V$ for $ALG$, higher throughput can be achieved without sacrificing delay performance.

Note that we can increase $V$ to further improve network performance of $ALG$. However, very large values of $V$ cause the virtual queue ($Q_l(t)$) to increase over the course of the 10000 time slot tests. A more detailed explanation of the effect of $V$ on the system convergence time is given through simulation results in Section 4.7.

4.7 Further Numerical Results

In this section, we further evaluate the performance of $ALG$ for constantly backlogged sources in simulation.
4.7.1 Effect of Parameter $V$

In Section 4.7.1 and Section 4.7.2, for the same topology of Figure 4.3(b), we employ an interference model different than that of Section 4.6, the well-known node-exclusive interference model in our simulation studies. Under this model, at most 2 links can be activated in each time slot and hence the optimal throughput is 2. As in Section 4.6, we adopt the utility function $f_l(x) = x$ and choose the link weights as $w_l = \frac{0.1}{q_M}U_l(t)Q_l(t), \forall l \in L$. The results reflect averages obtained over $10^5$ time slots for each run. We fix the maximal data admission rate as $\mu_M = 2$ and let $v_M = 2$.

We first illustrate the algorithm performance in Figure 4.6 by varying the parameter $V$, which we recall is a control parameter in the $R_l(t)$ regulator (4.3.1) proposed for the backlogged source system. In the simulation, we fix the buffer size as $q_M = 5$ and the utility constraint (equivalent to data rate constraint) as $d_l = 0.1, \forall l \in L$. By increasing $V$, the proposed ALG approaches the optimal throughput as stated in Remark 6.5.1, while the minimum data rate requirements are satisfied and the delay remains almost unaffected.

However, a large value of $V$ has a negative effect on the convergence rates of virtual queues. Note that with a large $V$, the term $-V f_l(R_l(t))$ will dominate the $R_l(t)$ regulator (4.3.1) when the virtual queue $Q_l(t)$ is comparatively small, and hence the convergence rate of the system will be very slow: $R_l(t)$ will be set as $R_l(t) = v_M$ for $t = 1, 2, ...$ until $Q_l(t)$ grows large enough such that $R_l(t)(\frac{q_M - \mu_M}{q_M}Q_l(t) - Z_l(t))$ is approximately in the same order as $V f_l(R_l(t))$. The virtual queue evolutions over time are shown in Figure 4.4. We observe that a larger $V$ indeed indicates a slower convergence rate of virtual queues. For $V = 100$, the per-link average of virtual queue backlog $Q_l(t)$ and $Z_l(t)$ constantly grow over the observed simulation duration and the data rate constraints cannot be met within the simulation time of $10^5$ time slots, although a convergence would be achieved as $t \to \infty$. Nevertheless, we observe in
Figure 4.4: Virtual queue evolutions

Figure 4.6 that $V = 20$ and $V = 40$ are sufficient to achieve more than 95% and 99% of the optimal throughput of the network, respectively.

4.7.2 Comparative Performance Evaluation

In this subsection, based on the simulation setup in Section 4.7.1, we compare ALG with two other algorithms: (i) A distributed throughput-optimal algorithm [28], with link weights in the scheduler chosen as $w_l(t) = \frac{\log(U_l(t))}{\log(e + \log(1 + U_l(t)))}$, which has been shown in simulation to achieve better delay performance than the throughput-optimal algorithms in [51][85]. Since it is observed in [28] that this algorithm is similar in throughput and delay performance with the Q-CSMA [83], we denote it by QCSMA. (ii) A traditional CSMA algorithm which employs RTS/CTS handshake to reserve the channel. In the following, we denote this algorithm as CSMA.

Note that for both QCSMA and CSMA, we employ the same congestion controller (4.3.2) in ALG with buffer size denoted as $q_M$ for fairness in the comparison.
Figure 4.5 shows the throughput and delay performances by varying the buffer size $q_M$ and assuming backlogged sources, where we measure packet delay in time slots and throughput in packets per time slot. In the simulation, we choose $V = 50$ and $d_l = 0.1, \forall l \in \mathcal{L}$, for $\text{ALG}$. By increasing $q_M$, i.e., by allowing more packets into the queues, the throughput increases closer to the optimality with a tradeoff of delay performance. For a given $q_M$ value, $\text{ALG}$ and QCSMA have similar delay performances, but the throughput of $\text{ALG}$ significantly outperforms that of QCSMA. As an illustration, with $q_M = 5$ $\text{ALG}$ can achieve a throughput of 1.9924, which is 11.8% more than the throughput of 1.7825 for QCSMA. The remaining unused channel opportunity (the optimality minus the throughput) of QCSMA is 28.6 times that of $\text{ALG}$. In addition, each link can meet the minimum data rate requirement of 0.1 under $\text{ALG}$.

We observe in Figure 4.5 that the CSMA algorithm has the worst delay performance, since its scheduler does not utilize any queue backlog information, an indicator of congestion level of links. CSMA also has the worst throughput performance since links are not scheduled for transmission in CSMA when their RTS/CTS handshakes collide, which is in sharp contrast with Proposition 4.3.2, where we show that the scheduler of $\text{ALG}$ can approach a maximal weight matching with high probability.

In Figure 4.6, we compare the delay performances of $\text{ALG}$ and QCSMA as a function of throughput. In the simulation, we set $q_M = 5$ for $\text{ALG}$, and we increase the simulation time to $10^7$ to obtain reliable results for QCSMA with $q_M = 100000$. We observe that QCSMA requires a much larger buffer size (and hence a significantly larger average packet delay) to achieve comparable throughput levels as $\text{ALG}$. $\text{ALG}$ can operate at higher throughput rates by only adjusting the $V$ parameter, which does not affect the delay performance. Note that a larger value of $V$ in $\text{ALG}$ only affects the convergence rate of virtual queues and we have shown in Figure 4.4 that
Figure 4.5: Algorithm comparison with different buffer sizes

$V \leq 50$ is a feasible choice for the considered system. Consequently, we have shown that $\textit{ALG}$ can achieve close-to-optimal throughput with significantly more favorable delay with respect to QCSMA.

4.7.3 Simulation Comparison in Grid Topology

As pointed out in [72], CSMA algorithms such as QCSMA suffer from “temporal starvation” problem\(^3\) which can lead to large delays in network topologies such as grid and torus ones. In this subsection, we consider a grid topology shown in Figure 4.7 with node-exclusive interference model. We will show that $\textit{ALG}$ does not suffer from large delays as much as QCSMA in the grid topology setting. We choose the link weights as $w_l = \frac{0.1}{q_M} U_l(t) Q_l(t)$ and fix $v_M = \mu_M = 2$. We consider throughput as

\(^3\)It may take a long time to switch from one maximal weight schedule to another.
the utility metric, i.e., $f_l(x) = x$, $\forall l \in L$. With the link notations in Figure 4.7, we set the minimum throughput constraints as $d_l = 0.2$ for $l = 1, 3, 28, 30$; $d_l = 0$, for $l = 6, 7, 8, 24, 25, 26$; and $d_l = 0.1$ otherwise.

In Figure 4.8, we observe that $ALG$ results in a far better delay performance than QCSMA to achieve comparable throughput levels. Similar to the results discussed in Section 4.7.2, a larger value of $V$ improves the throughput performance while not increasing the delay. In addition, the minimum throughput constraints are met for all individual links. As a conclusion, we show through numerical results that $ALG$ employing virtual queues can reduce delays in grid topologies.
4.8 Conclusions

In this chapter, we proposed a cross-layer utility-optimal scheduling algorithm with finite buffers that guarantees individual link-based/flow-based minimum utility constraints. The finite buffer size of packet queues implies upper-bounded average packet delay in the network and is shown to be of order $O\left(\frac{1}{\epsilon}\right)$, where $\epsilon$ characterizes the distance between the achieved utility and the optimal value. The distributed algorithm can be implemented via CSMA/CA methods and is shown to achieve close to optimal throughput and good delay performance both in simulation and numerical results. In our future work, we will implement the proposed multi-hop extension of the algorithm and analyze its performance and scalability.
Figure 4.8: Simulation comparison for constantly backlogged sources: delay versus throughput in grid topology
CHAPTER 5

$V(T)$ REGULATED CSMA ALGORITHM

5.1 Introduction

Recently, a class of distributed queue-length-based CSMA algorithms have been proposed in the literature [51, 83, 85, 28] that achieve throughput optimality, which we refer to as regular throughput-optimal CSMA algorithms in the following discussion. Although these CSMA algorithms have been proved to be throughput-optimal [83][85], they suffer from the following problems: (1) Temporal starvation, defined in [61] as the phenomenon of links “being starved for prolonged periods indefinitely often despite having good stationary throughput”. In other words, links usually undergo prolonged periods of inactivity followed by a prolonged period of activity. Temporal starvation leads to bursty service and undesirable jitter performance. The reason for this behavior is the operation of regular throughput-optimal CSMA algorithms: These algorithms schedule a link that was already active with high probability for prolonged periods, even if there are few (or even no) packets in its queue, during which its neighboring links suffer from starvation. (2) Undesirable delay performance [83][117]. This behavior of regular throughput-optimal CSMA algorithms also leads to the scheduling of links with short queues while there exist unscheduled links with longer queues in the network, resulting in long average packet delays.

There are a limited number of works analyzing the delay and temporal starvation
problems in the literature. It has been shown in [48] that regular throughput-optimal CSMA algorithms achieve polynomial delay upper-bound for a fraction of capacity region in networks with single-hop transmissions. The effect of number of channels on temporal starvation is analyzed in [61]. Congestion control using virtual queues has been proposed in [117] in an attempt to reduce delay without addressing the temporal starvation problem. An unlocking mechanism has been proposed in [72] to mitigate temporal starvation and achieve order-optimal delay in lattice and torus topologies. However, it is difficult to generate the results to general topologies. In [61], the authors show that the temporal starvation problem can be mitigated with multiple frequency agility (i.e., by increasing the number of channels), but the fundamental problem of temporal starvation in a single-channel case is not improved.

Under the VMC-CSMA algorithm in [44], the temporal starvation problem is avoided via a virtual multi-channel approach and a congestion controller. Specifically, there are $C$ virtual channels under VMC-CSMA. Over each virtual channel, the virtual schedule updates similar to a Glauber dynamics system [83][85]. Then a virtual channel is chosen by all communication links uniformly at random from the $C$ virtual channels, and the actual schedule follows the schedule of the chosen virtual channel.

The key difference between the VMC-CSMA and the regular throughput-optimal CSMA is that a link’s transmission probability is a function of its own utility under VMC-CSMA (correspondingly its own queue length under regular CSMA). There is a tradeoff between the number of virtual channels of order $O\left(\frac{\log L}{\epsilon^2}\right)$ and the $\epsilon$-close optimality, where $L$ is the number of communication links. The expected delay is shown to be upper-bounded by a constant independent of network size.

To address the delay and temporal starvation issues, in [111], we propose a $v(t)$-regulated CSMA algorithm that achieves fully local implementation without global message passing. Under the proposed algorithm, only links with weights above a
certain threshold qualify to be scheduled. Since link weights are increasing functions of packet queue lengths, resources are scheduled to links with sufficiently large queue lengths only. This approach potentially alleviates the “persistent” scheduling problem of regular throughput-optimal CSMA algorithms. Compared to those algorithms, the $v(t)$-regulated CSMA algorithm possesses the following two salient features: (1) Links with larger queue lengths are scheduled. By favoring longer queues over shorter ones, delays in longer queues are potentially reduced, and this reduction outweighs the increase in the delay of packets in the shorter, unserved queues. Thus, the average delay is potentially reduced. (2) The change in link schedules is more frequent. When an active link does not have sufficient data packets, the $v(t)$-regulated CSMA algorithm requires the link to relinquish the wireless resource (i.e., channel). By handing over the channel much earlier, i.e., when the packet queue length drops below a threshold, the $v(t)$-regulated CSMA algorithm ensures a faster and more frequent switch between schedules, mitigating the temporal starvation problem.

While achieving an improvement on delay and temporal starvation, we prove that the $v(t)$-regulated CSMA algorithm is throughput-optimal, as well. The proof is based on the time-scale separation assumption (i.e., the Markov chain of the schedules chosen by the scheduler is in steady state in each time slot) which has been employed in [51][83] and verified in [84][50]. Furthermore, our proposed CSMA algorithm is shown via both hardware implementation and simulations to have a much more favorable delay performance than a regular throughput-optimal CSMA algorithm [83] for the same set of arrival rate vectors. The temporal starvation problem is also shown to be mitigated significantly, where we use the second moment of inter-service intervals as the metric to characterize the degree of temporal starvation. There are two major differences between $v(t)$-regulated CSMA and VMC-CSMA [44]. First, the algorithm considered in [44] only works in a closed-loop system, i.e., there is a window-based
congestion controller with a constantly backlogged source. Packet delay is defined to start from the time the packet is admitted into the network by the congestion controller to the time it’s serviced. On the other hand, we consider an open-loop system in this work. That is, there is an arrival process at the source, and packet delay is counted from the external arrival of a packet. Typical applications in closed-loop system include variable-rate multimedia encoder, and applications in open-loop system include real-time wireless services such as VoIP/real-time gaming. Second, like regular throughput-optimal CSMA algorithms, \( v(t) \)-regulated CSMA algorithm achieves distributed implementation with \( O(1) \) complexity, while the complexity of VMC-CSMA grows logarithmically with the network size.

The rest of the work is organized as follows: We propose the \( v(t) \)-regulated CSMA algorithm and present its theoretical performance analysis in Section 5.2. Further discussions are provided in Section 5.3. Specifically, a method to approach timescale separation is presented in Section 5.3.1, and we provide a guideline on choosing thresholds for the proposed algorithm in Section 5.3.3. We present the implementation results and numerical results in Section 5.4 and Section 5.5, respectively. We conclude our work in Section 5.6.

5.2 \( v(t) \)-Regulated CSMA Algorithm

We introduce the network model in Section 5.2.1, with the proposal and theoretical performance analysis of the \( v(t) \)-regulated CSMA algorithm presented in Sections 5.2.2 and 5.2.3, respectively.

5.2.1 Network Model

Consider a wireless network with network topology \((\mathcal{N}, \mathcal{L})\), where \(\mathcal{N}\) denotes the node set and \(\mathcal{L}\) denotes the set of single-hop directional communication links with
Each \( l \in \mathcal{L} \) can be represented by \( l = (m, n) \) as a single-hop flow from source \( m \) to destination \( n \), for some \( m, n \in \mathcal{N} \). We consider a general conflict graph interference model \([49, 51, 83, 85, 28]\). Specifically, for each link \( l \in \mathcal{L} \), we define an interference set \( \mathcal{N}_l \subseteq \mathcal{L} \), such that link \( l \) cannot transmit simultaneously with any link in \( \mathcal{N}_l \). Without loss of generality, we let \( l \in \mathcal{N}_l, \forall l \in \mathcal{L} \), and assume symmetric interference: \( j \in \mathcal{N}_i \) if and only if \( i \in \mathcal{N}_j \), \( \forall i, j \in \mathcal{L} \). A set of links \( x \subseteq \mathcal{L} \) is called an independent set if none of the links in \( x \) interfere with each other, i.e., \( i \notin \mathcal{N}_j \), \( \forall i, j \in x \) with \( i \neq j \). We denote the set of all independent sets by \( \mathcal{I} \) associated to the network topology \((\mathcal{N}, \mathcal{L})\).

We assume the considered wireless system is time-slotted, as is typical in many wireless standards (such as WLANs). We also assume that the transmission rate of each link is normalized and takes value in \( \{0, 1\} \). Let \( A_l(t) \) be the arrival process of the communication link \( l \in \mathcal{L} \) over time slots \( t \). For analytical simplicity, we assume the arrival process \( A_l(t) \) is independent across \( l \in \mathcal{L} \) and i.i.d. over time slots \( t \) with mean \( \lambda_l \).\(^1\) Without loss of generality, we assume that \( A_l(t) \) is upper-bounded by some constant \( A_M, \forall l \in \mathcal{L} \). At each time slot \( t \), we represent a schedule by a vector \( \mu(t) \triangleq (\mu_l(t))_{l \in \mathcal{L}} \), with \( \mu_l(t) \in \{0, 1\} \) denoting the link rate schedule for link \( l \in \mathcal{L} \). A schedule is said to be feasible if \( \sum_{j \in \mathcal{N}(\setminus \{l\})} \mu_j(t) = 0 \), for any \( l \in \mathcal{L} \) with \( \mu_l(t) = 1 \). Thus, when we associate each link \( l \in \mathcal{L} \) with a packet queue length \( Q_l(t) \), the corresponding queue dynamics can be written as:

\[
Q_l(t+1) = [Q_l(t) - \mu_l(t)]^+ + A_l(t), \ \forall t \geq 0 \tag{5.2.1}
\]

where the operator \([\cdot]^+ = \max\{0, \cdot\}\) and we assume that the arriving packets \( A_l(t) \)

\(^1\)We note that the analysis can be readily extended to the case when \( A_l(t) \) are Markovian over time.
are admitted to the packet queue at the end of each time slot $t$. The queue dynamics (5.2.1) can be equivalently represented as:

$$Q_l(t + 1) = Q_l(t) - \mu_l(t) + A_l(t) + \beta_l(t),$$

(5.2.2)

where $\beta_l(t) \triangleq (\mu_l(t) - Q_l(t))1_{\{Q_l(t) < \mu_l(t)\}}$ denotes the unused service for link $l$ at time slot $t$, with $1_{\{E\}}$ being an indicator function of the event $E$.

### 5.2.2 $v(t)$-Regulated CSMA Algorithm and its Distributed Implementation

Central to our proposed $v(t)$-regulated CSMA algorithm is the establishment of a vector of thresholds $(\eta_l)_{l \in \mathcal{L}}$, such that, if the packet queue of an active link $l$ has a link weight below threshold $\eta_l$, the active link $l$ relinquishes the wireless resource and becomes idle. Since link weights are increasing functions of packet queue lengths, only links with sufficiently large queue lengths are scheduled. In comparison, under regular throughput-optimal CSMA algorithms, when a link occupies the channel, even if it has few packets (or even no packets) in its queue, it is highly likely that this link will remain scheduled for a considerably long period of time. By always scheduling links with sufficiently large queue lengths, the $v(t)$-regulated CSMA algorithm potentially results in a reduction of packet delays in these scheduled queues, which outweigh the increase in delay of the packets in the other unscheduled queues (which have fewer packets). Hence, the $v(t)$-regulated CSMA algorithm potentially reduces the average delay. In addition, under the proposed algorithm, the switch between schedules becomes more frequent than under regular throughput-optimal CSMA algorithms, mitigating the temporal starvation.

In the following, we introduce definitions necessary for our CSMA algorithm. We
first define the indicator variable \( v_l(t) = 1_{\{w_l(t) > \eta_l\}}, \ l \in \mathcal{L} \), where \( w_l(t) \) is the link weight of \( l \in \mathcal{L} \) and \( \eta_l \) is the algorithm designed threshold for link \( l \). We denote \( v(t) \triangleq \{v_l(t)\}_{t \in \mathcal{L}} \). Since we require that the \( v(t) \)-regulated CSMA algorithm only schedule links \( l \) with link weights \( w_l(t) \) larger than threshold \( \eta_l \) (i.e., \( v_l(t) = 1 \)), we first define a \( v(t) \)-regulated network topology \((\mathcal{N}, \mathcal{L}(v(t)))\) generated based on the original topology \((\mathcal{N}, \mathcal{L})\). The \( v(t) \)-regulated link set \( \mathcal{L}(v(t)) \) is defined as:

\[
\mathcal{L}(v(t)) \triangleq \{l \in \mathcal{L} : v_l(t) = 1\},
\]

i.e., \( \mathcal{L}(v(t)) \) is the set of links whose link weights \( w_l(t) \) are greater than the corresponding thresholds \( \eta_l \). Under the \( v(t) \)-regulated topology, we further define the set of all \( v(t) \)-regulated independent sets \( \mathcal{I}(v(t)) \) as:

\[
\mathcal{I}(v(t)) \triangleq \{x \in \mathcal{I} : \forall l \in x, v_l(t) = 1\} \subseteq \mathcal{I}.
\]

In addition, we define \( v(t) \)-regulated interference sets \( \mathcal{N}_l(v(t)), l \in \mathcal{L} \), as follows:

\[
\mathcal{N}_l(v(t)) \triangleq \{l \in \mathcal{N}_l : v_l(t) = 1\}, \ \forall l \in \mathcal{L}.
\]

In Figure 5.1, we propose the \( v(t) \)-regulated CSMA algorithm. In Step 1, an independent set \( x(t) \) is selected probabilistically from \( \mathcal{I}(v(t)) \). Links in \( x(t) \) are scheduled in Step 2.1, and other links are scheduled in Step 2.2. Specifically:

- In Step 2.1, for any link \( l \in x(t) \), if its neighboring links in the interference set \( \mathcal{N}_l \) are not scheduled in the previous time slot or do not have a link weight above the threshold, i.e.,

\[
\sum_{j \in \mathcal{N}_l \setminus \{l\}} \mu_j(t-1)v_j(t) = 0,
\]

then link \( l \) is scheduled.
### v(t)-Regulated CSMA Algorithm:

1. Randomly select an independent set $x(t) \in \mathcal{I}(v(t))$ with probability (w.p.) $p_{x(t)}$, such that:

$$\sum_{x(t) \in \mathcal{I}(v(t))} p_{x(t)} = 1$$

and $\bigcup_{p_{x(t)}>0} x(t) = \mathcal{L}(v(t))$.  \(\text{(5.2.3)}\)

2. Scheduling link rates:

   (a) $\forall l \in x(t)$,

   2.1.1 If $\sum_{j \in \mathcal{N}(l)}^{N(l)} \mu_j(t-1)v_j(t) = 0$:

   $$\mu_l(t) = 1, \text{ w.p. } p_l(t) \triangleq \frac{e^{w_l(t)}}{1 + e^{w_l(t)}}, \mu_l(t) = 0, \text{ w.p. } 1 - p_l(t).$$

   2.1.2 Else, $\mu_l(t) = 0$.

   (b) $\forall l \in \mathcal{L} \setminus x(t)$, $\mu_l(t) = \mu_l(t-1)v_l(t)$.

---

**Figure 5.1:** v(t)-regulated CSMA algorithm

service ($\mu_l(t) = 1$) with link activation probability

$$p_l(t) \triangleq \frac{e^{w_l(t)}}{1 + e^{w_l(t)}}.$$  \(\text{(5.2.4)}\)

Otherwise, $\mu_l(t) = 0$.

- In step 2.2, for any link $l$ not belonging to $x(t)$, $\mu_l(t) = 0$ when $v_l(t) = 0$; otherwise, the schedule for link $l$ is unchanged, i.e., $\mu_l(t) = \mu_l(t-1)$.

From the selection of $x(t)$ and Step 2.2, we know that $\mu_l(t) = 1$ only if $v_l(t) = 1$ (i.e., $w_l(t) > \eta_l$). The link weight $w_l(t)$ is defined as $w_l(t) = f(Q_l(t))$, where the link weight function $f : \mathbb{R}^+ \to \mathbb{R}^+$ is chosen as follows: $f(x) = x$ in [49]; $f(x) = \log \log(x + e)$ in [85]; $f(x) = \log(x + 1)$ in [83][28]. It is easy to check that the above choices for function $f$ satisfy the following properties:
• Property (i): \( f \) is an increasing function with \( \lim_{x \to \infty} f(x) = \infty \).

• Property (ii): For any given \( 0 < \epsilon_3 < 1 \), there exists \( Q_M > 0 \) such that for all \( x > Q_M \),

\[
(1 - \epsilon_3) f(x) < f(x - 1) < f(x + A_M) < (1 + \epsilon_3) f(x).
\]

• Property (iii): \( f'(x) \leq 1, \forall x \geq 0 \).

Since \( w_l(t) \) is an increasing function of queue length \( Q_l(t) \), the \( v(t) \)-regulated CSMA algorithm ensures that only links with sufficiently large queues (such that the corresponding link weights \( w_l(t) \) are larger than the thresholds \( \eta_l \)) can be scheduled. Hence, an active link will switch to an idle state when it does not have a sufficiently large number of data packets in its queue, handing the resource over to other links with larger packet queues. On the other hand, under the the regular throughput-optimal CSMA algorithms, even if an active link has few or no packets in its queue, it will continue to occupy the channel for prolonged periods with high probability, during which other links in its interference set suffers from starvation. Thus, with the \( v(t) \)-regulated CSMA algorithm, the problem of temporal starvation is mitigated. Since service is scheduled only to links with sufficiently large queue lengths (i.e., links with packets having potentially large delays), the delay performance is also improved.

In Section 5.4.1, we introduce in detail a distributed implementation of the \( v(t) \)-regulated CSMA algorithm, based on an RTS/CTS handshake mechanism. In the next subsection (Section 5.2.3), we show the throughput-optimality of the proposed algorithm. Implementation results and numerical analysis on delay performance and the temporal starvation issue are provided in Section 5.4.2 and Section 5.5, respectively.
5.2.3 Throughput Optimality of the $v(t)$-Regulated CSMA Algorithm

In this subsection, we prove the throughput optimality of the proposed algorithm in Theorem 5.2.4. We will first show in Proposition 5.2.1 that the $v(t)$-regulated CSMA algorithm always produces feasible schedules.

**Proposition 5.2.1.** The schedule produced by the $v(t)$-regulated CSMA algorithm is feasible, i.e., if $y(t-1) \in I$, then $y(t) \in I$, where $y(t) \triangleq \{l \in \mathcal{L} : \mu_l(t) = 1\}$.

**Proof.** For any $l \in y(t)$, we consider the following two cases:

- If $l \in y(t) \cap x(t)$, where $x(t)$ is the independent set chosen according to Step 1 in Figure 5.1, then $\mu_j(t-1)v_j(t) = 0$, $\forall j \in N_l \setminus \{l\}$, according to Step 2.1. Given any $j \in N_l \setminus \{l\}$, we know that $j \notin x(t)$, since $x(t)$ is an independent set and $l \in x(t)$. If $v_j(t) = 0$, then $\mu_j(t) = 0$ according to Step 2.2. Otherwise (i.e., when $v_j(t) = 1$), $\mu_j(t-1) = 0$, and hence $\mu_j(t) = \mu_j(t-1) = 0$ according to Step 2.2. Therefore, $j \notin y(t)$.

- If $l \in y(t) \setminus x(t)$, then $\mu_l(t-1) = 1$ and $v_l(t) = 1$ according to Step 2.2. For any given $j \in N_l \setminus \{l\}$, we have

\[
\sum_{k \in N_l \setminus \{j\}} \mu_k(t-1)v_k(t) \geq \mu_l(t-1)v_l(t) = 1. \tag{5.2.5}
\]

In addition, $\mu_j(t-1) = 0$, since $\mu_l(t-1) = 1$, i.e., $l \in y(t-1) \in I$.

If $j \in x(t)$, then $\mu_j(t) = 0$ by (5.2.5) and Step 2.1.2. Otherwise (i.e., when $j \notin x(t)$), from Step 2.2, $\mu_j(t) \leq \mu_j(t-1) = 0$. Therefore, $j \notin y(t)$.

Since the above analysis holds for any $l \in y(t)$, we have shown that $y(t) \in I$, i.e., $y(t)$ is an independent set: for any given $l \in y(t)$, we have $j \notin y(t)$, $\forall j \in N_l \setminus \{l\}$. \qed
To support our analysis of the throughput performance, we introduce two related lemmas, Lemma 5.2.2 and Lemma 5.2.3, to assist the proof of throughput optimality in Theorem 5.2.4. Specifically, in Lemma 5.2.2, we show that the schedules produced by the $v(t)$-regulated CSMA algorithm approximate a maximum weight matching scheduler with high probability. In Lemma 5.2.3, we introduce an auxiliary stationary randomized algorithm.

**Lemma 5.2.2.** Under the time-scale separation assumption (the Markov chain of the schedules chosen by the scheduler is in steady state in each time slot), for any given $\epsilon_1$ and $\delta_1$ satisfying $0 < \epsilon_1, \delta_1 < 1$, we can find a constant $B(\epsilon_1, \delta_1) > 0$ such that for any time slot $t$ and with probability greater than $(1 - \delta_1)$, the link rate scheduler finds a schedule $(\mu_l(t))_{l \in \mathcal{L}}$, satisfying:

$$\sum_{l \in \mathcal{L}} w_l(t) \mu_l(t) \geq (1 - \epsilon_1) \max_{x \in \mathcal{I}} \sum_{l \in x} w_l(t), \text{ whenever } ||w(t)||_\infty > B,$$

(5.2.6)

where $w(t) \triangleq (w_l(t))_{l \in \mathcal{L}}$, $||w(t)||_\infty \triangleq \max_{l \in \mathcal{L}} |w_l(t)|$, and

$$B \triangleq \max \left\{ \frac{1}{\epsilon_1} \left( L \log 2 + \log \frac{1}{\delta_1} \right), \frac{2}{\epsilon_1} \max_{x \in \mathcal{I}} \sum_{l \in x} \eta_l \right\}.$$

For notational simplicity, we denote $|| \cdot || \triangleq || \cdot ||_\infty$ in the following discussion. In (5.2.6), $\max_{x \in \mathcal{I}} \sum_{l \in x} w_l(t)$ can be considered as the maximal weight matching over all feasible schedulers.

**Proof.** The proof of Lemma 5.2.2 is provided in Section 5.2.4. \qed

We define the capacity region $\Lambda$ as the set of all arrival rate vectors $(\lambda_l)_{l \in \mathcal{L}}$ supportable by the network, i.e., there exists a feasible scheduling algorithm, centralized or distributed, which is able to stabilize all the packet queues. Then, for any rate
vector in Λ, there exists an (auxiliary) stationary randomized algorithm as stated in Lemma 5.2.3.

**Lemma 5.2.3.** For any rate vector \((\lambda_l)_{l \in \mathcal{L}}\) strictly within the capacity region Λ, i.e., we can find some \(\epsilon_2 > 0\) such that \(((1 + \epsilon_2)\lambda_l)_{l \in \mathcal{L}} \in \Lambda\), there exists a stationary randomized algorithm with schedules \((\mu_l^{STAT}(t))\) independent of the queue lengths \((Q_l(t))_{l \in \mathcal{L}}\), such that, for any time slot \(t\),

\[
\mathbb{E}\{\mu_l^{STAT}(t)\} = (1 + \epsilon_2)\lambda_l, \forall l \in \mathcal{L}.
\]

Similar formulations of randomized algorithm STAT and corresponding proofs have been given in [26][78], so we omit the proof of Lemma 5.2.3 for brevity.

The throughput optimality is concluded in Theorem 5.2.4.

**Theorem 5.2.4.** The \(v(t)\)-regulated CSMA algorithm is throughput-optimal, i.e., the packet queues are stable in the mean [22][28], for any arrival rate vector strictly within the capacity region Λ:

\[
\limsup_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E} \left\{ \left[ \sum_{l \in \mathcal{L}} f^2(Q_l(t)) \right]^{\frac{1}{2}} \right\} \leq \frac{B_2}{\epsilon}, \quad (5.2.7)
\]

where \(B_2 > 0\) and \(0 < \epsilon < 1\) are constants defined as follows:

\[
B_2 \triangleq Lf(A_M) + Lf(Q_M) + Lf(Q_M + A_M)A_M + \gamma BL, \quad (5.2.8)
\]

\[
\epsilon \triangleq \min_{l \in \mathcal{L}}[(\gamma(1 + \epsilon_2) - 1)\lambda_l - \epsilon_3 A_M] > 0,
\]

with \(\gamma \triangleq (1 - \epsilon_1)(1 - \delta_1)\). Note that \(\epsilon_1, \delta_1, B\) are defined in Lemma 5.2.2; \(\epsilon_2\) defined in Lemma 5.2.3; and \(\epsilon_3, Q_M\) defined in Property (ii) of the link weight function \(f\).

**Proof.** The proof of Theorem 5.2.4 is given in Section 5.2.5. \qed
Since \( \epsilon > 0 \) is required to ensure a positive upper-bound in (5.2.7), we must have from definition of \( \epsilon \) that

\[
\gamma > \frac{1}{1 + \epsilon_2},
\]

(5.2.9)

and

\[
\epsilon_3 < \min_{l \in \mathcal{L}} \frac{[\gamma (1 + \epsilon_2) - 1] \lambda_l}{A_M}.
\]

(5.2.10)

Since \( \gamma < 1 \) can be chosen arbitrarily close to 1 (due to the fact that \( \epsilon_1 \) and \( \delta_1 \) can be arbitrarily small according to their definitions in Lemma 5.2.2) and \( \epsilon_3 > 0 \) can be chosen arbitrarily small according to Property (ii) of the link weight function, there exist \( \gamma \) and \( \epsilon_3 \) such that the inequalities (5.2.9) and (5.2.10) hold.

Note that the underlying Markov chain is positive recurrent due to the queue stability in the mean (5.2.7) according to [74], which implies the stability of the network [48].

### 5.2.4 Proof of Lemma 5.2.2

In this subsection, we prove Lemma 5.2.2 introduced in Section 5.2.3. We first show that the \( v(t) \)-regulated CSMA algorithm is equivalent to the scheduler in Figure 5.2 in the following lemma:

**Lemma 5.2.5.** Given \( \mu(t - 1) \), \( w(t) \), and \( v(t) \), the \( v(t) \)-regulated CSMA algorithm is equivalent to the scheduler in Figure 5.2.

**Proof.** Since Step 1 (of selecting an independent set \( x(t) \in \mathcal{L}(v(t)) \)) is the same for both the \( v(t) \)-regulated CSMA algorithm and the equivalent scheduler, proving
Lemma 5.2.5 is equivalently to proving that the link schedules are equivalent under both algorithms given \( x(t) \). For any given \( l \in \mathcal{L} \), we consider the following two cases:

- \( l \in x(t) \): Since
  \[
  \sum_{j \in \mathcal{N} \setminus \{l\}} \mu_j(t-1)v_j(t) = \sum_{j \in \mathcal{N}(v(t)) \setminus \{l\}} \mu_j(t-1),
  \]
  Step 2.1 under the \( v(t) \)-regulated CSMA algorithm is equivalent to Step 2.1 under the equivalent scheduler. Thus, the schedule for \( \mu_l(t), l \in x(t) \), is equivalent under both algorithms.

- \( l \in \mathcal{L} \setminus x(t) \): In this case, \( \mu_l(t) = \mu_l(t-1)v_l(t) \) under the \( v(t) \)-regulated CSMA algorithm. Since
  \[
  \mathcal{L} \setminus \{x(t)\} = (\mathcal{L}(v(t)) \setminus x(t)) \cup (\mathcal{L} \setminus \mathcal{L}(v(t))),
  \]
  we consider the following two subcases under the equivalent scheduler. If \( l \in \mathcal{L}(v(t)) \setminus x(t) \), then according to Step 2.2 under the equivalent scheduler, \( \mu_l(t) = \mu_l(t-1) = \mu_l(t-1)v_l(t) \). Otherwise (i.e., when \( l \in \mathcal{L} \setminus \mathcal{L}(v(t)) \)), according to Step 3 under the equivalent scheduler, \( \mu_l(t) = 0 = \mu_l(t-1)v_l(t) \). Therefore, the scheduler for \( \mu_l(t), l \in \mathcal{L} \setminus x(t) \), is equivalent under both algorithms.

Since the above discussion holds for any given \( l \in \mathcal{L} \), we conclude that the two algorithms are equivalent.

\( \square \)

Step 1 and Step 2 of the equivalent scheduler in Figure 5.2 form the regular throughput-optimal CSMA algorithm \([48]\) with respect to the \( v(t) \)-regulated topology \((\mathcal{N}, \mathcal{L}(v(t)))\). According to Proposition 2 in \([51]\) under the time-scale separation
An equivalent scheduler on \( v(t) \)-regulated topology \((\mathcal{N}, \mathcal{L}(v(t)))\):

1. Randomly select an independent set \( x(t) \in \mathcal{I}(v(t)) \) w.p. \( p_{x(t)} \), such that:

\[
\sum_{x(t) \in \mathcal{I}(v(t))} p_{x(t)} = 1 \quad \text{and} \quad \bigcup_{p_{x(t)}>0} x(t) = \mathcal{L}(v(t)).
\]

2. Schedule link rates in \( \mathcal{L}(v(t)) \):
   
   (a) \( \forall l \in x(t) \),
   
   2.1.1 If \( \sum_{j \in \mathcal{N}_l(v(t)) \setminus \{l\}} \mu_j(t-1) = 0: \)
   
   \( \mu_l(t) = 1 \), w.p. \( p_l(t) \); \( \mu_l(t) = 0 \), w.p. \( 1 - p_l(t) \).
   
   2.1.2 Else, \( \mu_l(t) = 0 \).

   (b) \( \forall l \in \mathcal{L}(v(t)) \setminus x(t) \), \( \mu_l(t) = \mu_l(t-1) \).

3. Schedule link rates in \( \mathcal{L} \setminus \mathcal{L}(v(t)) \):
   
   \( \mu_l(t) = 0 \), \( \forall l \in \mathcal{L} \setminus \mathcal{L}(v(t)) \). Or equivalently, \( \mu_l(t) = 0 \), \( \forall l \in \mathcal{L} \) such that \( w_l(t) \leq \eta_l \).

Figure 5.2: An equivalent scheduler for the proof in Lemma 5.2.2

assumption, for any given \( 0 < \epsilon_1, \delta_1 < 1 \), we can find \( B_1(\epsilon_1, \delta_1) > 0 \) such that, with probability greater than \( (1 - \delta_1) \), the equivalent scheduler (and hence the \( v(t) \)-regulated CSMA algorithm according to Lemma 5.2.5) schedules \( (\mu_l(t))_{l \in \mathcal{L}} \) satisfying the following:

\[
\sum_{l \in \mathcal{L}} w_l(t) \mu_l(t) \geq (1 - \frac{\epsilon_1}{2}) \max_{x \in \mathcal{I}(v(t))} \sum_{l \in x} w_l(t), \quad \text{whenever} \quad ||w(t)|| > B_1,
\]

\[
= (1 - \frac{\epsilon_1}{2}) \max_{x \in \mathcal{I}} \sum_{l \in x} w_l(t)v_l(t),
\]

where \( B_1 \triangleq \frac{1}{\epsilon_1} \left( L \log 2 + \log \frac{1}{\delta_1} \right) \).

153
When \( \| w(t) \| > B \), we have:

\[
\frac{\epsilon_1}{2} \max_{x \in I} \sum_{l \in x} w_l(t) \geq \frac{\epsilon_1}{2} \| w(t) \| > \frac{\epsilon_1 B}{2} \geq \max_{x \in I} \sum_{l \in x} \eta_l \geq \max_{x \in I} \sum_{l \in x} w_l(t)(1 - v_l(t)),
\]

where the last inequality follows the definition of \( v_l(t) \). Hence, we obtain the following inequality, when \( \| w(t) \| > B \),

\[
(1 - \frac{\epsilon_1}{2}) \max_{x \in I} \sum_{l \in x} w_l(t)
\]

\[
< \max_{x \in I} \sum_{l \in x} w_l(t)(v_l(t) + 1 - v_l(t)) - \max_{x \in I} \sum_{l \in x} w_l(t)(1 - v_l(t))
\]

\[
\leq \max_{x \in I} \sum_{l \in x} w_l(t)v_l(t).
\]

Combining (5.2.11), we know that, w.p. larger than \((1 - \delta_1)\), whenever \( \| w(t) \| > B \),

\[
\sum_{l \in \mathcal{L}} \mu_l(t)w_l(t)
\]

\[
\geq (1 - \frac{\epsilon_1}{2}) \max_{x \in I} \sum_{l \in x} w_l(t)v_l(t)
\]

\[
> (1 - \frac{\epsilon_1}{2})^2 \max_{x \in I} \sum_{l \in x} w_l(t) \geq (1 - \epsilon_1) \max_{x \in I} \sum_{l \in x} w_l(t),
\]

which concludes the proof of Lemma 5.2.2.

### 5.2.5 Proof of Theorem 5.2.4

In this subsection, we provide the proof for Theorem 5.2.4 introduced in Section 5.2.3. We first define the Lyapunov function \( L(Q(t)) \triangleq \sum_{l \in \mathcal{L}} g(Q_l(t)) \), where \( Q(t) \triangleq (Q_l(t))_{l \in \mathcal{L}} \) and \( g'(x) = f(x) \). We denote the corresponding Lyapunov drift as \( \Delta(t) \triangleq \)
\[ \mathbb{E}\{L(Q(t+1)) - L(Q(t))|Q(t)\}. \]

From Taylor’s Theorem, we have the following

\[ \Delta(t) = \sum_{l \in \mathcal{L}} \mathbb{E}\{f(\bar{Q}_l(t))(Q_l(t+1) - Q_l(t))|Q(t)\} \]

\[ = \sum_{l \in \mathcal{L}} \mathbb{E}\{f(\bar{Q}_l(t))\beta_l(t)|Q(t)\} + \sum_{l \in \mathcal{L}} \mathbb{E}\{f(\bar{Q}_l(t))(A_l(t) - \mu_l(t))|Q(t)\}, \]

where \( \bar{Q}_l(t) \) lies between \( Q_l(t) \) and \( Q_l(t+1) \), \( \forall l \in \mathcal{L} \).

Since \( \beta_l(t) = 0 \) if \( Q_l(t) \geq 1 \), we have

\[ f(\bar{Q}_l(t))\beta_l(t) \leq f(Q_l(t+1))1_{\{Q_l(t)=0\}} \leq f(A_M). \]

Consequently,

\[ \mathbb{E}\{\Delta(t)\} \leq Lf(A_M) + \sum_{l \in \mathcal{L}} \mathbb{E}\{f(\bar{Q}_l(t))(A_l(t) - \mu_l(t))\}. \] (5.2.12)

To upper-bound the expectation of the Lyapunov drift \( \mathbb{E}\{\Delta(t)\} \) in (5.2.12), we first find an upper-bound for \( f(\bar{Q}_l(t))(A_l(t) - \mu_l(t)) \). From Property (ii) of the link weight function \( f \), for any given \( \epsilon_3 > 0 \), there exists \( Q_M > 0 \) such that \( \forall Q_l(t) > Q_M \),

\[ (1 - \epsilon_3)f(Q_l(t)) < f(\bar{Q}_l(t)) < (1 + \epsilon_3)f(Q_l(t)). \]

Utilizing the above property, if \( Q_l(t) > Q_M \), we have

\[ f(\bar{Q}_l(t))(A_l(t) - \mu_l(t)) \]

\[ < (1 + \epsilon_3)f(Q_l(t))[A_l(t) - \mu_l(t)]^+ - (1 - \epsilon_3)f(Q_l(t))[\mu_l(t) - A_l(t)]^+ \]

\[ = f(Q_l(t))(A_l(t) - \mu_l(t)) + \epsilon_3f(Q_l(t))[A_l(t) - \mu_l(t)] \]

\[ \leq f(Q_l(t))(A_l(t) - \mu_l(t)) + \epsilon_3A_Mf(Q_l(t)). \]
Hence, \( f(\tilde{Q}_i(t))(A_i(t) - \mu_i(t)) \) can be bounded from above as follows:

\[
\begin{align*}
f(\tilde{Q}_i(t))(A_i(t) - \mu_i(t)) \\
\leq f(\tilde{Q}_i(t))(A_i(t) - \mu_i(t))1_{\{Q_i(t) > Q_M\}} + f(\tilde{Q}_i(t))(A_i(t) - \mu_i(t))1_{\{Q_i(t) \leq Q_M\}} + \epsilon_3 A_M f(Q_i(t)) \\
\leq f(\tilde{Q}_i(t))(A_i(t) - \mu_i(t)) + f(Q_i(t))(\mu_i(t) - A_i(t))1_{\{Q_i(t) \leq Q_M\}} \\
\quad + \epsilon_3 A_M f(Q_i(t)) + f(Q_M + A_M) A_M \\
\leq f(\tilde{Q}_i(t))(A_i(t) - \mu_i(t)) + \epsilon_3 A_M f(Q_i(t)) + f(Q_M) + f(Q_M + A_M) A_M,
\end{align*}
\]

which leads to the following inequality:

\[
\mathbb{E}\{\Delta(t)\} \leq Lf(A_M) + Lf(Q_M) + Lf(Q_M + A_M) A_M \\
+ \sum_{i \in \mathcal{L}} \mathbb{E}\{f(Q_i(t)) (\epsilon_3 A_M + A_i(t))\} - \sum_{i \in \mathcal{L}} \mathbb{E}\{f(Q_i(t))\mu_i(t)\}. \tag{5.2.13}
\]

The last term of the RHS of the inequality (5.2.13) can be upper-bounded by

\[
\begin{align*}
- \sum_{i \in \mathcal{L}} \mathbb{E}\{f(Q_i(t))\mu_i(t)\} \\
= - P(||w(t)|| > B) \sum_{i \in \mathcal{L}} \mathbb{E}\{w_i(t)\mu_i(t)||w(t)|| > B\} \tag{5.2.14} \\
- P(||w(t)|| \leq B) \sum_{i \in \mathcal{L}} \mathbb{E}\{w_i(t)\mu_i(t)||w(t)|| \leq B\} \\
\leq - \gamma P(||w(t)|| > B) \mathbb{E}\{\max_{x \in \mathcal{L}} \sum_{i \in \mathcal{L}} w_i(t)||w(t)|| > B\} \tag{5.2.15} \\
\leq - \gamma P(||w(t)|| > B) \sum_{i \in \mathcal{L}} \mathbb{E}\{w_i(t)\mu_i^{STAT}(t)||w(t)|| > B\} \\
= - \gamma \sum_{i \in \mathcal{L}} \mathbb{E}\{w_i(t)\mu_i^{STAT}(t)\} + \gamma P(||w(t)|| \leq B) \sum_{i \in \mathcal{L}} \mathbb{E}\{w_i(t)\mu_i^{STAT}(t)||w(t)|| \leq B\} \\
\leq - \gamma \sum_{i \in \mathcal{L}} \mathbb{E}\{w_i(t)\mu_i^{STAT}(t)\} + \gamma BL,
\end{align*}
\]

where we have employed Lemma 5.2.2 to (5.2.14) and substituted in (5.2.15) the stationary randomized algorithm STAT defined in Lemma 5.2.3.
Employing the above result to (5.2.13), we have

\[ \mathbb{E}\{\Delta(t)\} \leq B_2 + \sum_{l \in \mathcal{L}} \mathbb{E}\left\{ f(Q_l(t)) \left[ \epsilon_3 A_M + A_l(t) - \gamma \mu_l^{ST,AT}(t) \right] \right\} \]

\[ \text{(5.2.16)} \]

\[ = B_2 + \sum_{l \in \mathcal{L}} \mathbb{E}\left\{ f(Q_l(t)) \left[ \epsilon_3 A_M + \lambda_l - \gamma \lambda_l(1 + \epsilon_2) \right] \right\} \]

\[ \leq B_2 - \epsilon \sum_{l \in \mathcal{L}} \mathbb{E}\{f(Q_l(t))\}, \]

where we have employed Lemma 5.2.3 to (5.2.16).

Taking the time-average over \( t = 0, 1, ..., T - 1 \) of both sides of (5.2.16) and taking the limsup with respect to \( T \), we conclude

\[ \limsup_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left\{ \left( \sum_{l \in \mathcal{L}} f^2(Q_l(t)) \right)^{\frac{1}{2}} \right\} \]

\[ \leq \limsup_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{ \sum_{l \in \mathcal{L}} f(Q_l(t)) \} \leq \frac{B_2}{\epsilon}, \]

which proves (5.2.7).

### 5.3 Further Discussions

In this section, we provide further discussions on the implementation issues of the \( v(t) \)-regulated CSMA algorithm. Specifically, a modified \( v_T(t) \)-regulated CSMA algorithm is proposed to approach time-scale separation in Section 5.3.1 and a feasible choice of thresholds is introduced in 5.3.3.

#### 5.3.1 Approaching Time-Scale Separation

We recall that Lemma 5.2.2 (and hence Theorem 5.2.4) in Section 5.2.3 is based on the time-scale separation assumption. This assumption requires the schedule determined by the algorithm to converge to its steady state faster than the rate at which link weights \( w_l(t) \) change over time. In this section, we propose a method, referred to as \( v_T(t) \)-regulated
\( v_T(t) \)-Regulated CSMA algorithm:

1. Randomly select an independent set \( x(t) \in \mathcal{I}(v_T(t)) \) w.p. \( p_{x(t)} \), such that:

\[
\sum_{x(t) \in \mathcal{I}(v_T(t))} p_{x(t)} = 1 \text{ and } \bigcup_{p_{x(t)}>0} x(t) = \mathcal{L}(v_T(t)).
\]

2. Scheduling link rates:

\( \forall l \in x(t) \),

2.1.1 If \( \sum_{j \in N \setminus \{l\}} \mu_j(t-1)v_{j,T}(t) = 0 \):

\( \mu_l(t) = 1 \), w.p. \( p_{l,T}(t) = \frac{e^{w_{l,T}(t)}}{1 + e^{w_{l,T}(t)}} \); \( \mu_l(t) = 0 \), w.p. \( 1 - p_{l,T}(t) \).

2.1.2 Else, \( \mu_l(t) = 0 \).

\( \forall l \in \mathcal{L} \setminus x(t) \), \( \mu_l(t) = \mu_l(t-1)v_{l,T}(t) \).

Figure 5.3: \( v_T(t) \)-regulated CSMA algorithm

CSMA algorithm, to approximate this time-scale separation by updating the link weights less frequently.

The \( v_T(t) \)-regulated CSMA algorithm is illustrated in Figure 5.3. Specifically, we make the Markov chain of the schedules converge to the steady state distributions by updating the link weights \( w_{l,T}(t) \) in the \( v_T(t) \)-regulated CSMA algorithm periodically every \( T \) time slots, \( l \in \mathcal{L} \),

\[
w_{l,T}(t) = f(Q_l(kT)), \ kT \leq t < (k+1)T,
\]

where \( T \) denotes the update period and \( k \) takes integer values. Similarly, we also update the indicator vector periodically as \( v_T(t) = (v_{l,T}(t))_{l \in \mathcal{L}} \) with

\[
v_{l,T}(t) = v_l(kT), \ kT \leq t < (k+1)T, \ l \in \mathcal{L}.
\]
We note that the link activation probability is redefined as

\[ p_{l,T}(t) \triangleq \frac{e^{w_{l,T}(t)}}{1 + e^{w_{l,T}(t)}} \]

The throughput-optimality still holds with the modified algorithm, which is formally stated in the following theorem.

**Theorem 5.3.1.** The \( v_T(t) \)-regulated CSMA algorithm is throughput-optimal.

The proof of Theorem 5.3.1 is provided in following section.

### 5.3.2 Proof of Theorem 5.3.1

We utilize the following two lemmas to prove Theorem 5.3.1.

**Lemma 5.3.2.** Given \( kT \leq t < (k + 1)T \) and any \( \epsilon_4 > 0 \), the following inequality holds

\[
(1 + \epsilon_4) \max_{x \in \mathcal{I}(v(kT))} \sum_{l \in x} w_l(t) \geq \max_{x \in \mathcal{I}(v(t))} \sum_{l \in x} w_l(t),
\]

when

\[
\|w(t)\| > \frac{1}{\epsilon_4} \max_{x \in \mathcal{I}} \sum_{l \in x} f(f^{-1}(\eta_l) + (T - 1)A_M).
\]  

(5.3.2)

**Proof.** Let \( t = kT + i \), \( 0 \leq i < T \). We can bound \( Q_l(t) \) by the following

\[
Q_l(kT) - i \leq Q_l(t) \leq Q_l(kT) + iA_M,
\]

(5.3.3)

according to the queue dynamics (5.2.2). For analytical simplicity, we denote the following independent sets

\[
x_1 \triangleq \arg \max_{x \in \mathcal{I}(v(kT))} \sum_{l \in x} w_l(t),
\]
\[ x_2 \triangleq \arg \max_{x \in \mathcal{L}(v(t))} \sum_{l \in x} w_l(t). \]

Since

\[ x_2 = (x_2 \cap \mathcal{L}(v(kT))) \cup (x_2 \setminus \mathcal{L}(v(kT))) \]

and

\[ \sum_{l \in x_2 \cap \mathcal{L}(v(kT))} w_l(t) \leq \sum_{l \in x_1} w_l(t), \]

we obtain that

\[ \sum_{l \in x_2} w_l(t) - \sum_{l \in x_1} w_l(t) \leq \sum_{l \in x_2 \setminus \mathcal{L}(v(kT))} w_l(t) \leq \sum_{l \in x_2 \setminus \mathcal{L}(v(kT))} f(f^{-1}(\eta_l) + (T - 1)A_M), \]

where we have employed (5.3.3) in the last inequality.

When (5.3.2) holds,

\[ ||w(t)|| > \max_{l \in \mathcal{L}} f(f^{-1}(\eta_l) + (T - 1)A_M), \]

which leads to the following inequality:

\[ \max_{l \in \mathcal{L}} Q_l(t) > \max_{l \in \mathcal{L}} f(f^{-1}(\eta_l) + (T - 1)A_M) \]
\[ \geq Q_j(kT) + (T - 1)A_M \geq Q_j(t), \]

for any \( j \in \mathcal{L} \setminus \mathcal{L}(v(kT)) \). Hence, when (5.3.2) holds, \( \arg \max_{l \in \mathcal{L}} Q_l(t) \in \mathcal{L}(v(kT)) \) and

\[ \sum_{l \in x_1} w_l(t) \geq \max_{l \in \mathcal{L}(v(kT))} w_l(t) = ||w(t)||. \] (5.3.5)
When (5.3.2) is satisfied, we conclude

\[(1 + \varepsilon_4) \sum_{l \in x_1} w_l(t) \geq \sum_{l \in x_1} w_l(t) + \varepsilon_4 ||w(t)|| \]

\[> \sum_{l \in x_1} w_l(t) + \max_{x \in \mathcal{I}} \sum_{l \in \mathcal{X}} f(f^{-1}(\eta_l) + (T - 1)A_M) \geq \sum_{l \in x_2} w_l(t), \]

where the first and the last inequalities follow (5.3.5) and (5.3.4), respectively. This completes the proof of Lemma 5.3.2.

**Lemma 5.3.3.** With the \(v_T(t)\)-regulated CSMA algorithm, under the time-scale separation assumption, for any given \(0 < \varepsilon'_1, \delta_1 < 1\), we can find \(B'(\varepsilon'_1, \delta_1)\), such that, whenever, \(||w(t)|| > B'\)

\[\sum_{l \in \mathcal{L}} w_l(t)\mu_l(t) \geq (1 - \frac{\varepsilon'_1}{2}) \max_{x \in \mathcal{I}(w(t))} \sum_{l \in x} w_l(t). \quad (5.3.6)\]

**Proof.** Similar to the derivation of inequality (5.2.11) in Lemma 5.2.2, under the time-scale separation assumption, we can obtain inequality (5.3.7) through the regular throughput-optimal CSMA algorithm on \(v_T(t)\)-regulated topology \((\mathcal{N}, L(v_T(t)))\). Specifically, given \(0 < \varepsilon'_1, \delta_1 < 1\) and \(\varepsilon_1 = \frac{\varepsilon'^2}{4}\), we can find \(B_1(\varepsilon_1, \delta_1) > 0\) such that w.p. greater than \((1 - \delta_1)\), the \(v_T(t)\)-regulated CSMA algorithm schedules \((\mu_l(t))_{l \in \mathcal{L}}\) satisfy:

\[\sum_{l \in \mathcal{L}} w_{l,T}(t)\mu_l(t) \geq (1 - \frac{\varepsilon'_1}{2}) \max_{x \in \mathcal{I}(w(t))} \sum_{l \in x} w_l(t), \text{ whenever } \max_{l \in \mathcal{L}} w_{l,T}(t) > B_1. \quad (5.3.7)\]

Since \(f'(x) \leq 1, \forall x \geq 0\) (from Property (iii) of the link weight function \(f\)), and

\[f(Q_l(t)) - f(Q_l(kT)) = f'(Q_l')(Q_l(t) - Q_l(kT))\]
where \( Q'_l \) lies between \( Q_l(t) \) and \( Q_l(kT) \), from (5.3.3), we have

\[
w_l(kT) - i \leq w_l(t) \leq w_l(kT) + iA_M, \forall t \in \mathcal{L}.
\] (5.3.8)

From (5.3.8) we obtain that

\[
\sum_{l \in \mathcal{X}_1} w_l(t) \leq \sum_{l \in \mathcal{X}_1} (w_l(kT) + iA_M)
\]

\[
\leq \max_{x \in \mathcal{I}(\nu(kT))} \sum_{l \in \mathcal{X}} w_l(kT) + iA_M \max_{x \in \mathcal{I}} \sum_{l \in \mathcal{L}} 1_{\{l \in x\}},
\] (5.3.9)

and

\[
\sum_{l \in \mathcal{L}} w_l(t)\mu_l(t) \geq \sum_{l \in \mathcal{L}} w_l(kT)\mu_l(t) - i \max_{x \in \mathcal{I}} \sum_{l \in \mathcal{L}} 1_{\{l \in x\}}.
\] (5.3.10)

Applying (5.3.9)(5.3.10) to (5.3.7), we have, w.p. greater than \((1 - \delta_1)\), whenever \( ||w(kT)|| > B_1 \),

\[
\sum_{l \in \mathcal{L}} w_l(t)\mu_l(t)
\]

\[
\geq (1 - \frac{\epsilon_1}{2}) \max_{x \in \mathcal{I}(\nu(kT))} \sum_{l \in \mathcal{X}} w_l,t(t) - i \max_{x \in \mathcal{I}} \sum_{l \in \mathcal{L}} 1_{\{l \in x\}}
\]

\[
\geq (1 - \frac{\epsilon_1}{2}) \sum_{l \in \mathcal{X}_1} w_l(t) - (T - 1) \left(1 + A_M(1 - \frac{\epsilon_1}{2}) \right) \max_{x \in \mathcal{I}} \sum_{l \in \mathcal{L}} 1_{\{l \in x\}}
\] (5.3.11)

\[
\geq (1 - \epsilon_1) \sum_{l \in \mathcal{X}_1} w_l(t), \text{ when } ||w(t)|| > B_3,
\]

\[
=(1 - \sqrt{\epsilon_1})(1 + \sqrt{\epsilon_1}) \sum_{l \in \mathcal{X}_1} w_l(t)
\]

where

\[
B_3 \triangleq \max \left\{ f \left( f^{-1}(B_1) + A_M(T - 1) \right), \frac{2}{\epsilon_1}(T - 1) \left[1 + A_M(1 - \frac{\epsilon_1}{2}) \right] \max_{x \in \mathcal{I}} \sum_{l \in \mathcal{L}} 1_{\{l \in x\}} \right\}.
\]
Note that when \( \|w(t)\| > B_3 \),

\[
\|w(kT)\| \geq f(\|Q(t)\| - iA_M) \\
> f(f^{-1}(B_3) - (T - 1)A_M) \geq B_1,
\]

where we have utilized inequality (5.3.3).

Employing Lemma 5.3.2 to (5.3.11) with \( \epsilon_4 = \sqrt{\epsilon_1} \), we conclude, w.p. greater than \((1 - \delta_1)\), whenever \( \|w(t)\| > B' \),

\[
\sum_{l \in \mathcal{L}} w_l(t) \mu_l(t) \geq (1 - \frac{\epsilon_4}{2}) \sum_{l \in \mathcal{N}_2} w_l(t),
\]

where \( B' = \max\{B_3, \frac{2}{\epsilon_1} \max_{x \in \mathcal{L}} \sum_{l \in x} f(f^{-1}(\eta_l) + (T - 1)A_M)\} \), which completes the proof. \(\square\)

Since the \( v_T(t) \)-regulated CSMA algorithm has the same property ((5.3.6) in Lemma 5.3.3) as the \( v(t) \)-regulated CSMA algorithm ((5.2.6) in Lemma 5.2.2), the proof of Theorem 5.3.1 directly follows that of Theorem 5.2.4 in Section 5.2.5.

### 5.3.3 A Guideline on Choosing Thresholds \((\eta_l)_{l \in \mathcal{L}}\)

The selection of the thresholds \((\eta_l)_{l \in \mathcal{L}}\) is essential to the performance of the \( v(t) \)-regulated algorithm. Since it is extremely hard to find closed-form results on packet delay for queue-length-based CSMA algorithms (though there are a few works in the literature that provide order results, e.g., [48]), instead of finding an optimal threshold that minimizes delay, we provide a guideline on the selection of thresholds \((\eta_l)_{l \in \mathcal{L}}\) in the following. Note that the RHS \( \frac{B_2}{\epsilon} \) of (5.2.7) can be considered as an upper-bound for packet queue lengths, which is an indicator on the delay performance. Thus, we choose the thresholds \( \eta_l, l \in \mathcal{L} \), such that this upper-bound \( \frac{B_2}{\epsilon} \) is minimized. In the following proposition, we introduce such a feasible choice of \((\eta_l)_{l \in \mathcal{L}}\) for the proposed \( v(t) \)-regulated CSMA algorithm.
Proposition 5.3.4. Given $\epsilon_1 = \delta_1 \leq \frac{\epsilon_2}{2(1 + \epsilon_2)^2}$, 
\[
\eta_l = \eta_C \triangleq \frac{(L + 1) \log 2 + \log \frac{1 + \epsilon_2}{\epsilon_2}}{2 \max_{x \in \mathcal{I}} \sum_{l \in \mathcal{L}} 1_{l \in x}}, l \in \mathcal{L},
\]
 guarantees that $\frac{B_2}{\epsilon}$ is minimized.

Proof. According to the definition of $B_2$ in (5.2.8) and the definition of $B$ in Lemma 5.2.2, it is sufficient to prove that the choice (5.3.12) ensures that
\[
\frac{2}{\epsilon_1} \max_{x \in \mathcal{I}} \sum_{l \in \mathcal{X}} \eta_l \leq B_1 = \frac{1}{\epsilon_1} \left( L \log 2 + \log \frac{1}{\delta_1} \right).
\]

For any $l \in \mathcal{L}$, we obtain from (5.3.12) that
\[
\frac{2}{\epsilon_1} \max_{x \in \mathcal{I}} \sum_{l \in \mathcal{X}} \eta_l = \frac{1}{\epsilon_1} \left( (L + 1) \log 2 + \log \frac{1 + \epsilon_2}{\epsilon_2} \right) \leq B_1,
\]
completing the proof. \qed

If local links do not have the knowledge of $\epsilon_2$, which can be considered as the “distance” between the arrival rate vector and the maximal throughput, we can utilize a more conservative (smaller) choice of $\eta_l$:
\[
\eta_l = \frac{(L + 1) \log 2}{2 \max_{x \in \mathcal{I}} \sum_{l \in \mathcal{L}} 1_{l \in x}}, l \in \mathcal{L}.
\]

We note again that the value $\eta_C$ does not necessarily minimize the delay or the expectation of packet queue lengths. Instead, we show in Proposition 5.3.4 that this choice is suboptimal in that it minimizes an upper-bound for queue lengths. Through numerical evaluations presented in Section 5.5, we show that choosing the threshold as $\eta_C$ indeed

\footnote{Since $\epsilon_1$ and $\delta_1$ can be chosen arbitrarily small, the given choice of $\epsilon_1$ and $\delta_1$ is feasible and ensures that the constraint (5.2.9) holds, i.e., $\gamma = (1 - \delta_1)^2 > 1 - 2\delta_1 \geq \frac{1}{1 + \epsilon_2}$.}

164
leads to favorable delay performance and significantly mitigates the temporal starvation compared to a regular throughput-optimal CSMA algorithm.

5.4 Implementations

In this section, we first introduce a distributed implementation of the $v(t)$-regulated CSMA algorithm in Section 5.4.1. We further implement the proposed algorithm in hardware on the Crossbow TelosB platform and present the implementation results in Section 5.4.2.

5.4.1 Distributed Implementation of the $v(t)$-Regulated CSMA Algorithm

In the following, we present a distributed implementation of the $v(t)$-regulated CSMA algorithm. This distributed implementation is based on the RTS/CTS mechanism. Note that the RTS/CTS handshake is only a tool to implement the distributed local interaction of the algorithm and other alternatives may exist. We assume each node has a single transceiver, i.e., a node cannot transmit and receive at the same time. For each node $n \in \mathcal{N}$, we define the following link set: $I_n(t) \triangleq \{(n, i) \in \mathcal{L} : v_{ni}(t) = 1\}$, where $v_{ni}(t)$ denotes $v_l(t)$ with $l = (n, i)$. For initialization, we let $\mu_l(0) = 0$, $\forall l \in \mathcal{L}$.

The main difficulty in the distributed implementation is to randomly select an independent link set satisfying condition (5.2.3). To achieve this, we employ the RTS/CTS mechanism. Specifically, in each time slot, we assign a number of $T_s$ control mini-slots before data transmissions. The CSMA algorithm is executed at each node $n \in \mathcal{N}$ for each time slot $t$, as illustrated in Figure 5.4. At the beginning of each time slot $t$, the node $n$ selects a link $(n, r_n)$ from $I_n(t)$ uniformly at random and chooses a random backoff time from the first $(T_s-1)$ mini-slots (Step 1 in Figure 5.4). Then under the conditions specified in Step 2 in Figure 5.4, node $n$ initiates an RTS (Request-To-Send)-CTS (Clear-To-Send) handshake with its destination node $r_n$ when the backoff duration is over. To facilitate the RTS/CTS handshake, each mini-slot is further split into two micro-slots (Step 2.1 and Step
2.2 in Figure 5.4), dedicated for CTS and RTS transmissions, respectively. After the RT-S/CTS handshakes, the CSMA algorithm assigns link rates (Step 3 in Figure 5.4) following the $v(t)$-regulated CSMA algorithm described in Section 5.2.2. At the end of each time slot $t$, the source node $n$ of any scheduled link $(n, r_n)$ is required to broadcast its updated $v_{(n, r_n)}(t + 1)$ to its neighboring nodes (Step 4 in Figure 5.4).

For narrative clarity, in Figure 5.4 we let RTS($n$) and CTS($n$) denote, respectively, the RTS intended for node $n$ and the CTS intended for node $n$, with $n \in \mathcal{N}$.

It is not difficult to check that under this distributed implementation, any link set $x(t)$ whose links succeed in the RTS-CTS handshake (i.e., Step 2 in Figure 5.4) is an independent link set and $\bigcup_{P(x(t)) > 0} x(t) = \mathcal{L}(v(t))$, satisfying (5.2.3). Thus, the distributed algorithm is equivalent to the $v(t)$-regulated CSMA algorithm introduced in Section 5.2.2.

### 5.4.2 Implementation Results

In this section, $v(t)$-regulated CSMA algorithm is validated in implementation vis-a-vis the QCSMA algorithm [51], a throughput-optimal queue-length-based algorithm. We employ the link weight function $f(x) = \log(x + 1)$, since it is shown via simulation in [51][28] that the log($\cdot$) form yields better delay performance than $f(x) = x$ [49] and $f(x) = \log \log(x + e)$ [85].

To quantify the degree of temporal starvation, we use the second moment of inter-service intervals as our metric. Specifically, $s_l(i)$ denoting the scheduled service time of $i$-th packet of link $l \in \mathcal{L}$, we define the second moment of inter-service intervals $J_l$ for link $l$ as:

$$J_l \triangleq \lim_{I \to \infty} \frac{1}{I} \sum_{i=1}^{I} (s_l(i + 1) - s_l(i))^2.$$

Note that a smaller $J_l$ implies that link $l$ is scheduled more frequently and its idle periods are shorter, indicating a lower level of temporal starvation. For analytical simplicity, in the implementation and simulation evaluation, we refer to the second moment of inter-service intervals as the average of $J_l$ over all links $l \in \mathcal{L}$. 

166
Distributed CSMA Algorithm at node $n$ in time slot $t$

Step 1. Initialization:
A node $r_n$ is selected in $I_n(t)$ uniformly at random;
$n$ chooses a backoff time $b_n$ uniformly at random from the
first $(T_s - 1)$ mini-slots.

Step 2. RTS-CTS handshake:
for mini-slot $t_s = 1 : T_s$ (i.e., $t_s$ is the current mini-slot)

Step 2.1. The first micro-slot: CTS transmission
if $n$ has not overheard in the past any RTS($m$), $\forall m \neq n$
and $n$ has not overheard any collision of RTS before
and $n$ has not sent out a CTS before
and $n$ received only one RTS($n$) in mini-slot ($t_s - 1$)
do: node $n$ sends out a CTS responding to the RTS($n$)
received in mini-slot ($t_s - 1$);
end if

Step 2.2. The second micro-slot: RTS transmission
if $n$ has not overheard in the past any CTS($m$), $\forall m \neq n$
and $n$ has not overheard any collision of CTS before
and $n$ has not sent a CTS before and $b_n = t_s$
do: node $n$ sends out an RTS($r_n$) to node $r_n$;
end if
end for

Step 3. Schedule for links $\{(n, i) : (n, i) \in \mathcal{L}\}$:
$\mu_{ni}(t) = \mu_{ni}(t - 1)v_{ni}(t), \forall i \neq r_n : (n, i) \in \mathcal{L}$.
if $n$ received CTS($n$) from $r_n$ in Step 2 do:
if $\sum_{j \in \mathcal{N}(n, r_n) \setminus \{(n, r_n)\}} \mu_{j}(t - 1)v_{j}(t) > 0$
do: $\mu_{nr_n}(t) = 0$;
else
do: $\mu_{nr_n}(t) = 1$ w.p. $p(n, r_n)$;
$\mu_{nr_n}(t) = 0$ w.p. $(1 - p(n, r_n))$;
end if
else (i.e., $(n, r_n)$ failed the RTS-CTS handshake)
do: $\mu_{nr_n}(t) = \mu_{nr_n}(t - 1)v_{nr_n}(t)$;
end if

Step 4. At the end of time slot $t$:
if $\mu_{nr_n}(t) = 1$ do:
broadcast the updated $v_{(n, r_n)}(t + 1)$ to neighboring nodes,
i.e., nodes $i$ such that there exists some node $j$ with $(i, j) \in \mathcal{N}(n, r_n)$;
end if

Figure 5.4: Distributed implementation of the $v(t)$-regulated CSMA algorithm
Figure 5.5: Implementation results on delay and 2nd moment of inter-service intervals with respect to arrival rate under topology in Figure 4.3(b), given $\eta_l = \eta_C$, $\forall l \in \mathcal{L}$.

We implement the proposed algorithm (introduced in Section 5.4.1) and QCSMA algorithm in hardware on the Crossbow TelosB platform running NanoQplus OS [60]. Each Crossbow TelosB node is equipped with an IEEE 802.15.4 compliant RF transceiver and a programmable MSP430 processor [46]. To maintain the time-slotted structure, nodes periodically exchange timing information every 50 time slots to realign their internal clocks, which is sufficient in this experiment to achieve a data collision rate of less than 0.01%.

We employ a node-exclusive interference model (or one-hop interference model) under the 10-link topology in Figure 4.3(b). We assume an identical arrival rate for all 10 communication links, i.e, $\lambda_l = \lambda$, $l \in \mathcal{L}$. Since at most two links can be active at each time slot under the node-exclusive model, the stabilizable range of $\lambda$ is $0 < \lambda < 0.2$, with $\epsilon_2 = \frac{0.2}{\lambda} - 1$ characterizing the distance between $\lambda$ and the maximum per-link throughput 0.2. The arrival processes $A_l(t)$ follow independent Bernoulli processes with parameter $\lambda$.

Figure 5.5 illustrates the delay and the second moment of inter-service intervals under the two algorithms, with $\eta_l = \eta_C$, $\forall l \in \mathcal{L}$. The implementation results are averaged over 5000 time slots to 90000 time slots for different arrival rates in 5 runs such that the
confidence interval is of acceptable range. When arrival rate is lower than 0.18 (90% of maximum throughput), the packet delay and the second moment of inter-service intervals, both averaged over all 10 links, are much smaller under the \( v(t) \)-regulated CSMA algorithm than the QCSMA algorithm. The difference between the two algorithms in performance diminishes when the arrival rate further increases. This can be explained as follows: When the arrival rate increases towards the maximum (the value of 0.2 in this case), the queue lengths become larger, and hence the value of the threshold \( \eta_C \) (\( \eta_C = 2.7 \) in this implementation setup) becomes negligibly small compared to the queue length. Thus, the behavior of \( v(t) \)-regulated CSMA algorithm approaches the QCSMA algorithm when the arrival rate \( \lambda \) is close to 0.2. However, even in a large-arrival-rate region, the \( v(t) \)-regulated CSMA algorithm still yields much better performance. For instance, at an arrival rate \( \lambda = 0.19 \) (achieving 95% of maximum throughput), the packet delay is 53% smaller under the \( v(t) \)-regulated CSMA algorithm than the QCSMA algorithm; and the second moment of inter-service intervals is 46% smaller.

We also note that under both algorithms, in a small-arrival-rate regime (e.g., \( \lambda = 0.05 \), achieving 25% of the maximum throughput), the inter-service intervals are potentially spaced farther apart because packet arrivals are farther apart, as well. Therefore, the second moment of inter-service intervals decreases when the arrival rate increases initially, as shown in Figure 5.5. When arrival rate further increases, average queue lengths become larger, and it is more likely that an active link occupies the channel for prolonged periods of time followed by prolonged periods of inactivity, leading to a larger second moment of inter-service intervals, i.e., a higher level of temporal starvation.

5.5 Further Numerical Results

In this section, we further present a numerical comparative study of \( v(t) \)-regulated CSMA algorithm with reference to the QCSMA algorithm. In the following two subsections, we
Figure 5.6: Numerical results: Snapshot of Link schedules when arrival rate $\lambda = 0.19$

illustrate the numerical performance using two different interference models under two different network topologies with the link weight function $f(x) = \log(x + 1)$.

5.5.1 Numerical Evaluation in a 10-Link Topology

In this subsection, we consider the same topology of Figure 4.3(b) and the node-exclusive interference model with same arrival processes as in Section 5.4. To complement the implementation results, we further study the transient behavior of the system and the effect of thresholds $(\eta_l)_{l \in L}$ on the algorithm performance.

When arrival rate $\lambda = 0.19$ (i.e., achieving 95% of the maximum stabilizable throughput), we find that all queues are stabilized and the throughput of 1.9 (summed over ten links) is indeed achieved. A snapshot of link rate schedules is shown in Figure 5.6 with the suggested thresholds $\eta_l = \eta_C$, $\forall l \in L$. To deliver a clear picture of instantaneous schedules, we only show the schedules for links (1, 2), (1, 3), (4, 1), and (5, 1) in Figure 5.6. Compared to QCSMA algorithm under which a single link can occupy the channel over hundreds of time slots, the switch of link schedules is much more frequent under the $v(t)$-regulated CSMA algorithm. Thus, the temporal starvation issue is successfully mitigated under the proposed algorithm.
Figure 5.7: Numerical performance of delay and 2nd moment of inter-service interval with respect to threshold $\eta$ under topology in Figure 4.3(b), given arrival rate $\lambda = 0.1$

We now study the effect of thresholds $(\eta_l)_{l \in \mathcal{L}}$ on the performance of the $v(t)$-regulated CSMA algorithm. In Figure 5.7 with arrival rate $\lambda = 0.1$, we show the performance of delay and second moment of inter-service intervals by varying the value of $\eta$, where we let $\eta_l = \eta$, $l \in \mathcal{L}$. The starred point in Figure 5.7 denotes the case with $\eta_l = \eta_C$, $l \in \mathcal{L}$. We observe that $\eta_C$ is indeed a favorable choice for the thresholds $(\eta_l)_{l \in \mathcal{L}}$ in that it leads to comparably good delay performance among all the thresholds. $J_l$ performance could further be improved when $\eta$ grows larger than $\eta_C$. However, we note that the decrease in $J_l$ is trivial compared to the increase in delay as $\eta$ becomes larger.

From Figure 5.7, we see that even a small value of threshold (e.g., $\eta = 1$) can significantly reduce the delay and mitigate the issue of temporal starvation compared to the QCSMA algorithm. However, we notice that, as $\eta$ gets larger, the delay increases. Recall that under $v(t)$-regulated CSMA algorithm, only link weights (increasing functions $f$ of queue lengths) greater than the threshold $\eta$ can be scheduled. Thus, the average queue packet length is greater than or equal to $f^{-1}(\lfloor \eta \rfloor)$, where $\lfloor \cdot \rfloor$ denotes the floor function. When $\eta$ grows
significantly large, \( f^{-1}([\eta]) \) dominates the queue length. Thus, according to the Little’s Law, the average packet delay increases accordingly in a large-\( \eta \)-regime.

We also observe that, when \( \eta \) increases, the duration of a link occupying the channel is expected to become smaller, leading to a more frequent change in schedules. Hence, the second moment of inter-service intervals \( J_l \) becomes smaller in Figure 5.7. However, we note this decrease in \( J_l \) is trivial compared to the increase in delay when \( \eta \) becomes large. Therefore, we consider \( \eta_C \) to be a good choice for the thresholds.

5.5.2 Numerical Comparison in a Grid Topology

It is pointed out in [72] that regular throughput-optimal CSMA algorithms (such as the QCSMA algorithm) suffer from serious temporal starvation in grid/lattice topology. In this subsection, we show that under a grid topology, our proposed \( v(t) \)-regulated CSMA algorithm does not suffer as much from the temporal starvation as the QCSMA algorithm. In fact, our proposed algorithm far outperforms the QCSMA algorithm in both delay and second moment of inter-service intervals. Note that different from the algorithm in [72] which is intended for grid/lattice topology only, our proposed algorithm works for arbitrary network topologies under general conflict graph interference model.

Specifically, we employ a two-hop interference model under the grid topology in Figure 4.7. The arrival rate vector is set as \( \lambda_l = \frac{1}{4} \lambda \) for \( l = 1, 4 \); \( \lambda_l = \frac{1}{6} \lambda \), for \( l = 5, 9, 14, 18, 23, 25, 27, 28, 31 \); and \( \lambda_l = \frac{1}{12} \lambda \), otherwise. The stabilizable range for \( \lambda \) is \( 0 < \lambda < 1 \), since these arrival rates can be represented by a convex combination of 12 maximal matching schedules. The arrival processes \( A_l(t) \) follow independent Bernoulli processes with parameter \( \lambda_l, \forall l \in L \). In addition, we let \( \epsilon_2 = \frac{1}{\lambda} - 1 \) which denotes the distance between the current arrival rates and the maximum per-link throughput.

Similar to the results under the 10-link topology in the previous section, results in Figure 5.8 indicate that the proposed algorithm significantly outperforms the QCSMA algorithm in terms of delay and second moment of inter-service intervals. In Figure 5.9, we show that \( \eta_C \), determined by (5.3.12), is a suitable choice for the threshold, since a larger threshold
Figure 5.8: Numerical performance comparison of delay and 2nd moment of inter-service interval under grid topology in Figure 4.7, with $\eta_l = \eta_C$, $\forall l \in \mathcal{L}$

Figure 5.9: Numerical performance of delay and 2nd moment of inter-service interval with respect to threshold $\eta$ under grid topology in Figure 4.7, when $\lambda = 0.9$
can lead to a large increase in delay with insignificant improvement on the second moment of inter-service intervals.

5.6 Conclusions

In this chapter, we proposed a $v(t)$-regulated CSMA algorithm that can be implemented via a distributed method and achieve optimal throughput in wireless networks with single-hop transmissions. In the algorithm, link scheduling is performed favoring links with sufficiently large queue lengths to reduce average delay and ensure a more frequent switch between schedules. Via both hardware implementation and numerical evaluations, we show that compared to the QCSMA algorithm, the proposed algorithm significantly improves the delay performance and mitigates the problem of temporal starvation.
CHAPTER 6
QUEUEING BEHAVIOR OF CSMA SCHEDULING
UNDER THE MANY-CHANNEL REGIME

6.1 Introduction

Throughput and delay are two important metrics to analyze the quality of service (QoS) in wireless networks. Throughput optimality of the throughput-optimal distributed algorithms has been proved in [51, 83, 85, 28, 48], and their average queue backlog/delay performance has been studied in [48, 72, 11]. Specifically, through a mixing-time analysis, it is shown in [72] that an upper-bound for the time-averaged data queue backlog is of exponential order in the number of communication links in the network, and an upper-bound of polynomial order in the number of links is provided in [48] for a fraction of the capacity region. A lower-bound of expected data queue backlogs has been derived in [11]. However, these (average) delay bounds are not tight in general. In fact, no closed-form analysis/estimation of the queue backlog/delay performance of these random access algorithms is available in the literature, since interactions between links lead to complicated queuing behaviors and make such characterization intractable. Not surprisingly, queuing behavior and delay analysis of multi-channel extensions of these distributed algorithms is non-trivial and remains an open research problem, since the state space of schedules grows exponentially with the number of channels.

Wireless systems with multiple channels, e.g., OFDM systems, have important applications in next-generation networks such as WiMAX, 4G cellular networks, cognitive radio
networks. With the above motivation, we study the transient queuing behavior of multi-channel random access algorithms for a fully connected wireless network (e.g., WLAN and cellular networks). To this end, we develop a novel equivalent queue system to model and show that the queuing behavior of individual links converges to the equivalent queue system as the number of channels grows. Note that transient queuing behavior (including transient evolution of service rates) implies the ability to analyze the transient delay performance in addition to the steady-state study of the queuing behavior and the expected delay performance of the wireless system.

In this chapter, we are especially interested in the behavior of a system with a fixed number of links where the number of channels scales over a constant capacity bandwidth. This approach is motivated and supported by the properties and trends of recently deployed wireless systems such as WiMAX. The physical service area in which high data rates are provided to links is generally limited, which means the number of links is roughly upper bounded (e.g., in WiMAX pico and femto cells [25]), as well. On the other hand, the same systems operate over hundreds of orthogonal channels. So, the focus of our investigation is the behavior of a system with a large number of channels over a given finite frequency band, serving a given constant population size. Note that the growing number of channels represents diminishing bandwidth per channel, where the sum of all bands is constant. This scenario can be justified by the fact that inter-symbol interference (ISI) can be significantly reduced in an OFDM system by transmitting data in parallel over a large number of low-rate (sub)channels [118].

In [107], we first generalize the random access algorithms from a single-channel setting [51, 83, 85, 28, 48] to a multi-channel one, denoted as multi-channel random access algorithm. The multi-channel random access algorithm is shown to be throughput-optimal for any finite number of non-fading channels under the time-scale separation assumption.¹

¹The time-scale separation assumption (i.e., the underlying Markov chain of the scheduling algorithm converges instantaneously to its steady state) has been employed in [51][83] and justified in [84][50].
With the introduction of the novel *equivalent deterministic single-queue system*, the salient features of our work can be listed as follows:

1. Assuming (asymptotically) uniform arrival rates, we show that the queuing behavior of the network asymptotically (with respect to the number of channels) approaches that of the equivalent deterministic single-queue system. Law-of-large-numbers (LLNs) results have been established in the many-channel regime for the queue backlog and the scheduled service rate, which are governed by simple deterministic dynamics. These results are the first of their kind for random access scheduling, which yield a simplified and scalable characterization of the individual queuing behavior in the many-channel regime.

2. By studying the equivalent deterministic single-queue system, we find a *closed-form* approximation for the queue backlog in its steady state in the many-channel regime. While the delay bounds derived in [48, 72, 11] for a single-channel setting are generally not tight, we show that the closed-form steady state queue backlog/delay approximation becomes accurate as the number of channel increases. Furthermore, in resonance with the findings in [24, 11, 27], we show for a fully connected network that the more aggressive the weight function\(^2\) is, the smaller the asymptotic queue length becomes in the many-channel regime.\(^3\)

3. Based on a steady state study, the multi-channel random access algorithm is proved to be asymptotically (with respect to the number of channels) throughput-optimal *without the time-scale separation assumption*.

4. We show that with heterogeneous arrivals, the queuing behavior of the network asymptotically converges in the number of channels to an equivalent multi-queue system. We also derive the corresponding closed-form approximation for the queue backlogs in the steady state in the many-channel regime.

---

\(^2\)Weight functions are employed to determine the transmission probability in the random access algorithms.

\(^3\)Results have been provided in [24], [11], and [27], respectively, on the optimality, a lower-bound on queue backlogs, and the stability for random access algorithms in a single-channel setting. In comparison, we provide a *closed-form* queue backlog approximation for random access algorithms in the many-channel regime.
It is shown in [63] that the random access algorithms [51, 83, 85, 28, 48, 117] are no longer throughput-optimal under a single fading channel. However, in the context of many fading channels, we prove that the multi-channel random access algorithm (with minor modifications) is throughput optimal asymptotically with respect to the number of channels.

Via numerical evaluations, we validate the accuracy of the equivalent queue system representation and show that the queue backlogs/delays and service rates indeed converge to the derived closed-form steady state results in the many-channel regime.

Following the many-channel regime queuing analysis, we also propose a distributed scheduling algorithm for a cognitive radio scenario in [114] that achieves at least a guaranteed fraction of the optimal throughput for secondary users.

The rest of the chapter is organized as follows: We introduce the multi-channel random access algorithm in Section 6.2 and introduce the equivalent queue system that asymptotically represents the queuing behavior of the network under the random access algorithm in Section 6.3. A steady state study is carried out in Section 6.5 that introduces a closed-form approximation for the queue backlogs and service rates under the random access algorithm. We extend our analysis from a uniform arrival rate case to a heterogeneous arrival rate case in Section 6.6. In Section 6.7, we show that the multi-channel random access algorithm is asymptotically throughput-optimal under fading channels. We present the numerical results in Section 6.8 and conclude this work in Section 6.9. Following similar queuing analysis, we propose a distributed scheduling algorithm for a cognitive radio scenario in Section 6.10.

### 6.2 Network Model and the Multi-Channel Random Access Algorithm

For simplicity, we introduce the notations $P \xrightarrow{N} L$ and $L \xrightarrow{N} L$ to denote convergence in probability and convergence in law (or convergence in distribution) [56], respectively, as $N \rightarrow \infty$. 
6.2.1 Network Elements

We consider a time-slotted fully connected single-hop wireless network where \( M \) communication links contend for \( N \) orthogonal channels. Each link \( i \) maintains a data queue \( q_i(t) \) updated at the beginning of a time slot \( t = 0, 1, 2, ..., \) with \( i = 1, 2, ..., M \). Let \( A_i(t) \) be the amount of data (in unit of bits) arriving at queue \( i \) at the beginning of a time slot \( t \). We assume that the number of links does not scale with the number of channels, which typically applies to WiMAX pico-cell, and femto-cell scenarios [25].

We assume that each channel is non-fading and always available with a capacity (i.e., maximum data rate per time slot) \( \frac{C}{N} \), where \( C \) denotes the total capacity of the considered wireless system. The scenario of fading channels is discussed in Section 6.7. We denote by \( \mu_{ij}(t) \in \{0, 1\} \) the schedule of link \( i, i = 1, 2, ..., M \), over channel \( j, j = 1, 2, ..., N \), at time slot \( t \). Specifically, \( \mu_{ij}(t) = 1 \) if link \( i \) is scheduled over channel \( j \); \( \mu_{ij}(t) = 0 \), otherwise. We consider an OFDM mechanism, i.e., one link can transmit over multiple channels in a time slot. Since we have assumed a fully connected network, simultaneous transmissions over the same channel will cause interference at all nodes and, hence, each channel can only be allocated to one link, i.e., \( \sum_i \mu_{ij}(t) \leq 1, \forall t \). Thus, the queue of each link evolves as, \( \forall i \),

\[
q_i(t) = [q_i(t-1) + A_i(t-1) - \frac{C}{N} \sum_{j=1}^{N} \mu_{ij}(t-1)]^+, \forall t \geq 0, \tag{6.2.1}
\]

where \( [\cdot]^+ \triangleq \max\{\cdot, 0\} \).

6.2.2 Multi-Channel Random Access Algorithm

In this section, we introduce a distributed throughput-optimal multi-channel random access algorithm, which is a generalization of the single-channel random access algorithms [83, 85, 28, 48, 117]. For each time slot \( t \), the algorithm is composed of two parts: Exchange Phase and Scheduling Phase.

The exchange phase is scheduled at the beginning of time slot \( t \) and is composed of
M mini-slots reserved for (control) message exchange. Each link is assigned a dedicated mini-slot out of the M mini-slots. Transmissions during mini-slots use the entire spectrum C (i.e., channels \{1, ..., N\}) to minimize the length of mini-slots, which we assume is negligible compared to that of a unit time slot. Specifically, the transmitter of each link \(i\) broadcasts the following three binary vectors to all other nodes (i.e., the receiver of link \(i\), and the transmitters and the receivers of all the other links) during its dedicated mini-slot:

\[
((\mu_{ij}(t-1))_{j=1}^{N}, (a_{ij}(t))_{j=1}^{N}, (p_{ij}(t))_{j=1}^{N}),
\]

where \((\mu_{ij}(t-1))_{j=1}^{N}\) denotes the schedules of link \(i\) at the previous time slot \(t-1\). The contention variables \(a_{ij}(t)\) are independent over link \(i\) and channel \(j\) with

\[
a_{ij}(t) = \begin{cases} 
1, \text{ w.p. } \beta, \\
0, \text{ w.p. } 1 - \beta. 
\end{cases}
\]

The contention probability \(0 < \beta < 1\) is typically chosen as \(\frac{1}{M}\) [29]. The transmission variables \(p_{ij}(t)\) are independent over links \(i\) and i.i.d. over channels \(j\) with

\[
p_{ij}(t) = \begin{cases} 
1, \text{ w.p. } \frac{h(q_{i}(t-1))}{1 + h(q_{i}(t-1))}, \\
0, \text{ w.p. } \frac{1}{1 + h(q_{i}(t-1))}, 
\end{cases}
\]

where we denote the transmission weight function by \(h : [0, \infty) \rightarrow [0, \infty)\). These binary vectors \(((\mu_{ij}(t-1))_{j=1}^{N}, (a_{ij}(t))_{j=1}^{N}, (p_{ij}(t))_{j=1}^{N})\) are received by all links in the single-hop network and used to determine the transmission schedules for individual links. Note that the vector of contention variables \((p_{ij}(t))_{j=1}^{N}\) will be used only by the transmitter \(i\) and its intended receiver in the scheduling phase.

After the exchange phase, the schedule \(\mu_{ij}(t)\), for any given link \(i\) and channel \(j\), is determined in the scheduling phase based on the Glauber dynamics in a single channel setting [83, 85, 28, 48]. Specifically, the schedule \(\mu_{ij}(t)\) for any given link \(i\) and channel \(j\) at time slot \(t\) depends on the following three conditions: Condition (i): The “contention” of link \(i\) for channel \(j\) is successful, i.e., \(a_{ij}(t) \prod_{k \neq i}(1 - a_{kj}(t)) = 1\). Condition (ii): \(\sum_{k \neq i} \mu_{kj}(t-1) = 0\), i.e., no other links were allocated channel \(j\) in the previous time slot. Condition (iii): The transmission variable \(p_{ij}(t) = 1\).
The scheduling phase is performed locally at each node (i.e., the transmitter and the receiver of each link $i$) as follows:

**Scheduling Phase**

The transmitter and the receiver of each link $i$ determine $\mu_{ij}(t)$, $j = 1, ..., N$, according to the following:

- **If** Conditions (i), (ii), and (iii) hold,
  then $\mu_{ij}(t) = 1$ (i.e., channel $j$ is allocated to link $i$ for time slot $t$);
- **Else if** Condition (i) does not hold,
  then $\mu_{ij}(t) = \mu_{ij}(t - 1)$;
- **Otherwise**, $\mu_{ij}(t) = 0$.

From the multi-channel random access algorithm introduced above, we conclude, for any given link $i$ and any given channel $j$,

$$
\mu_{ij}(t) = a_{ij}(t) \prod_{k \neq i} (1 - a_{kj}(t)) \left(1 - \sum_{k \neq i} \mu_{kj}(t - 1)\right) p_{ij}(t) \\
+ \left(1 - a_{ij}(t) \prod_{k \neq i} (1 - a_{kj}(t))\right) \mu_{ij}(t - 1).
$$

(6.2.2)

The first term on the RHS of (6.2.2) corresponds to the case when Conditions (i)(ii)(iii) hold, and the second term on the RHS of (6.2.2) corresponds to the case when Condition (i) does not hold.

Since both the transmitter and the receiver of link $i$ have a copy of the schedule vector $(\mu_{ij}(t))_{j=1}^{N}$ when the scheduling phase ends, they will tune to the set of channels $\{j : \mu_{ij}(t) = 1\}$ for data transmission in the remaining time slot.

In the following theorem, we state the throughput optimality of the above random access algorithm.
Theorem 6.2.1. Assume that the transmission weight function takes the form of $h(x) = e^{f(x)}$ with $f(x)$ satisfying the following two properties: (A1) $f : [0, \infty) \to [0, \infty)$ is non-decreasing and continuous with $\lim_{x \to \infty} f(x) = \infty$. (A2) Given any $M_1, M_2 > 0$ and any $\epsilon' > 0$ arbitrarily small, there exists $M_3 > 0$ such that for all $x > M_3$: $f(x)(1 - \epsilon') \leq f(x-M_1) \leq f(x+M_2) \leq f(x)(1 + \epsilon').$ If the arrival process is stationary and bounded above (i.e., $A_i(t) \leq A_{\text{max}}, \forall t, \forall i$, for some sufficiently large $A_{\text{max}} > 0$), under the time-scale separation assumption [51] (which is employed in [51]/[83] and justified in [84]/[50]), the algorithm is throughput-optimal$^4$ for any given $N \geq 1$.

Proof. Theorem 6.2.1 is proved in Section 6.2.3.

Random access algorithms proposed in [51, 83, 85, 28] have been proved to be throughput-optimal in a single channel setting. However, there are few works analyzing the queue backlog/delay performance for these throughput-optimal algorithms other than the order results [48]/[72] on the upper-bounds and [11] on the lower-bounds of the expected queue backlogs, which are not tight in general. In Section 6.3, we show that the queue backlogs $(q_i(t))$ and service rates asymptotically converge to an equivalent deterministic single-queue system. In Section 6.5, by studying the steady state of the equivalent queue, we find a closed-form approximation for the steady state queue backlogs/delays in the many-channel regime.

6.2.3 Proof of Theorem 6.2.1

To prove Theorem 6.2.1, we first introduce two lemmas.

Lemma 6.2.2. Under the multi-channel random access algorithm in Section 6.2.2, for any

$^4$We say that an algorithm is throughput-optimal if it can stabilize any arrival rate vector within the capacity region (which is the closure of all arrival rate vectors that can be stably supported by the network) [26].
$0 < \epsilon_1, \delta_1 < 1$, there exists $B_1(\epsilon_1, \delta_1) > 0$, such that, $\forall j$, $\forall t$, with probability greater than 
$(1 - \frac{\delta_1}{N})$, whenever $\|Q(t - 1)\|_\infty > B_1,$

$$\sum_{i=1}^{M} f(q_i(t - 1)) \mu_{ij}(t) \geq (1 - \frac{\epsilon_1}{4}) \max_i f(q_i(t - 1)),$$

where $Q(t) = \{q_i(t)\}_{i=1}^{M}$ and $\|(x_i)\|_\infty = \max_i x_i$. For notational simplicity, we let $\| \cdot \|_\infty = \| \cdot \|$.

**Lemma 6.2.3.** Under a given scheduling algorithm (centralized or distributed), for any 
$0 < \epsilon_1, \delta_1 < 1$, if there exists $B > 0$ such that $\forall t$, w.p. greater than $(1 - \delta_1)$, whenever 
$\|Q(t)\| > B,$

$$\sum_{j=1}^{N} \sum_{i=1}^{M} f(q_i(t)) \mu_{ij}(t) \geq N(1 - \epsilon_1) \max_i f(q_i(t)),$$

then the scheduling algorithm is throughput-optimal.

Lemma 6.2.2 directly follows from Proposition 2 in [83] under the time-scale separation assumption (i.e., the underlying Markov chain of the scheduling algorithm converges instantaneously to its steady state distribution compared to link weight adaptation rate of the algorithm) and Lemma 6.2.3 follows from Theorem 1 in [22].

Consider any given $0 < \epsilon_1, \delta_1 < 1$. Since the contention variables $(a_{ij}(t))_i$ and transmission variables $(p_{ij}(t))_i$ are i.i.d. over channels $j$, the schedule of a channel under the multi-channel random access algorithm is independent of that of any another channel given queue backlogs $Q(t - 1)$. Hence, we have from Lemma 6.2.2, w.p. greater than $(1 - \frac{\delta_1}{N})^N \geq 1 - \delta_1$, whenever $\|(Q(t - 1))\| > B_1,$

$$\sum_{j=1}^{N} \sum_{i=1}^{M} f(q_i(t - 1)) \mu_{ij}(t) \geq N(1 - \frac{\epsilon_1}{4}) \max_i f(q_i(t - 1)). \tag{6.2.3}$$
By property (A2), there exists some $B_2 > 0$ such that for any $x > B_2$,

$$f(x - \max\{C, A_{\text{max}}\}) \geq (1 - \frac{\epsilon_1}{4})f(x). \tag{6.2.4}$$

From (6.2.4) and the fact that

$$[q_i(t - 1) - C]^+ \leq q_i(t) \leq q_i(t - 1) + A_{\text{max}},$$

we obtain

$$\sum_{i=1}^{M} \sum_{j=1}^{N} f(q_i(t))\mu_{ij}(t)$$

$$\geq (1 - \frac{\epsilon_1}{4}) \sum_{i, q_i(t-1) > B_2} \sum_{j=1}^{N} f(q_i(t-1))\mu_{ij}(t), \tag{6.2.5}$$

Combining (6.2.3)(6.2.5), w.p. greater than $1 - \delta_1$, whenever $\|Q(t)\| > \max\{B_1 + A_{\text{max}}, B_2\}$, we have

$$\sum_{i=1}^{M} \sum_{j=1}^{N} f(q_i(t))\mu_{ij}(t)$$

$$\geq N(1 - \frac{\epsilon_1}{4})^2 \max_i f(q_i(t-1)) - (1 - \frac{\epsilon_1}{4})Nf(B_2) \tag{6.2.6}$$

$$\geq N(1 - \frac{\epsilon_1}{4})^2 \max_i f(q_i(t) - A_{\text{max}}) - (1 - \frac{\epsilon_1}{4})Nf(B_2)$$

$$\geq N(1 - \frac{\epsilon_1}{4})^3 \max_i f(q_i(t)) - (1 - \frac{\epsilon_1}{4})Nf(B_2),$$

where the last inequality follows from (6.2.4).
According to property (A1), we can find $B_3 > 0$ such that whenever $||Q(t)|| > B_3$,

$$N\frac{\epsilon_1}{4}(1 - \frac{\epsilon_1}{4})^3 \max_i f(q_i(t)) \geq (1 - \frac{\epsilon_1}{4})Nf(B_2).$$

Combining with (6.2.6), we conclude, w.p. greater than $1 - \delta_1$, whenever $||Q(t)|| > \max\{B_1 + A_{\max}, B_2, B_3\}$,

$$\sum_{i=1}^{M} \sum_{j=1}^{N} f(q_i(t))\mu_{ij}(t) \geq N(1 - \frac{\epsilon_1}{4})^4 \max_i f(q_i(t)) \geq N(1 - \epsilon_1) \max_i f(q_i(t)).$$

Hence, the condition in Lemma 6.2.3 holds with probability greater than $(1 - \delta_1)$ for $B = \max\{B_1 + A_{\max}, B_2, B_3\}$, which completes the proof of Theorem 6.2.1 by Lemma 6.2.3.

### 6.3 Asymptotic Queuing Behavior under the Many-Channel Regime

Before we present the asymptotic queuing behavior under the multi-channel random access algorithm in Theorem 6.3.1, we introduce the following limit law assumption on arrival processes:

$$A_i(t) \overset{P}{\to} N \alpha, \forall i, \forall t,$$

where $\alpha > 0$ can be interpreted as the arrival rate normalized by the number of channels. For analytical simplicity, we consider a uniform $\alpha$ for all links in Section 6.3 and Section 6.5, and extend our analysis to heterogeneous arrival rates in Section 6.6. The arrival process defined above is general: It can be non-stochastic, dependent over links or even bursty (with the constraint (6.3.1) satisfied). The arrival process does not necessarily depend on the number of channels, e.g., $A_i(t) = \alpha, \forall i, \forall t$ (i.e., constant homogenous arrivals). Some examples of arrival processes are given with numerical results in Section 6.8.2. Note
that the analysis can be readily extended to the model with assumption (6.3.1) relaxed as
$$A_i(t) \xrightarrow{P} \alpha(t), \forall i, \forall t,$$
where $\alpha(t)$ is deterministic for each time slot $t$.

For analytical simplicity, we initialize the system at $t = -1$ as follows:

$$q_i(-1) = 0, \mu_{ij}(-1) = 0, \text{ and } A_i(-1) \xrightarrow{P} \alpha, \forall i, j.$$  \hspace{1cm} (6.3.2)

Then, we show in the following theorem that the queuing behavior of individual links under
the multi-channel random access algorithm converges to an equivalent deterministic single-
queue system as the number of channels grows.

**Theorem 6.3.1.** Assume that the transmission weight function $h$ is invertible and assumption (6.3.1) holds. There exist an equivalent deterministic queue $q(t)$ and an equivalent schedule variable $v(t)$ for each time slot $t$, such that the following four arguments (denoted by $I(t)$, $II(t)$, $III(t)$, and $IV(t)$) hold $\forall t$ under the multi-channel random access algorithm:

1. **The data queue backlogs converge in probability to the equivalent deterministic queue $q(t)$ for each link:**
   $$I(t): q_i(t) \xrightarrow{P} q(t), \forall i.$$  \hspace{1cm} (6.3.3)

2. **The schedules of individual links converge in law to $v(t)$:**
   $$II(t): \mu_{ij}(t) \xrightarrow{L} v(t), \forall i, j.$$  \hspace{1cm} (6.3.4)

3. **The following law-of-large-numbers result holds for each link:**
   $$III(t): \frac{1}{N} \sum_{j=1}^{N} \mu_{ij}(t) \xrightarrow{P} \mathbb{E}\{v(t)\}, \forall i.$$  \hspace{1cm} (6.3.5)

4. **Schedules of different channels become asymptotically mutually independent.** Specifically, for any given two links $i_1, i_2 \in \mathcal{L}$, and any two distinct channels $j_1 \neq j_2$, the scheduling
decisions are independent, i.e., \( \forall k_1, k_2 \in \{0, 1\} \),

\[
IV(t) \triangleq \lim_{N \to \infty} P\{\mu_{i_1j_1}(t) = k_1, \mu_{i_2j_2}(t) = k_2\} = P\{v(t) = k_1\}P\{v(t) = k_2\}.
\] (6.3.6)

The dynamics of the equivalent queue and the equivalent schedule variable are defined as

\[
q(t) = [q(t-1) + \alpha - CE\{v(t-1)\}]^+, t \geq 0,
\] (6.3.7)

\[
v(t) = V_1(t)v(t-1) + V_2(t)(1 - v(t-1)), t \geq 0,
\] (6.3.8)

with the initial states

\[
q(-1) = 0 \text{ and } v(-1) = 0.
\] (6.3.9)

\( V_1(t) \) and \( V_2(t) \) are mutually independent random variables, independent over time slot \( t \), defined in the following:

\[
V_1(t) = \begin{cases} 
1, & \text{w.p. } F_1 + (2 - M)F_0(q(t-1)), \\
0, & \text{otherwise},
\end{cases}
\]

\[
V_2(t) = \begin{cases} 
1, & \text{w.p. } F_0(q(t-1)), \\
0, & \text{w.p. } 1 - F_0(q(t-1)),
\end{cases}
\]

where for notational simplicity we define

\[
F_1 \triangleq 1 - \beta(1 - \beta)^M - 1
\]

\[
F_0(x) \triangleq \beta(1 - \beta)^{M-1} \frac{h(x)}{1 + h(x)}
\]

Proof. The proof of Theorem 6.3.1 is provided in Section 6.4. \( \square \)
In the following, we verify that $V_1(t)$ is a valid random variable, i.e., $0 < F_1 + (2 - M)F_0(q(t - 1)) < 1$. Note that different from Theorem 6.2.1, the assumptions of stationary arrival processes and time-scale separation are not required in Theorem 6.3.1.

Proof (Validity of $V_1(t)$): For notational simplicity, we denote

$$U \triangleq F_1 + (2 - M)F_0(q(t - 1)) = 1 + \beta(1 - \beta)^{M-1} \left[ \frac{h(q(t - 1))}{1 + h(q(t - 1))} (2 - M) - 1 \right].$$

In the following, we prove $0 < U < 1$ to show that $V_1(t)$ is a valid random variable.

First, it is easy to verify that $U < 1$, $\forall M \geq 1$ and $U > 0$, when $M = 1$.

Now consider the case of $M \geq 2$:

$$U \geq 1 + \beta(1 - \beta)^{M-1}(1 - M) \geq 1 - \left( 1 - \frac{1}{M} \right)^M > 0,$$

where the second inequality in (6.3.10) holds with equality when $\beta = \frac{1}{M}$.

Hence, we have proved that $0 < U < 1$, i.e., $V_1(t)$ is a valid random variable.

**Remark 6.3.2.** Theorem 6.3.1 states that the queuing behavior of each individual link converges to an equivalent deterministic single-queue system as the number of channels goes to infinity. Specifically, according to (6.3.3) and (6.3.5), the queuing behavior of each link $i$ ($q_i(t), \frac{C}{N} \sum_{j=1}^{N} \mu_{ij}(t)$) (i.e., queue backlog and scheduled service rate) converges asymptotically in the number of channels to the equivalent queue system $(q(t), CE\{v(t)\})$.

By taking expectation on both sides of (6.3.8), we know that the equivalent queue system $(q(t), CE\{v(t)\})$ can be updated deterministically and independent of individual links with (6.3.7) and

$$\mathbb{E}\{v(t)\} = [F_1 + (1 - M)F_0(q(t - 1))]\mathbb{E}\{v(t - 1)\} + F_0(q(t - 1)).$$

188
Thus, the easy updates of \((q(t), CE\{v(t)\})\) yield a simplified and scalable characterization of the queuing behavior for individual links under the multi-channel random access algorithm.

In Section 6.5, we will study the steady state of the equivalent queue system \((q(t), CE\{v(t)\})\), which becomes the (asymptotic) steady state of each link under the multi-channel random access algorithm according to Theorem 6.3.1. In the following corollary, we present the LLN results for the aggregated queue backlog and aggregated service rate under the multi-channel random access algorithm.

**Corollary 6.3.3.** Given \(q(t)\) and \(v(t)\) updated as in Theorem 6.3.1, the following LLN results hold for any time slot \(t\):

\[
\frac{1}{M} \sum_{i=1}^{M} q_i(t) \overset{P}{\rightarrow} N \gets q(t),
\]

\[
\frac{C}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \mu_{ij}(t) \overset{P}{\rightarrow} N \gets CE\{v(t)\}.
\]

Corollary 6.3.3 directly follows (6.3.3) and (6.3.5). Note that \(\frac{1}{M} \sum q_i(t)\) and \(\frac{C}{MN} \sum \sum \mu_{ij}(t)\) can be considered as the average queue occupancy (which represents the congestion level/delay performance) and average service rate under the multi-channel random access algorithm, respectively.

### 6.4 Proof of Theorem 6.3.1

Theorem 6.3.1 can be proved by mathematical induction over time slot \(t\). With the initialization (6.3.2)(6.3.9), I(−1), II(−1), III(−1), and, IV(−1) trivially hold as the base case. Suppose I\((t−1)\), II\((t−1)\), III\((t−1)\), and, IV\((t−1)\) hold, which we call as induction hypothesis. We prove that the induction step holds, i.e., I\((t)\), II\((t)\), III\((t)\), and, IV\((t)\) hold in the following subsections, which completes the proof for Theorem 6.3.1.
6.4.1 Proof of I(t)

From the induction hypothesis I(t - 1) and III(t - 1), we have

\[ q_i(t - 1) + A_i(t - 1) - \frac{C}{N} \sum_{j=1}^{N} \mu_{ij}(t - 1) \]

\[ \xrightarrow{P} q(t - 1) + \alpha - CE \{ v(t - 1) \}. \]

Thus, we conclude that I(t) holds, i.e., \( q_i(t) \xrightarrow{P} q(t), \forall i, \) by queue dynamics (6.2.1)(6.3.7) and the continuity of the function \[ \cdot \] .

6.4.2 Proof of II(t)

Taking the conditional expectation on both sides of (6.2.2) leads to

\[ \mathbb{E}\{\mu_{ij}(t)|(\mu_{kj}(t - 1))_{k=1}^{M}, q_i(t - 1)\} \]

\[ = F_0(q_i(t - 1)) - \sum_{k \neq i} F_0(q_i(t - 1)) \mu_{kj}(t - 1) \]

\[ + F_1 \mu_{ij}(t - 1). \tag{6.4.1} \]

Further taking the expectation of both sides of (6.4.1), we get

\[ \mathbb{E}\{\mu_{ij}(t)\} = \mathbb{E}\{F_0(q_i(t - 1))\} \]

\[ - \sum_{k \neq i} \mathbb{E}\{F_0(q_i(t - 1)) \mu_{kj}(t - 1)\} + F_1 \mathbb{E}\{\mu_{ij}(t - 1)\}. \tag{6.4.2} \]

From the induction hypothesis I(t - 1) and II(t - 1), we have

\[ (q_i(t - 1), \mu_{kj}(t - 1)) \xrightarrow{L} (q(t - 1), v(t - 1)). \]

By applying the continuous mapping theorem [56], we find that

\[ F_0(q_i(t - 1)) \mu_{kj}(t - 1) \xrightarrow{L} F_0(q(t - 1)) v(t - 1). \]
By the bounded convergence theorem [56], we further obtain
\[
\lim_{N \to \infty} \mathbb{E}\{F_0(q_i(t-1))\mu_{kj}(t-1)\} = F_0(q(t-1))\mathbb{E}\{v(t-1)\}.
\]

Similarly, we can obtain
\[
\lim_{N \to \infty} \mathbb{E}\{F_0(q_i(t-1))\} = F_0(q(t-1)),
\]
\[
\lim_{N \to \infty} F_1\mathbb{E}\{\mu_{ij}(t-1)\} = F_1\mathbb{E}\{v(t-1)\}.
\]

Applying the above results to (6.4.2), we conclude
\[
\lim_{N \to \infty} P\{\mu_{ij}(t) = 1\} = \lim_{N \to \infty} \mathbb{E}\{\mu_{ij}(t)\}
= F_0(q(t-1)) - (M-1)F_0(q(t-1))\mathbb{E}\{v(t-1)\}
+ F_1\mathbb{E}\{v(t-1)\}
= F_0(q(t-1)) + [(1-M)F_0(q(t-1)) + F_1]\mathbb{E}\{v(t-1)\}
= \mathbb{E}\{v(t)\} = P\{v(t) = 1\},
\]
where the second to last equality follows from (6.3.11). Hence, \(\Pi(t)\) holds, i.e., \(\mu_{ij}(t) \overset{L}{\to}_N v(t), \forall i, j\), by definition of convergence in law.

### 6.4.3 Proof of IV\((t)\)

We prove IV\((t)\) by enumerating all \(k_1, k_2\) in (6.3.6). Consider the case \(k_1 = k_2 = 1\). Since \(\mu_{i_1j_1}(t)\) and \(\mu_{i_2j_2}(t)\) are binary random variables, we observe that
\[
P\{\mu_{i_1j_1}(t) = 1, \mu_{i_2j_2}(t) = 1\} = \mathbb{E}\{\mu_{i_1j_1}(t)\mu_{i_1j_1}(t)\},
\]
\[
(P\{v(t) = 1\})^2 = \mathbb{E}\{v(t)\}^2.
\]
Thus, proving (6.3.6) for the case $k_1 = k_2 = 1$ is equivalent to proving

$$
\lim_{N \to \infty} \mathbb{E}\{g_1(t)g_2(t)\} = \mathbb{E}\{v(t)\}^2.
$$

For cases when $k_1 = 0$ and/or $k_2 = 0$, result (6.4.4) holds with $\mu_{i,j_1}(t)$ and/or $\mu_{i,j_2}(t)$ replaced by $(1 - \mu_{i,j_1}(t))$ and/or $(1 - \mu_{i,j_2}(t))$.

To analyze the above enumerations collectively, we define two functions $g^{(0)}, g^{(1)} : \{0,1\} \to \{0,1\}$ as $g^{(1)}(x) = x$ and $g^{(0)}(x) = 1 - x$. Then, proving (6.3.6) is equivalent to proving, $\forall i_1, i_2, \forall j_1 \neq j_2, \forall g_1, g_2 \in \{g^{(0)}, g^{(1)}\}$,

$$
\lim_{N \to \infty} \mathbb{E}\{g_1(t)g_2(t)\} = \mathbb{E}\{g_1(v(t))\} \mathbb{E}\{g_2(v(t))\}.
$$

According to the schedule dynamics (6.2.2), for any given link $i$ and channel $j$, $\mu_{i,j}(t)$ is determined by $(a_{kj}(t))_{k=1}^M$, $p_{ij}(t)$, and $(\mu_{kj}(t - 1))_{k=1}^M$. Note that the contention variables $a_{ij}(t)$ are independent over $i,j$ by definition and the transmission variables $p_{ij}(t)$ are independent over $i,j$ given $q_i(t - 1)$ by definition. Hence, for any given links $i_1$ and $i_2$ with channels $j_1 \neq j_2$, $\mu_{i_1,j_1}(t)$ and $\mu_{i_2,j_2}(t)$ are independent given the schedule and queue backlog.
information at the previous time slot $\mathcal{H}(t-1)$, i.e.,

$$\mathcal{H}(t-1) \triangleq \left( (\mu_{ij}(t-1))_{j=1}^{M} q_{i_1}(t-1), q_{i_2}(t-1) \right).$$

As a result, for any two functions $g_1, g_2 \in \{g^{(0)}, g^{(1)}\}$, we get the following conditional expectation:

$$E\{g_1(\mu_{i_1 j_1}(t)) g_2(\mu_{i_2 j_2}(t)) | \mathcal{H}(t-1) \}$$

$$= E\{g_1(\mu_{i_1 j_1}(t)) | \mathcal{H}(t-1) \} E\{g_2(\mu_{i_2 j_2}(t)) | \mathcal{H}(t-1) \}$$

$$= [F_{g_1,0}(q_{i_1}(t-1)) - \sum_{k_1 \neq i_1} F_{g_1,2}(q_{i_1}(t-1)) \mu_{k_1 j_1}(t-1) + F_{g_1,1} \mu_{i_1 j_1}(t-1)]$$

$$\times [F_{g_2,0}(q_{i_2}(t-1)) - \sum_{k_2 \neq i_2} F_{g_2,2}(q_{i_2}(t-1)) \mu_{k_2 j_2}(t-1) + F_{g_2,1} \mu_{i_2 j_2}(t-1)],$$

\forall i_1, i_2, \forall j_1 \neq j_2,$n where we denote

$$F_{g^{(0)},0}(x) \triangleq 1 - F_0(x), \quad F_{g^{(0)},1} \triangleq -F_1, \quad F_{g^{(0)},2}(x) \triangleq -F_0(x),$$

$$F_{g^{(1)},0}(x) = F_{g^{(1)},2}(x) \triangleq F_0(x), \quad F_{g^{(1)},1} \triangleq F_1.$$

Note that the functions $F_{g^{(0)},0}, F_{g^{(1)},0} : \mathbb{R}^+ \rightarrow \mathbb{R}$.

By the induction hypothesis of $\mathcal{I}(t-1)$, we have

$$(q_{i_1}(t-1), q_{i_2}(t-1)) \xrightarrow{P} \mathcal{N} (q(t-1), q(t-1)),$$

which further implies from the continuous mapping theorem and the bounded convergence
theorem

\[ \lim_{N \to \infty} \mathbb{E}\{F_{g_1,0}(q_{i_1}(t-1))F_{g_2,0}(q_{i_2}(t-1))\} = F_{g_1,0}(q(t-1))F_{g_2,0}(q(t-1)). \]

Note that by the induction hypothesis IV\((t-1)\), \(\mu_{i_1j_1}(t-1)\) and \(\mu_{i_2j_2}(t-1)\) are asymptotically mutually independent given \(j_1 \neq j_2\). Defining \(v_1(t-1)\) and \(v_2(t-1)\) as i.i.d. random variables distributed according to \(v(t-1)\) (i.i.d. with respect to \(v(t-1)\)), we have

\[ (\mu_{i_1j_1}(t-1), \mu_{i_2j_2}(t-1)) \overset{L}{\to} N(v_1(t-1), v_2(t-1)). \]

\( (\mu_{i_1j_1}(t-1), \mu_{i_2j_2}(t-1), q_{i_2}(t-1)) \overset{L}{\to} N(v_1(t-1), v_2(t-1), q(t-1)), \forall k. \)

\[ (F_{g_1,0}(q_{i_1}(t-1))F_{g_2,2}(q_{i_2}(t-1)), \mu_{k_2j_2}(t-1)) \overset{L}{\to} N(F_{g_1,0}(q(t-1))F_{g_2,2}(q(t-1)), v_1(t-1)), \forall k. \]

\[ (F_{g_1,2}(q_{i_1}(t-1))F_{g_2,2}(q_{i_2}(t-1)), \mu_{k_1j_1}(t-1), \mu_{k_2j_2}(t-1)) \overset{L}{\to} N(F_{g_1,2}(q(t-1))F_{g_2,2}(q(t-1)), v_1(t-1), v_2(t-1)), \forall k_1, k_2. \]

Hence, by taking the expectation of both sides of (6.4.5), and employing the bounded convergence theorem to the above convergence results, we obtain (6.4.3), where the last equality in (6.4.3) follows from (6.3.11). Since \(g_1, g_2\) are arbitrary functions chosen from \(\{g^{(0)}, g^{(1)}\}\), (6.3.6) holds for any \(k_1, k_2 \in \{0, 1\}\), which completes the proof of IV\((t)\).
6.4.4 Proof of III(t)

We first take the variance of \( \frac{1}{N} \sum_{j=1}^{N} \mu_{ij}(t) \), for any given link \( i \):

\[
\text{Var} \left\{ \frac{1}{N} \sum_{j=1}^{N} \mu_{ij}(t) \right\} = N^{-2} \sum_{j=1}^{N} \text{Var}\{\mu_{ij}(t)\} + N^{-2} \sum_{j,k=1,\ldots,N, j \neq k} \text{Cov}\{\mu_{ij}(t), \mu_{ik}(t)\}
\]

\[
= N^{-1} \text{Var}\{\mu_{i1}(t)\} + \frac{N-1}{N} \text{Cov}\{\mu_{i1}(t), \mu_{i2}(t)\}
\]

(6.4.6)

\[
\xrightarrow{N \to \infty} 0,
\]

(6.4.7)

where (6.4.6) follows the exchangeability of \( \mu_{i1}(t), \ldots, \mu_{iN}(t) \), and (6.4.7) is implied from

\[
\lim_{N \to \infty} \text{Cov}(\mu_{i1}(t), \mu_{i2}(t)) = 0
\]

since we have proved IV(t) in Section 6.4.3, i.e., \( \mu_{i1}(t) \) and \( \mu_{i2}(t) \) are asymptotically mutually independent.

Employing Chebyshev’s inequality to \( \text{Var}\{\frac{1}{N} \sum_{j=1}^{N} \mu_{ij}(t)\} \), we have \( \forall \epsilon > 0 \):

\[
P \left\{ \left| \frac{1}{N} \sum_{j=1}^{N} \mu_{ij}(t) - \text{E}\{\frac{1}{N} \sum_{j=1}^{N} \mu_{ij}(t)\} \right| \geq \epsilon \right\} \xrightarrow{N \to \infty} 0.
\]

Thus, by the exchangeability and the definition of convergence in probability, we conclude

\[
\frac{1}{N} \sum_{j=1}^{N} \mu_{ij}(t) - \text{E}\{\mu_{i1}(t)\}
\]

\[
= \frac{1}{N} \sum_{j=1}^{N} \mu_{ij}(t) - \text{E}\{\frac{1}{N} \sum_{j=1}^{N} \mu_{ij}(t)\} \xrightarrow{P} 0.
\]
Employing II\((t)\) (proved in Section 6.4.2) to the above convergence result yields

\[
\frac{1}{N} \sum_{j=1}^{N} \mu_{ij}(t) \overset{P}{\to}_N \mathbb{E}\{v(t)\},
\]

which completes the proof of III\((t)\).

### 6.5 A Steady State Study

#### 6.5.1 Steady State under the Many-Channel Regime

We have shown that the equivalent queue system \((q(t), C\mathbb{E}\{v(t)\})\) represents the queuing behavior of individual links in the many-channel regime according to Theorem 6.3.1 and Corollary 6.3.3. We now study the steady state result for \((q(t), C\mathbb{E}\{v(t)\})\), i.e., the case when \(t \to \infty\), in an attempt to approximate the steady state of the queue backlogs under the multi-channel random access scheduling algorithm. By analyzing the steady state of \(q(t)\), we show that the multi-channel random access algorithm is asymptotically (with respect to \(N\)) throughput-optimal \textit{without the time-scale separation assumption}, which is required for Theorem 6.2.1 to hold.

Since the maximal stabilizable normalized arrival rate cannot exceed \(\frac{C}{M}\) as \(N \to \infty\) \(^5\), we only analyze the case for \(\alpha < \frac{C}{M}\). We assume that the following limit exists: \(^6\)

\[
(q, v) \overset{\Delta}{=} \lim_{t \to \infty} (q(t), C\mathbb{E}\{v(t)\}), \text{ with } 0 < q \leq \infty. \tag{6.5.1}
\]

\(^5\)Recall assumption (6.3.1) and that the total channel capacity is \(C\). We must have \(\alpha \leq \frac{C}{M}\) as \(N \to \infty\).

\(^6\)We will show in Section 6.5.2 that \((q, v)\) is indeed a stable equilibrium of the system \((q(t), C\mathbb{E}\{v(t)\})\). We also show through extensive numerical results in Section 6.8 that the queuing behavior of individual links \(q_i(t), \frac{C}{N} \sum_{j=1}^{N} \mu_{ij}(t)\) converges to \((q, v)\) in time, for sufficiently large \(N\).
Note that \( q = \infty \) stands for the case when the normalized arrival rate \( \alpha \) cannot be stabilized by the algorithm. We say that the multi-channel random access algorithm is \textit{asymptotically throughput-optimal} if the steady state \( q \) of the equivalent queue \( q(t) \) corresponding to the algorithm is finite (i.e., \( q < \infty \)) for all \( 0 < \alpha < \frac{C}{M} \).

By taking the limit (over time) of both sides of (6.3.7) and (6.3.11), we obtain the steady state in its closed form

\[
(q, v) = \left( h^{-1} \left( \frac{\alpha}{C - \alpha M} \right), \alpha \right).
\]

Since \( \alpha < \frac{C}{M} \), we can find \( \epsilon \equiv C - \alpha M > 0 \) such that

\[
0 < q = h^{-1} \left( \frac{\alpha}{\epsilon} \right) < \infty.
\]

The steady-state equivalent queue backlogs that correspond to different weight functions are listed in Table 6.1.

\textbf{Remark 6.5.1.} \( \epsilon \) can be considered as the closeness of the traffic load to the optimality. We know from (6.3.3) that the data queue backlogs converge in probability to \( q(t) \). Therefore, when the number of channels \( N \) becomes large, the individual queue backlogs in the steady state converge to the steady state of the equivalent queue \( q \). We can see from Table 6.1 that more aggressive weight functions lead to smaller delays. For example, \( h(x) = e^x - 1 \) results in a better delay performance than \( x \) and \( \log(x + 1) \). This is in accordance with the findings in a single channel setting [11], where a lower-bound on the expected queue backlog is derived with respect to weight functions. We note that while a more aggressive weight function reduces delays, it can potentially aggravate the temporal starvation [61] when there is a limited number of channels, i.e., links can undergo prolonged periods of inactivity followed by a prolonged period of activity, leading to bursty service and undesirable jitter performance. The study of this tradeoff between delay and temporal starvation will be one of our future works.
Since channel resources are shared by \( M \) links, the interval of \( 0 < \alpha < \frac{C}{M} \) denotes the stabilizable range of (normalized) arrival rates. From (6.5.2), the steady state value \( q \) is finite for any \( \alpha < \frac{C}{M} \), and hence, the random access algorithm is asymptotically (with respect to \( N \)) throughput-optimal under the assumption that \( \lim_{t \to \infty} (q(t), C\mathbb{E}\{v(t)\}) \) exists. This asymptotic result is summarized in the following proposition:

**Proposition 6.5.2.** The multi-channel random access algorithm is asymptotically throughput-optimal with respect to \( N \) under assumption (6.5.1).

We note that Proposition 6.5.2 is a steady state outcome of the convergence results I(\( t \))-IV(\( t \)) in Theorem 6.3.1, which hold for each time slot \( t \). Thus, like Theorem 6.3.1, Proposition 6.5.2 does not require the assumptions of stationary arrival processes and the time-scale separation.

In the next subsection, we show that \( \left(h^{-1}\left(\frac{\alpha}{\epsilon}\right), \alpha\right) \) is indeed a stable equilibrium of the system \((q(t), C\mathbb{E}\{v(t)\})\).

**Table 6.1:** Steady state equivalent queue backlog \( q \) with different weight functions \( h \)

<table>
<thead>
<tr>
<th>( h(x) )</th>
<th>( e^x - 1 )</th>
<th>( x )</th>
<th>( \log(x + 1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q )</td>
<td>( \log\left(\frac{\alpha}{\epsilon} + 1\right) )</td>
<td>( \frac{\alpha}{\epsilon} )</td>
<td>( e^\frac{\alpha}{\epsilon} - 1 )</td>
</tr>
</tbody>
</table>

### 6.5.2 The Stability of the Equilibrium of the Equivalent Queue System \((q(t), C\mathbb{E}\{v(t)\})\)

For notational simplicity, we define \( W \triangleq \beta(1-\beta)^{M-1} \) and denote the state of the equivalent queue system by \( \mathbf{x}(t) \triangleq \begin{bmatrix} q(t) \\ C\mathbb{E}\{v(t)\} \end{bmatrix} \) with an initial state \( \mathbf{x}_0 \triangleq \mathbf{x}(-1) \). By the dynamics (6.3.7)(6.3.11), the discrete-time equivalent queue system (as shown in (6.5.3)) has an
\[ x(t) = \begin{bmatrix} q(t-1) + \alpha - CE\{v(t-1)\} \\ 1 - W + (1 - M)W \frac{h(q(t-1))}{1 + h(q(t-1))} \end{bmatrix} CE\{v(t-1)\} + WC \frac{h(q(t-1))}{1 + h(q(t-1))} . \tag{6.5.3} \]

\[ \dot{x}(t) = \begin{bmatrix} \alpha - CE\{v(t)\} \\ -W + (1 - M)W \frac{h(q(t))}{1 + h(q(t))} \end{bmatrix} CE\{v(t)\} + WC \frac{h(q(t))}{1 + h(q(t))} , \quad x(t) > 0. \tag{6.5.4} \]

equilibrium at \((h^{-1}(\frac{\alpha}{\epsilon}), \alpha)\): Assuming \(q(t) = h^{-1}(\frac{\alpha}{\epsilon})\) and \(CE\{v(t)\} = \alpha\), we have \((q(t + 1), CE\{v(t + 1)\}) = (q(t), CE\{v(t)\})\). We next show that this equilibrium is stable by using a continuous-time approximation.

For analytical simplicity, we study the stability of the equilibrium under the corresponding continuous-time model for the discrete-time nonlinear system \((q(t), CE\{v(t)\})\) shown in (6.5.3). The continuous-time equivalent of the nonlinear system (6.5.3) can be characterized by (6.5.4). We know that \((h^{-1}(\frac{\alpha}{\epsilon}), \alpha)\) is the only equilibrium of the system (6.5.4) by checking the solution to \(\dot{x}(t) = 0\). Note that the condition \(x(t) > 0\) holds for any \(x(t)\) within some neighborhood of the equilibrium since both \(h^{-1}(\frac{\alpha}{\epsilon})\) and \(\alpha\) are positive.

A system \(x(t)\) is said to be asymptotically stable about its equilibrium \(x^*\) [58] if (a) The system is stable in the sense of Lyapunov, i.e., for any \(\epsilon_2 > 0\), there exists a constant \(\delta_2(\epsilon_2) > 0\), such that for any \(x_0\) with \(||x_0 - x^*|| < \delta_2\), we have \(||x(t) - x^*|| < \epsilon_2, \forall t\); (b) There exists a constant \(\delta_3\) such that, for any \(x_0\) with \(||x_0 - x^*|| < \delta_3\), we have \(\lim_{t \to \infty} ||x(t) - x^*|| = 0\).

In other words, if a system is asymptotically stable, any state in some neighborhood of the equilibrium will be contracted to the equilibrium asymptotically in time.

Under the continuous-time model, the stability of the equilibrium \(h^{-1}(\frac{\alpha}{\epsilon}), \alpha\) is stated in the following proposition.

**Proposition 6.5.3.** System (6.5.4) is asymptotically stable with respect to its equilibrium \((h^{-1}(\frac{\alpha}{\epsilon}), \alpha)\).
Proof. The Jacobian of system (6.5.4) evaluated at the equilibrium is

\[
A = \begin{bmatrix}
0 & -1 \\
\frac{WC}{\epsilon + \alpha} & -\frac{W h'(h^{-1}(\frac{\alpha}{\epsilon}))}{\epsilon + \alpha} \epsilon^2
\end{bmatrix}.
\]

It is easy to check that the two eigenvalues of \( A \) have negative real parts. Hence, according to the First Method of Lyapunov [58], the system is asymptotically stable with respect to \( (h^{-1}(\frac{\alpha}{\epsilon}), \alpha) \), i.e., statements (a) and (b) hold.

According to Proposition 6.5.3, the equilibrium \( (h^{-1}(\frac{\alpha}{\epsilon}), \alpha) \) is indeed a local sink of the system.

### 6.6 Asymptotic Results under Heterogeneous Arrival Rates

In Section 6.3 and Section 6.5, we have assumed a uniform arrival rate (6.3.1) for all links. In this section, we consider the case of heterogeneous arrival rates:

\[
A_i(t) \xrightarrow{P} N \alpha_i, \forall i, \forall t,
\]

where \( \alpha_i > 0 \) represents the arrival rate of link \( i \) normalized by the number of channels. We will show in Theorem 6.6.1 that the queuing behavior of the system converges to an equivalent \( M \)-queue system under the multi-channel random access algorithm.

**Theorem 6.6.1.** Under assumption (6.6.1), there exist \( M \) equivalent deterministic queues \((\bar{q}_i(t))_{i=1}^M\) and \( M \) corresponding equivalent schedule variables \((v_i(t))_{i=1}^M\) for each time slot \( t \) such that, \( \forall i, \)

\[
q_i(t) \xrightarrow{P} N \bar{q}_i(t), \text{ and } \frac{1}{N} \sum_{j=1}^N \mu_{ij}(t) \xrightarrow{P} N \mathbb{E}\{v_i(t)\},
\]

\(200\)
with the deterministic dynamics of \((\bar{q}_i(t))\) and \((\mathbb{E}\{v_i(t)\})\) defined as follows:

\[\bar{q}_i(t) = [\bar{q}_i(t - 1) + \alpha_i - C\mathbb{E}\{v_i(t - 1)\}]^+, t \geq 0, \tag{6.6.2}\]

\[\mathbb{E}\{v_i(t)\} = F_1\mathbb{E}\{v_i(t - 1)\} + (1 - \sum_{i' \neq i} \mathbb{E}\{v_i'(t - 1)\})F_0(\bar{q}_i(t - 1)). \tag{6.6.3}\]

Theorem 6.6.1 states that the queuing behavior of individual links \(\left(\bar{q}_i(t), C\mathbb{E}\{v_i(t)\}\right)_{i=1}^M\) converges (in the number of channels) to an equivalent deterministic \(M\)-queue system \((\bar{q}_i(t), C\mathbb{E}\{v_i(t)\})_{i=1}^M\) with dynamics (6.6.2)(6.6.3). The proof of Theorem 6.6.1 follows the same steps in proving Theorem 6.3.1 (in Section 6.4) and is omitted for brevity.

Similar to the steady state study in Section 6.5, from dynamics (6.6.2)(6.6.3), we can find a steady state \((\bar{q}_i, v_i)_{i=1}^M\) of the equivalent queue system \((\bar{q}_i(t), C\mathbb{E}\{v_i(t)\})_{i=1}^M\) for any \((\alpha_i)\) satisfying \(\sum_{i=1}^M \alpha_i < C\):

\[\bar{q}_i, v_i) = \left(h^{-1}\left(\frac{\alpha_i}{C - \sum_{i=1}^M \alpha_i}\right), \alpha_i\right), \forall i. \tag{6.6.4}\]

Since the total channel capacity \(C\) is shared by \(M\) links, \(0 < \sum_{i=1}^M \alpha_i < C\) denotes the stabilizable range of (normalized) arrival rates. The result (6.6.4) thus shows that the queue backlogs at the steady state are finite for any stabilizable \((\alpha_i)\), and hence the multi-channel random access algorithm is asymptotically throughput-optimal with respect to \(N\), which is summarized in the following proposition:

**Proposition 6.6.2.** The multi-channel random access algorithm is asymptotically throughput-optimal with respect to \(N\) under assumption (6.6.1).

Note that similar to Theorem 6.3.1 and Proposition 6.5.2, we do not require stationary
arrival processes and the time-scale separation assumption for Theorem 6.6.1 and Proposition 6.6.2 to hold.

6.7 Asymptotic Results under Fading Channels

So far, we have assumed non-fading channels, i.e., channels are always available with a capacity constant over time. In this section, we consider the case of fading channels. We will show that the queuing behavior of individual links converges to an equivalent deterministic single-queue system and the multi-channel random access algorithm, with some modifications, is asymptotically throughput optimal in the many-channel regime.

Instead of a constant channel capacity of \( \frac{C}{N} \) in the previous discussion, we now denote by \( C_j(t) \) the maximum available service rate of channel \( j \) at time slot \( t \), where we can consider \( 0 < C_j(t) \leq \frac{C}{N} \) to be channel \( j \)'s state at time slot \( t \). We assume that the channel processes \( C_j(t), j = 1, ..., N \), are i.i.d. over \( j \). Note that \( C_j(t) > 0 \) can take an arbitrarily small value if a channel is not available to the links at time slot \( t \). For analytical simplicity, we assume that, given any channel \( j \), \( C_j(t) \) is i.i.d. over time taking values in a finite state space with \( 0 < \mathbb{E}\{NC_j(t)\} = c \leq C, \forall j, \forall t \). Since \( \sum_{j=1}^{N} \mathbb{E}\{C_j(t)\} = c \), we can consider \( c \) as the overall expected channel capacity for the wireless system.

In the following, we introduce a modified multi-channel random access algorithm, which is an extension of the multi-channel random access algorithm presented in Section 6.2.2. The modified multi-channel random access algorithm is still composed of the exchange phase and the scheduling phase introduced in Section 6.2.2. After the scheduling phase, for any given link \( i \) and channel \( j \), if \( \mu_{ij}(t) = 1 \), link \( i \) is scheduled to transmit at a maximum capacity \( C_j(t) \) over channel \( j \) at time slot \( t \); otherwise (i.e., when \( \mu_{ij}(t) = 0 \)), link \( i \) is not scheduled transmission over channel \( j \). Thus, the queue dynamics (6.2.1) can be updated, \( \forall i, \forall t \geq 0 \), as follows:

\[
q_i(t) = [q_i(t-1) + A_i(t-1) - \sum_{j=1}^{N} C_j(t-1)\mu_{ij}(t-1)]^+.
\]  

(6.7.1)
Recall that the throughput optimality of the multi-channel random access algorithm stated in Theorem 6.2.1 relies on the time-scale separation assumption, i.e., the transmission weight function $h$ does not change significantly over time. However, under fading channels, the channel states $C_j(t)$, and hence the queue backlogs by queue dynamics (6.7.1), can change significantly over time. Therefore, Theorem 6.2.1 no longer holds, i.e., the modified multi-channel random access algorithm is not necessarily throughput-optimal given a finite number of channels $N$. This is not surprising, since single-channel random access algorithms [51, 83, 85, 28], which are throughput-optimal with a non-fading channel, are no longer throughput optimal for fading channels [63]. However, we will show in the following theorem that, under the modified multi-channel random access algorithm, the queuing behavior of individual links under fading channels still converges asymptotically to an equivalent deterministic single-queue system. By a steady state study of the equivalent queue system, we will show that the modified algorithm is still asymptotically throughput optimal under the regime of many fading channels.

**Theorem 6.7.1.** Under assumption (6.3.1), there exist an equivalent deterministic queue $q(t)$ and an equivalent schedule variable $v(t)$ for each time slot $t$ such that, $\forall i$,

$$q_i(t) \xrightarrow{P} q(t), \quad \text{and} \quad \sum_{j=1}^{N} C_j(t)\mu_{ij}(t) \xrightarrow{P} cE\{v(t)\},$$

with the deterministic dynamics of $q(t)$ and $E\{v(t)\}$ defined as follows:

$$q(t) = [q(t-1) + \alpha - cE\{v(t-1)\}]^+, \quad t \geq 0,$$  \hspace{1cm} (6.7.2)

$$E\{v(t)\} = [F_1 + (1-M)F_0(q(t-1))]E\{v(t-1)\} + F_0(q(t-1)).$$  \hspace{1cm} (6.7.3)

Theorem 6.7.1 states that the queuing behavior of individual links $\left(q_i(t), \sum_{j=1}^{N} C_j(t)\mu_{ij}(t)\right)$
converges (in the number of channels) to an equivalent deterministic single-queue system $(q(t), c\mathbb{E}\{v(t)\})$ with dynamics (6.7.2)(6.7.3). The proof of Theorem 6.7.1 follows the same steps in proving Theorem 6.3.1 (in Section 6.4) by replacing queue dynamics (6.2.1) with (6.7.1), and is omitted for brevity. Since the proofs of Theorem 6.6.1 and Theorem 6.7.1 both follow that of Theorem 6.3.1, Theorem 6.7.1 is readily extendable to the case of heterogeneous arrivals (i.e., assumption (6.6.1)).

Similar to the steady state study in Section 6.5, from dynamics (6.7.2)(6.7.3), we can find a steady state $(q, v)$ of the equivalent queue system $(q(t), c\mathbb{E}\{v(t)\})$ for any $\alpha < \frac{c}{M}$:

$$(q, v) = \left(h^{-1}\left(\frac{\alpha}{c - \alpha M}\right), \alpha\right).$$ (6.7.4)

Since $c$ is the expectation of the available channel resources shared by $M$ links, the interval of $0 < \alpha < \frac{c}{M}$ denotes the stabilizable range of (normalized) arrival rates. The result (6.7.4) shows that the queue backlog at the steady state is finite for any $\alpha < \frac{c}{M}$, and hence the modified multi-channel random access algorithm is asymptotically throughput-optimal with respect to $N$, which is summarized in the following proposition:

**Proposition 6.7.2.** The modified multi-channel random access algorithm is asymptotically throughput-optimal with respect to $N$ under fading channels with the channel state processes $C_j(t), j = 1, ..., M$, satisfying the assumptions introduced in the second paragraph of this section.

Note that similar to Theorem 6.3.1 and Proposition 6.5.2, we do not require stationary arrival processes and the time-scale separation assumption for Theorem 6.7.1 and Proposition 6.7.2 to hold.
6.8 Numerical Results

In this section, we provide extensive numerical results for the multi-channel random access algorithm. For analytical simplicity, we assume a unit capacity system, i.e., $C = 1$. Following the steady-state analysis in Section 6.5 and Section 6.7, we only consider the case $\alpha < \frac{1}{M}$ under non-fading channels ($\alpha < \frac{c}{M}$ under fading channels with $0 < c < 1$) since the maximal stabilizable normalized arrival rate cannot exceed $\frac{1}{M} \left( \frac{c}{M} \right)$ under fading channels as $N \to \infty$. The contention probability is fixed as $\beta = \frac{1}{M}$ [29]. The equivalent queue system $x(t) = (q(t), E\{v(t)\})$ is updated deterministically according to dynamics (6.3.7)-(6.3.11) under non-fading channels and dynamics (6.7.2)-(6.7.3) under fading channels. We focus on the following two metrics to represent the aggregated algorithm performance in simulation: The average queue backlog $\left( \frac{1}{M} \sum_{i=1}^{M} q_i(t) \right)$ and the average service rate $\left( \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \mu_{ij}(t) \right)$.

6.8.1 Convergence of Queue Backlogs and Scheduled Service Rates with Increasing $N$

In Figure 6.1, we illustrate the simulation results of the evolution of actual queue backlogs and scheduled service rates with difference number of channels $N$, compared to the equivalent queue system $(q(t), E\{v(t)\})$. Figure 6.1 shows that the queuing behavior of the aggregated links approaches the equivalent queue system $(q(t), E\{v(t)\})$ as the number of channels increases, where we note that $C = 1$ in the simulations. In the numerical evaluations, we consider $M = 10$ links (corresponding to a typical WiMAX femto cell scenario [25]). Specifically, we consider a scenario with normalized arrival rate $\alpha = 0.5/M$, the weight function $h(x) = e^x - 1$, and identical arrival processes $A_i(t) = \alpha$, $\forall i \forall t$. It is observed in Figure 6.1 that the accuracy of the approximation of the average queue backlog and the average service rate by $q(t)$ and $E\{v(t)\}$ increases as $N$ increases, which coincides with the convergence results in Corollary 6.3.3. Since the dynamics of $(q(t), E\{v(t)\})$ is deterministic and independent of the queuing behavior of individual links, the equivalent
queue system provides a simplified but accurate estimation for the actual queuing behavior under the multi-channel random access algorithm in the many-channel regime.

We note that $N = 1000$ is a practical choice for OFDM-based wireless systems. For example, the number of sub-carriers can be up to 2048 in WiMAX [1]. Thus, in the following numerical evaluations, we consider the case of $N = 1000$. 

Figure 6.1: Average queue backlogs and service rates with different $N$. 

![Graph showing queue dynamics and scheduled service rate](image-url)
6.8.2 Effect of Different Arrival Processes

In this section, we illustrate the effect of different arrival processes via simulations. Specifically, we consider a scenario with $M = 10$ links, $N = 1000$ channels, the weight function $h(x) = e^x - 1$, and normalized arrival rate of $\alpha = 0.8/M$. Figure 6.2 illustrates the dynamics of queue backlogs and scheduled service rates for different arrival processes as defined below. 

**Arrival Process 1**: i.i.d. arrivals with 

$$A_i(t) = \alpha + \text{rand}(1) \times \frac{0.2\alpha}{\sqrt{N}}, \forall i, \forall t,$$

where $\text{rand}(1)$ outputs a random value uniformly distributed over the interval $(0, 1)$ independently across time slots and links. 

**Arrival Process 2**: Markovian arrivals with 

$$A_i(t) = \alpha - \frac{0.2}{\sqrt{N}} \text{rand}(1)A_i(t - 1), \forall i, \forall t \geq 0,$$

and $A_i(-1) = \alpha, \forall i$. 

**Arrival Process 3**: Link-correlated arrivals with 

$$A_i(t) = \alpha + \frac{0.1}{\sqrt{N}} \text{rand}(1)A_{i-1}(t), \forall i > 1, \forall t,$$

and $A_1(t) = \alpha, \forall t$.

Although different processes have different effects on convergence, Figure 6.2 confirms that $q(t)$ represents the queue backlogs accurately for the three arrival processes we consider, which satisfy assumption (6.3.1).

6.8.3 Effect of Different Normalized Arrival Rates $\alpha$

We compare the queue dynamics and scheduled service rates under different normalized arrival rates $\alpha$ in Figure 6.3. Specifically, we consider a scenario with $M = 10$ links, $N = 1000$ channels, weight function $h(x) = e^x - 1$, and arrival processes $A_i(t) = \alpha, \forall i \forall t$. Note that $\alpha = \frac{0.95}{M}$ corresponds to a normalized arrival rate achieving 95% of the maximum stabilizable capacity. In conformance with Corollary 6.3.3, the equivalent queue
Figure 6.2: Average queue backlogs and service rates with different arrival processes.
Figure 6.3: Average queue backlogs and service rates with different $\alpha$. 

(a) Queue dynamics

(b) Scheduled service rate
system \((q(t), \mathbb{E}\{v(t)\})\) accurately tracks the (average) queue backlogs and scheduled service rates of the multi-channel random access algorithm. As expected, the queue backlogs increase with a growing normalized arrival rate \(\alpha\). In addition, while tracking the queuing behavior of the multi-channel random access algorithm, \((q(t), \mathbb{E}\{v(t)\})\) indeed converges to the steady state \((q, \alpha)\) over time, as illustrated in Figure 6.3. We also observe in Figure 6.3 that the convergence of \(q(t)\) to steady state is slower with a larger traffic load (i.e., a larger \(\alpha\)).

### 6.8.4 Effect of Different Weight Functions

In Figure 6.4, we compare the multi-channel random access algorithm with different weight functions \(h\): \(h(x) = e^x - 1\), \(h(x) = x\), and \(h(x) = \log(x + 1)\). Specifically, we consider a scenario with \(M = 10\) links, \(N = 1000\) channels, \(A_i(t) = \alpha, \forall i \forall t\), and a heavy load \(\alpha = 0.8/M\). The results again show that, while tracking the average queue backlog, \(q(t)\) converges to and attains its limit \(q\) as theoretically calculated in Table 6.1. In conformance with Table 6.1, the more aggressive the weight function \(h\) is, the higher congestion level (illustrated as the average queue backlog in Figure 6.4) the system experiences.

### 6.8.5 Queuing Behavior with Heterogeneous Arrival Rates

In this section, we study the case of heterogeneous arrival rates. Specifically, we consider \(M = 20\) links with \(A_i(t) = \alpha_1 = 0.03, i = 1, 2, ..., 10\), and \(A_i(t) = \alpha_{11} = 0.05, i = 11, 12, ..., 20\). The queuing behavior of the system is illustrated in Figure 6.5 with \(N = 1000\), \(h(x) = e^x - 1\), where we take the average of the queue backlogs that share the same arrival rates. We observe that the queue backlogs with the same arrival rate evolve in time around equivalent queues \((\bar{q}_i(t))\) and converge to the steady state value \((\bar{q}_i)\) calculated according to (6.6.4). Thus, we can use \((\bar{q}_i)\) to estimate the congestion level of the queue backlogs in the system.
Figure 6.4: Dynamics of average queue backlogs with different weight function $h$

Figure 6.5: Dynamics of queue backlogs with heterogenous arrival rates
6.8.6 Results under Fading Channels

In this section, we numerically evaluate the performance of the modified multi-channel random access algorithm introduced in Section 6.7 under fading channels. Specifically, we consider $N = 1000$ independent fading channels with channel states i.i.d. distributed over time as follows (for any channel $j$, $\forall t$): $C_j(t) = \frac{0.1}{N}$, w.p. $0.5(1 - c)$; $C_j(t) = \frac{0.3}{N}$, w.p. $0.5(1 - c)$; $C_j(t) = \frac{0.6}{N}$, w.p. $0.5(1 - c)$; and $C_j(t) = \frac{1}{N}$, otherwise. Here, $c$ denotes the expectation of the channel states, i.e., $\sum_{j=1}^{N} \mathbb{E}\{C_j(t)\} = c$, and we let $0.4 < c < 1$ in this simulation.

We consider $M = 10$ links with a load $\alpha = \frac{0.49}{M}$ and $A_i(t) = \alpha$, $\forall i \forall t$. Figure 6.6(a) illustrates the queue dynamics of the modified multi-channel random access algorithm with different $c$. The queue backlog can still be tracked by the equivalent queue $q(t)$ which is updated independently and deterministically according to the equivalent queue system (6.7.2)(6.7.3). We observe that $q(t)$ converges in time to its steady state $q$ calculated by (6.7.4). In addition, as expected, the congestion level (i.e., the queue backlog) is lower with an increase in channel resources (i.e., a larger $c$).

To evaluate the throughput performance of the multi-channel random access algorithm under fading channels, we consider a scenario with an arrival rate achieving 98% of the maximum stabilizable capacity: $\alpha = \frac{0.49}{M}$ and $c = 0.5$. Figure 6.6(b) shows that the queue backlog, tracked by the equivalent queue $q(t)$, converges in time to $q = 1.7750$ (calculated by (6.7.4)). Hence, the modified multi-channel random access algorithm stabilizes the queue backlog with an arrival rate that achieves 98% of the maximum capacity.

6.9 Conclusions

Our work aims to better understand the delay performance and the fundamental properties of queuing dynamics for random access algorithms in a many-channel regime. Specifically, in this work, we generalize a class of throughput-optimal random access algorithms to
(a) Queue dynamics under fading channels with different $c$

(b) Queue dynamics with an arrival rate achieving 98% of the maximum capacity

Figure 6.6: Dynamics of average queue backlogs under fading channels.
a multi-channel setting. We show that the individual queuing behavior under the multi-channel random access algorithm converges to an equivalent deterministic single-queue system. By analyzing this equivalent queue system, we find a closed-form approximation for the steady state queue backlogs in the many-channel regime. We also show that the multi-channel random access algorithm is asymptotically throughput-optimal in the scenario of heterogeneous arrivals and fading channels.

6.10 Distributed Scheduling and its Asymptotic Analysis for Cognitive Radio Networks under the Many-Channel Regime

Developing efficient scheduling algorithms for CRNs is essential to garner the full potential of cognitive radio networks. In recent years, centralized opportunistic scheduling algorithms have been developed for cognitive radio networks. Throughput-optimal cooperative scheduling has been studied in [59, 103, 112], where PUs are aware of SU activities and SUs cooperatively relay PU data. Non-cooperative scheduling has been studied in [102, 70, 108] to achieve optimal SU throughput/utility. However, the above algorithms, though throughput-optimal, are centralized with high time complexity and hence not suitable for practical implementations. In addition to computational complexity, these algorithms do not work when a global centralized component is not available (e.g., a scenario where SUs transmit peer-to-peer without centralized control and only local information is available). Therefore, low-complexity and distributed algorithms are needed to deploy efficient and high performance CRNs. While heuristic distributed solutions have been proposed in the literature (e.g., [95]), the design of distributed algorithms for CRNs with provable properties (i.e., provable throughput/utility performance) remains an open research problem.

Distributed scheduling algorithms for traditional wireless networks have been proposed in the literature over the last decade. Earlier examples [22, 14] achieve at least certain fractions of the optimal throughput in single-channel wireless networks. More recently,
throughput optimality has been achieved with distributed queue-length-based scheduling algorithms [51, 83, 85, 28] for the same setting. Among the few attempts to design distributed scheduling algorithms for multi-channel networks, [35] guarantees at least a certain fraction (dependent on the network interference model) of the optimal throughput. However, these algorithms have been designed for wireless networks with a non-fading channel capacity, and thus are not suitable for CRNs where channel states are modulated by PU activities.

In this section, we propose a distributed scheduling algorithm, called collision-queue-regulated algorithm, for a fully-connected CRN in the many-channel regime. We consider an OFDM setting (which is the basis for IEEE 802.22 standard for cognitive radio networks), where spectrum is partitioned into tens or hundreds of orthogonal sub-channels. With collision rate constraints imposed by PUs, the algorithm achieves at least $\frac{1}{e}$ fraction of the capacity region as the number of channels grows, where $e$ is the Euler number. We also show via simulation results that the throughput performance is actually close to the optimal.

Salient contributions of our work are summarized in the following:

1) Under the collision-queue-regulated algorithm, the collision rates observed by the PUs are upper bounded by an arbitrary threshold;

2) We design a novel equivalent queuing system such that the queues in the original CRN converge asymptotically (with respect to the number of channels) to this system;

3) By analyzing the equivalent queuing system, we show that the collision-queue-regulated algorithm achieves at least $\frac{1}{e}$ fraction of the capacity region in the many-channel regime.

The rest of the section is organized as follows: The network model and the distributed collision-queue-regulated algorithm are described in Section 6.10.1. This is followed by an asymptotic analysis on the queuing behavior and the throughput performance in Section 6.10.2. Numerical results are provided in Section 6.10.3, and the section is concluded in Section 6.10.4.
6.10.1 Network Model and the Distributed Algorithm

Network Elements

Consider a time-slotted cognitive radio network (CRN) composed of a PU system and an SU system. We consider a fully-connected SU system composed of a set \( L \) of single-hop directional SU communication links, with \( |L| = L \), i.e., when the PU system is idle, the transmission of SU link \( i \in L \) over a channel fails if and only if there is a simultaneous transmission of another link \( l \in L, l \neq i \), over the same channel. Let \( A_i(t) \) be the amount of data (in unit of bits) arriving at SU link \( i \in L \) at the beginning of time slot \( t \). For analytical simplicity, we assume the arrivals are uniform and constant, i.e., \( A_i(t) = \lambda, \forall i \in L, \forall t \), with arrival rate \( \lambda \).

OFDM has proven to be one of the prime candidates for CRNs (e.g., IEEE 802.22 standard). The reason is two-fold: (i) We have irregular openings in the PU spectrum, and OFDM helps with collectively utilizing non-contiguous PU channels in one SU transmission; (ii) Inter-symbol interference (ISI) can be significantly reduced in an OFDM system by transmitting data in parallel over a large number of low-rate subchannels [118]. Thus, we consider an OFDM mechanism for SUs’ channel access: an SU link can transmit its data opportunistically over multiple PU channels in a time slot.

We consider a multiple-PU scenario. Specifically, the CRN is synchronized with a time-slotted PU system comprised of \( M \) PUs and \( N \) orthogonal PU subchannels, which we refer to as channels for short in the following. Each PU \( k \in \{1, 2, \ldots, M\} \) is licensed to a set of \( I_k \) with \( |I_k| = n_kN \) channels, where \( (n_k) \) are constant fractional numbers satisfying \( \sum_{k=1}^{M} n_k = 1 \). We assume each channel has a uniform capacity (i.e., maximum data rate in bit per time slot) equal to \( \frac{K}{N} \), where we can consider \( K \) (bit per time slot) as the total capacity of the considered PU system. Note that the growing number of channels leads to diminishing bandwidth per channel where the sum of all bands is constant \( K \) and the sum of licensed bands for PU \( k \) is \( n_kK \), which conforms to the setting of OFDM systems with a large number of low-rate channels [118]. We assume that PU activities are independent
over PUs and that the activities of each PU $k$ evolve according to an ON-OFF Markovian process $C_k(t)$: At time slot $t$, we let $C_k(t) = 1$ if PU $k$ is busy (PU $k$ is in ON state and occupies the entire set of channels $I_k$) and $C_k(t) = 0$ if PU $k$ is idle (PU $k$ is in OFF state and the entire set of channels $I_k$ are available to SUs). For analytical simplicity, we assume the process $C_k(t)$ starts with a steady state distribution at $t = 0, \forall k$. We denote by $H(t) = (C_k(t-1))_{k=1}^M$ the channel availability information of SUs at time slot $t$. Note that the exact knowledge of $(C_k(t))$ may not be available to SUs due to time-varying PU activities or sensing overheads. Thus,

$$S_k(t) \triangleq \mathbb{E}\{1 - C_k(t)|C_k(t-1)\} = \mathbb{E}\{1 - C_k(t)|H(t)\}$$

defines the probability that PU $k$ is OFF given $H(t)$, which is known to the SUs at time slot $t$. We note that $S_k(t)$ is simply the transition probability of $C_k(t)$ and can be obtained by SUs via the knowledge or observation of PU data traffic statistics.

Let $\mu_{ij}(t) \in \{0, 1\}$ denote the schedule of SU link $i \in \mathcal{L}$ over channel $j$ at time slot $t$, with $j = 1,...,N$. Specifically, $\mu_{ij}(t) = 1$ if SU link $i$ is scheduled over channel $j$; $\mu_{ij}(t) = 0$, otherwise. For analytical simplicity, we let $\mu_{ij}(0) = 0, \forall i,j$. Note that when $\mu_{ij}(t) = 1$, SU link $i$ is scheduled to transmit up to $\frac{K}{N}$ bits over channel $j$ in one time slot. We say a collision with PU $k$ occurs if $\mu_{ij}(t)C_k(t) = 1, j \in I_k$, i.e., there is a scheduled SU data transmission when PU $k$ is busy. Thus, for each SU data queue $q_i(t), i \in \mathcal{L}$, we have the following queue dynamics:

$$q_i(t) = [q_i(t - 1)$$

$$- \frac{K}{N} \sum_{k=1}^{M} \sum_{j \in I_k} (\mu_{ij}(t-1)(1 - C_k(t-1)))) + A_i(t-1)]^+, \quad (6.10.1)$$

with $q_i(0) = 0, \forall i \in \mathcal{L}$. To constrain the potential interference caused by the SUs to the PU system, we require that the collision rate (caused by any SU link $i$) observed by any given PU $k$ be upper-bounded by a maximum collision rate $\rho_k$ (normalized by the number
of channels licensed to PU $k$):

$$
\limsup_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{j \in I_k} \mu_{ij}(t)C_k(t) \leq \rho_k, \forall i \in L.
$$

(6.10.2)

As suggested in [16], for the considered OFDM-based CRN, we assume the existence of an out-of-band common control channel (CCC) which is not interrupted by PU activities. In the collision-queue-regulated algorithm, the exchange of local control information is performed over the CCC at the beginning of each time slot. Since CCC is dedicated only to the transmission and reception of control messages, CCC can utilize the small portions of the guard bands between the licensed channels [16].

Collision-Queue-Regulated Algorithm

In the following, we propose a distributed collision-queue-regulated algorithm. We will show in Section 6.10.2 that the proposed algorithm can achieve at least $\frac{1}{e}$ fraction of the capacity region asymptotically with respect to $N$.

We maintain a virtual collision queue $X_{ik}(t)$ at each SU link $i \in L$ corresponding to any given PU $k$, to assist the development of the proposed algorithm. Specifically, the queue dynamics of $X_{ik}(t)$ is defined as, $\forall i \in L, \forall k$,

$$
X_{ik}(t) = [X_{ik}(t-1) - \rho_k + \frac{1}{n_kN} \sum_{j \in I_k} \mu_{ij}(t-1)C_k(t-1)]^+,
$$

(6.10.3)

with $X_{ik}(0) = 0$. We note that the collision rate constraint (6.10.2) is satisfied if collision queues $X_{ik}(t)$ are stable.

At the beginning of each time slot $t$, the collision-queue-regulated algorithm consists of two phases: Exchange Phase and Scheduling Phase, the duration of which we assume is negligible compared to that of a unit time slot. The exchange phase is detailed as follows:
Exchange Phase:

The exchange phase takes place over the CCC. Specifically, the transmitter of each SU link \( i \in \mathcal{L} \) broadcasts the following three binary vectors to all its neighbors (its intended receiver and all nodes in \( \mathcal{L} \)) over the CCC: its schedules at the previous time slot \( (\mu_{ij}(t-1))_{j=1}^{N} \), a vector of contention variables \( (a_{ij}(t))_{j=1}^{N} \), and a vector of transmission variables \( (p_{ij}(t))_{j=1}^{N} \).

The contention variables \( (a_{ij}(t)) \) are i.i.d. over SUs \( i \) and channels \( j \) with

\[
    a_{ij}(t) = \begin{cases} 
        1, & \text{w.p. } \frac{1}{L} \\
        0, & \text{w.p. } \frac{L-1}{L}.
    \end{cases}
\]

The transmission variables \( (p_{ij}(t)) \) are independent over channels \( j \) and SUs \( i \) with

\[
    p_{ij}(t) = \begin{cases} 
        1, & \text{w.p. } \frac{e^{y_{ik}(t)}}{e^{y_{ik}(t)}+1} \\
        0, & \text{w.p. } \frac{1}{e^{y_{ik}(t)}+1},
    \end{cases}
\]

where \( j \in \mathcal{I}_k \) and the collision-queue-regulated weight \( y_{ik}(t) \) is defined as

\[
    y_{ik}(t) \triangleq q_i(t-1)S_k(t) - \gamma X_{ik}(t-1)(1 - S_k(t)), \tag{6.10.4}
\]

and \( \gamma > 0 \) is a constant parameter that serves as a weight to the collision queue \( X_{ik}(t-1) \).

After the exchange phase, the transmitter and receiver of each SU link \( i \) have the following information: \( (\mu_{ij}(t-1))_{j=1,...,N}, (a_{ij}(t))_{j=1,...,N}, (p_{ij}(t))_{j=1}^{N}) \), which will be used to determine the transmission schedules for SU link \( i \).

To assist the development of scheduling phase, we define the following three conditions, for any given SU link \( i \) and channel \( j \). *Condition (i):* The “contention” of SU link \( i \) for channel \( j \) is successful, i.e., \( a_{ij}(t) \Pi_{l \in \mathcal{L} \setminus \{i\}} (1 - a_{lj}(t)) = 1 \). *Condition (ii):* \( \sum_{l \in \mathcal{L} \setminus \{i\}} \mu_{lj}(t-1) = 0 \), i.e., none of the neighbors were scheduled at the previous time slot. *Condition (iii):* The transmission variable \( p_{ij}(t) = 1 \).

219
The scheduling phase is introduced as follows:

**Scheduling Phase:**

The transmitter and the receiver of each SU link \(i\) determine the schedules \(\mu_{ij}(t)\), \(j = 1, ..., N\), according to the following:

**Case 1:** \(\mu_{ij}(t) = 1\) if Conditions (i)(ii)(iii) hold.

**Case 2:** If Condition (i) does not hold and Condition (iii) holds, then \(\mu_{ij}(t) = \mu_{ij}(t-1)\).

**Case 3:** Otherwise, \(\mu_{ij}(t) = 0\).

According to the scheduling phase, we conclude that, \(\forall i \in L\), \(\forall j \in \{1, 2, ..., N\}\),

\[
\mu_{ij}(t) = p_{ij}(t) \times \{a_{ij}(t)\Pi_{l \in L \setminus \{i\}}(1 - a_{lj}(t))(1 - \sum_{l \in L \setminus \{i\}} \mu_{lj}(t - 1)) + [1 - a_{ij}(t)\Pi_{l \in L \setminus \{i\}}(1 - a_{lj}(t))]\mu_{ij}(t - 1)\},
\]

where the first and second terms in the \(\{\cdot\}\) in (6.10.5) correspond to Case 1 and Case 2 in the scheduling phase, respectively.

Since both the transmitter and the receiver of SU link \(i \in L\) have a copy of the schedule vector \((\mu_{ij}(t))_{j=1}^{N}\) when the scheduling phase ends, they will tune to the set of channels \(\{j : \mu_{ij}(t) = 1\}\) for SU data transmission in the remaining time slot \(t\).

We show in Proposition 6.10.1 that the collision-queue-regulated algorithm is feasible in that interfering links are never scheduled over a same channel in any time slot.

**Proposition 6.10.1.** The collision-queue-regulated algorithm provides a feasible schedule for each time slot \(t\), i.e., \(\forall i, j, t:\ \sum_{l \in L \setminus \{i\}} \mu_{ij}(t) = 0\), if \(\mu_{ij}(t) = 1\).

Proposition 6.10.1 can be proved easily by mathematical induction over time slot \(t\) and we omit the proof for brevity.
6.10.2 Performance Analysis in a Many-Channel Regime

In Section 6.10.2, we show that under the collision-queue-regulated algorithm, the original system of the queue lengths \( q_i(t) \) and the collision queues \( X_{ik}(t) \) converge to an equivalent queue system as the number of channels \( N \) grows. Based on the analysis of the equivalent queue system, we show that the algorithm achieves at least \( \frac{1}{e} \) fraction of the capacity region asymptotically with respect to \( N \).

Asymptotic Queuing Behavior of the Collision-Queue-Regulated Algorithm

We present the asymptotic queuing behavior of the collision-queue-regulated algorithm in Theorem 6.10.2.

**Theorem 6.10.2.** Given \( \mathcal{H}'(t) \equiv (\mathcal{H}(t), \mathcal{H}(t-1), ..., \mathcal{H}(1)) \), there exists an equivalent queuing system \( (q(t), (x_i(t))_{i=1}^M) \) with equivalent schedule variables \( (u_k(t))_{k=1}^M \), such that the following four arguments (I\((t)\), II\((t)\), III\((t)\), and IV\((t)\)) hold under the collision-queue-regulated algorithm for each time slot \( t \): I\((t)\): The queue lengths \( q_i(t) \) and the collision queue lengths \( X_{ik}(t) \) converge to \( q(t) \) and \( x_{ik}(t) \), respectively:

\[
q_i(t) \xrightarrow{P} N q(t), \quad X_{ik}(t) \xrightarrow{P} N x_{ik}(t), \quad \forall i \in \mathcal{L}, \forall k,
\]

\[
(6.10.6)
\]

where \( \xrightarrow{P} N \) denotes the convergence in probability \([56]\) as \( N \to \infty \). II\((t)\): The schedules \( \mu_{ij}(t) \) converge to the equivalent schedule variable \( u_k(t) \) with \( j \in \mathcal{I}_k \):

\[
\mu_{ij}(t) \xrightarrow{L} N u_k(t), \quad \forall i, \forall k, \forall j \in \mathcal{I}_k,
\]

\[
(6.10.7)
\]

where \( \xrightarrow{L} N \) denotes the convergence in distribution \([56]\) as \( N \to \infty \). III\((t)\): The schedules \( \mu_{ij}(t) \) follow a Law of Large Numbers (LLN):

\[
\frac{1}{n_k N} \sum_{j \in \mathcal{I}_k} \mu_{ij}(t) \xrightarrow{P} N \mathbb{E}\{u_k(t)|\mathcal{H}'(t)\}, \quad \forall i \in \mathcal{L}, \forall k.
\]

\[
(6.10.8)
\]
IV(t): The schedules \( \mu_{ij}(t) \) are asymptotically mutually independent. Specifically, for any given SU links \( i_1, i_2 \in L \), and any two distinct channels \( j_1 \neq j_2 \in \{1, 2, ..., N\} \), the scheduling decisions are independent, i.e., \( \forall w_1, w_2 \in \{0, 1\} \),

\[
\lim_{N \to \infty} \Pr\{\mu_{i_1 j_1}(t) = w_1, \mu_{i_2 j_2}(t) = w_2 | H'(t)\} = \Pr\{u_{k_1}(t) = w_1 | H'(t)\} \Pr\{u_{k_2}(t) = w_2 | H'(t)\},
\]

(6.10.9)

where \( k_1 \) and \( k_2 \) satisfy \( j_1 \in I_{k_1} \) and \( j_2 \in I_{k_2} \).

The equivalent queuing system \((q(t), (x_k(t))_{k=1}^M)\) and the equivalent schedule variables \((u_k(t))_{k=1}^M\) evolve as follows,

\[
q(t) = \lfloor q(t-1) - \sum_{k=1}^{M} (n_k K(1-C_k(t-1)) \times \mathbb{E}\{u_k(t-1)|H'(t-1)\}) + \lambda \rfloor^+,
\]

(6.10.10)

\[
x_k(t) = \lfloor x_k(t-1)
- \rho_k + C_k(t-1)\mathbb{E}\{u_k(t-1)|H'(t-1)\} \rfloor^+, \forall k,
\]

(6.10.11)

\[
u_k(t) = U_k(t) u_k(t-1) + V_k(t)(1-u_k(t-1)),
\]

(6.10.12)

where \( U_k(t) \) and \( V_k(t) \) are independent over time and PUs and defined as follows:

\[
U_k(t) = \begin{cases} 
1, \text{ w.p. } (1-(L-1)\beta) \frac{e^{y_k(t)}}{e^{y_k(t)}+1}, \\
0, \text{ otherwise},
\end{cases}
\]

\[
V_k(t) = \begin{cases} 
1, \text{ w.p. } \beta \frac{e^{y_k(t)}}{e^{y_k(t)}+1}, \\
0, \text{ otherwise},
\end{cases}
\]
with

$$\beta \triangleq \frac{(L - 1)^{L-1}}{L^L},$$

$$y_k(t) \triangleq q(t - 1)S_k(t) - \gamma x_k(t - 1)(1 - S_k(t)).$$

The initial conditions of the equivalent queue system are set as:

$$q(0) = 0, \quad x_k(0) = 0, \quad \text{and} \quad u_k(0) = 0, \quad \forall k. \quad (6.10.13)$$

**Proof.** The proof for Theorem 6.10.2 is provided in Section 6.10.5. □

**Remark 6.10.3.** According to (6.10.6) in Theorem 6.10.2, given $\mathcal{H}'(t)$, the data queues $q_i(t)$ and the collision queues $X_{ik}(t)$ converge (in probability) asymptotically to deterministic equivalent queues $q(t)$ and $x_k(t)$, respectively. By the dynamics of $u_k(t)$ in (6.10.12), we find the dynamics of $\mathbb{E}\{u_k(t)|\mathcal{H}'(t)\}$ as follows:

$$\mathbb{E}\{u_k(t)|\mathcal{H}'(t)\} = \beta \frac{e^{y_k(t)}}{e^{y_k(t)} + 1}$$

$$+ (1 - L\beta) \frac{e^{y_k(t)}}{e^{y_k(t)} + 1} \mathbb{E}\{u_k(t - 1)|\mathcal{H}'(t - 1)\},$$

where we note that $u_k(t - 1)$ is independent of $\mathcal{H}(t)$ given $\mathcal{H}'(t - 1)$.

In the following, we will study the stability of the equivalent queuing system $(q(t), (x_k(t))_{k=1}^M)$, which becomes the asymptotic network stability (i.e., the stability for the data queues $q_i(t)$ and the collision queues $X_{ik}(t)$) under the collision-queue-regulated algorithm.

**Performance Analysis**

We have shown through Theorem 6.10.2 that the equivalent system $(q(t), (x_k(t))_{k=1}^M)$ can represent the asymptotic queuing behavior of the data queues $q_i(t)$ and the collision queues $X_{ik}(t)$. We will show that under the collision-queue-regulated algorithm, $(q(t), (x_k(t))_{k=1}^M)$ are stable for at least $\frac{1}{e}$ fraction of the capacity region.
Specifically, we define the capacity region $\Lambda$ as

$$\Lambda = \{\lambda \geq 0 : \text{the arrivals } A_i(t) = \lambda, \forall i \in \mathcal{L},$$

are stabilizable by some scheduling algorithm}.

For any given $0 < \alpha < 1$, we let $\alpha \Lambda$ denote an $\alpha$ fraction of the capacity region such that

$$\alpha \Lambda \triangleq \{\lambda \geq 0 : \exists \lambda' \in \Lambda \text{ s.t. } \frac{\lambda}{\alpha} < \lambda'\}.$$

Before we present the asymptotic stability in Theorem 6.10.6, we introduce the following two lemmas to assist the proof of Theorem 6.10.6.

**Lemma 6.10.4.** For any given $0 < \delta < L\beta$, there exists $B_2(\delta) > 0$ such that for any time slot $t$, whenever $y_k(t) \geq B_2$ for any given PU $k$, we have

$$\Pr\{u_k(t) = 1\} \geq \beta - \frac{\delta}{L}.$$

**Proof.** Let $B_2 \triangleq \log\left(\frac{L\beta}{\delta} - 1\right)$. By taking the expectation of both sides of (6.10.14) over $H'(t)$ conditioned on $y_k(t) \geq B_2$, we have

$$\mathbb{E}\{u_k(t)|y_k(t) \geq B_2\}$$

$$= \beta \mathbb{E}\left\{\frac{e^{y_k(t)}}{e^{y_k(t)} + 1}|y_k(t) \geq B_2\right\}$$

$$+ (1 - \beta L)\mathbb{E}\left\{\frac{e^{y_k(t)}}{e^{y_k(t)} + 1}u_k(t-1)|y_k(t) \geq B_2\right\}$$

$$\geq \beta \frac{e^{B_2}}{e^{B_2} + 1} = \beta - \frac{\delta}{L}.$$

We show that for any $\lambda' \in \Lambda$, there exist (auxiliary) random variables $(\mu_k^{STAT}(t))_{k=1}^M$ for each time slot $t$ satisfying the properties described in Lemma 6.10.5.

**Lemma 6.10.5.** For any $\lambda' \in \Lambda$, there exist random variables $(\mu_k^{STAT}(t))$ with $\mu_k^{STAT}(t) \in$
$\{0,1\}$ \( \forall k \), which are dependent only on \( \mathcal{H}(t) \) for each time slot \( t \), such that the following holds:

\[
\sum_{k=1}^{M} \mathbb{E}\{n_k K \mu_k^{SAT}(t) S_k(t)\} = \lambda',
\]
(6.10.15)

\[
\mathbb{E}\{\mu_k^{SAT}(t)(1 - S_k(t))\} \leq \rho_k,
\]
(6.10.16)

\[
\mathbb{E}\{\mu_k^{SAT}(t) | \mathcal{H}(t)\} \leq \frac{1}{L}, \forall \mathcal{H}(t).
\]
(6.10.17)

**Proof.** Proof of Lemma 6.10.5 is provided in Section 6.10.6.

Utilizing Lemma 6.10.4 and Lemma 6.10.5, we show in Theorem 6.10.6 that the equivalent system is stable for at least a constant fraction of the capacity region \( \Lambda \) under the collision-queue-regulated algorithm.

**Theorem 6.10.6.** \( \forall \lambda \in \alpha \Lambda \), with \( \alpha = \left(\frac{L - 1}{L}\right)^{L-1} \), \( q(t) \) and \( x(t) \) are stable under the collision-queue-regulated algorithm, i.e.,

\[
\limsup_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{q(t) + \sum_{k=1}^{M} x_k(t)\} \leq \frac{B_3}{\epsilon_2},
\]
(6.10.18)

where positive constants \( B_3 \) and \( \epsilon_2 \) will be defined in the proof.

**Proof.** The proof of Theorem 6.10.6 is provided in Section 6.10.7, where we have employed Lemma 6.10.4 and Lemma 6.10.5.

The following corollary directly follows Theorem 6.10.6, where we note that \( \left(\frac{L - 1}{L}\right)^{L-1} > \frac{1}{e}, \forall L > 0 \).

**Corollary 6.10.7.** \( \forall \lambda \in \frac{1}{e} \Lambda \), \( q(t) \) and \( (x_k(t)) \) are stable under the collision-queue-regulated algorithm.

Since the queue lengths \( q_i(t) \) and the collision queues \( X_{ik}(t) \) converge to \( q(t) \) and
(x_k(t)), respectively, asymptotically with respect to N, the collision-queue-regulated algorithm achieves at least \( \alpha = \frac{1}{e} \) fraction of the capacity region in the many-channel regime by Theorem 6.10.6.

We refer to this fraction \( \alpha \) as the efficiency factor of the capacity region. For comparison, the efficiency factor of the distributed PLDS algorithm [35] proposed for a general multi-radio multi-nonfading-channel wireless network is \( \frac{1}{3e} \). Note that although both algorithms are proposed for a multi-channel scenario, the setting for the collision-queue-regulated algorithm is more stringent than that for PLDS, in that non-fading channels are assumed in [35] while we consider channels modulated by PU activities in this work. Yet, the provable efficiency factor of the collision-queue-regulated algorithm is larger than that of PLDS.

### 6.10.3 Numerical Results

In this section, via simulation, we compare the throughput performance of the proposed algorithm with a back-pressure-based centralized throughput-optimal algorithm, denoted as the BP algorithm. The BP algorithm is based on the throughput-optimal back-pressure algorithm [100] where we substitute the (generic) weight in [100] with the collision-queue-regulated weight \( y_{ik}(t) \) defined in (6.10.4) for each \( i \in \mathcal{L} \) and \( k = 1, 2, ..., M \). It can be shown, similar to the analysis [103][113] that the BP algorithm is optimal given the collision rate constraints. We consider a fully connected CRN with 10 SU communication links. We set the parameter in (6.10.4) as \( \gamma = 1 \) and consider two PUs with \( N = 100, n_1 = n_2 = 0.5 \).

The channel states of PUs evolve according to the transition diagram in Figure 6.7, where the “busy” and the “idle” states are represented as \( C_k(t) = 1 \) and \( C_k(t) = 0 \), respectively. Note that in Figure 6.7, \( p_{01}^k \) and \( p_{10}^k \) represent channel \( k \)’s transition probability from the idle state to the busy state and that from the busy state to the idle state, respectively, \( k = 1, 2 \). In the numerical evaluation, we let \( p_{01}^1 = 0.3, p_{10}^1 = 0.7 \) and \( p_{01}^2 = 0.4, p_{10}^2 = 0.6 \).

We illustrate the stability of queues through Figure 6.8 under the collision rate constraint \( \rho = 0.1 \), where queue backlogs are summed over all links/channels. In our numerical studies, we have observed that both algorithms stabilize data and collisions queues for \( \lambda = 0.064 \).
Again, both algorithms fail to stabilize the system for $\lambda = 0.066$, where the data queues keep growing, indicating network instability. Since the BP algorithm is throughput-optimal, we can expect that the maximum stabilizable $\lambda$ is in between 0.064 and 0.066. Hence, the collision-queue-regulated algorithm achieves at least $0.064/0.066 = 97\%$ of the throughput optimality under this simulation setting. Note that 0.97 is significantly higher than the efficiency factor $\alpha = \frac{1}{e} = 0.37$ in Theorem 6.10.6.
We now post a more stringent collision rate constraint \( \rho = 0.03 \) in Figure 6.10.3. Under both algorithms, at \( \lambda = 0.054 \), the data queues and collision queues are stable; at \( \lambda = 0.058 \), the data queues and collision queues are both increasing over the time slots \( t \), indicating network instability and collision rate violation. That is, both algorithms can stabilize \( \lambda = 0.054 \) but cannot stabilize \( \lambda = 0.058 \). We can expect that the collision-queue-regulated algorithm achieves at least \( 0.054/0.058 = 93\% \) of the throughput optimality under this simulation setting.

6.10.4 Conclusions

In this section, we proposed a distributed collision-queue-regulated scheduling algorithm for cognitive radio networks. We proved theoretically that the proposed algorithm can achieve at least \( \frac{1}{e} \) of the capacity region asymptotically in the many-channel regime via a novel equivalent queue system analysis. We also illustrated through numerical evaluation that the throughput performance of the proposed algorithm is close to optimal.
6.10.5 Proof of Theorem 6.10.2

We prove Theorem 6.10.2 by mathematical induction over timeslot \( t \). Given the initial conditions (6.10.13), the base case holds for time slot \( t = 0 \). Suppose the induction hypothesis (I\((t-1)\), II\((t-1)\), III\((t-1)\), and IV\((t-1)\)) holds, we prove I\((t)\), II\((t)\), III\((t)\), and IV\((t)\) hold in Section 6.10.5, Section 6.10.5, Section 6.10.5, and Section 6.10.5, respectively.

**Proof of I\((t)\)**

Given any SU link \( i \in \mathcal{L} \), I\((t-1)\) and III\((t-1)\), we have

\[
q_i(t-1) - \frac{K}{N} \sum_{k=1}^{M} \sum_{j \in I_k} ((1 - C_k(t-1)) \mu_{ij}(t-1)) + A_i(t-1)
\]

\[
\overset{P}{\to}_N q(t-1)
\]

\[
- \sum_{k=1}^{M} (q_k(t-1) K (1 - C_k(t-1)) \mathbb{E}\{u_k(t-1)|H(t-1)\}) + \lambda.
\]

By queue dynamics (6.10.1)(6.10.10) and the continuity of \([·]^+\), we conclude that \( q_i(t) \overset{P}{\to}_N q(t), \forall i \in \mathcal{L} \).

Similarly, we have

\[
X_{ik}(t-1) - \rho_k + \frac{1}{n_k N} \sum_{j \in I_k} \mu_{ij}(t-1) C_k(t-1)
\]

\[
\overset{P}{\to}_N x_k(t-1) - \rho_k + C_k(t-1) \mathbb{E}\{u_k(t-1)|H(t-1)\}.
\]

By queue dynamics (6.10.3)(6.10.11) and the continuity of \([·]^+\), we conclude that \( X_{ik}(t) \overset{P}{\to}_N x_k(t) \), which completes the proof of I\((t)\).
Proof of II(t)

Given any \( i \in \mathcal{L}, \) PU \( k, \) and \( j \in \mathcal{I}_k, \) by taking conditional expectation on both sides of (6.10.5), we obtain the following:

\[
\mathbb{E}\{\mu_{ij}(t) \mid q_i(t-1), X_{ik}(t-1), (\mu_{lj}(t-1))_{l \in \mathcal{L}}, \mathcal{H}'(t)\} = \beta \frac{e^{y_{ik}(t)}}{e^{y_{ik}(t)} + 1} + \frac{e^{y_{ik}(t)}}{e^{y_{ik}(t)} + 1} \left[ (1 - \beta)\mu_{ij}(t-1) - \beta \sum_{l \neq i} \mu_{lj}(t-1) \right],
\]

where we recall that \( y_{ik}(t) \) is defined in (6.10.4) and \( S_k(t) \) in \( y_{ik}(t) \) is defined as \( S_k(t) \equiv \mathbb{E}\{1 - C_k(t) \mid \mathcal{H}(t)\}. \) By taking the expectation over \( (q_i(t-1), X_{ik}(t-1), (\mu_{lj}(t-1))_{l \in \mathcal{L}}) \) on both sides of the above equation, we conclude

\[
\mathbb{E}\{\mu_{ij}(t) \mid \mathcal{H}'(t)\} = \mathbb{E}\{\beta \frac{e^{y_{ik}(t)}}{e^{y_{ik}(t)} + 1} + \frac{e^{y_{ik}(t)}}{e^{y_{ik}(t)} + 1} \left[ (1 - \beta)\mu_{ij}(t-1) - \beta \sum_{l \neq i} \mu_{lj}(t-1) \right] \mid \mathcal{H}'(t-1)\},
\]

where \( S_k(t) \) in \( y_{ik}(t) \) in the above equation is deterministic and we note that \( q_i(t-1), X_{ik}(t-1), \) and \( (\mu_{lj}(t-1))_{l \in \mathcal{L}} \) are independent of \( \mathcal{H}(t) \) given \( \mathcal{H}'(t-1). \)

From I\((t-1)\) and III\((t-1),\) we have the following convergence result given \( \mathcal{H}'(t-1):\)

\[
(q_i(t-1), X_{ik}(t-1), \mu_{ij}(t-1)) \xrightarrow{L} N(q(t-1), x_k(t-1), u_k(t-1)), \forall l \in \mathcal{L},
\]

to which by applying the continuous mapping theorem and the bounded convergence theorem [56], we obtain

\[
\lim_{N \to \infty} \mathbb{E}\left\{ \frac{e^{y_{ik}(t)}}{e^{y_{ik}(t)} + 1} \mid \mathcal{H}'(t-1) \right\} = \frac{e^{y_{ik}(t)}}{e^{y_{ik}(t)} + 1}.
\]
\[
\lim_{N \to \infty} \mathbb{E} \left\{ \frac{e^{y_k(t)} (t-1)}{e^{y_k(t)} + 1} \mu_{ij}(t-1)|\mathcal{H}'(t-1) \right\} \\
= \frac{e^{y_k(t)}}{e^{y_k(t)} + 1} \mathbb{E}\{u_k(t-1)|\mathcal{H}'(t-1)\}, \ \forall t \in \mathcal{N}.
\] (6.10.21)

By taking the limit of \(N\) over both sides of (6.10.19) and applying (6.10.20)(6.10.21), we conclude

\[
\lim_{N \to \infty} \text{Pr}\{\mu_{ij}(t) = 1|\mathcal{H}'(t)\} = \lim_{N \to \infty} \mathbb{E}\{\mu_{ij}(t)|\mathcal{H}'(t)\} \\
= \beta \frac{e^{y_k(t)}}{e^{y_k(t)} + 1} \\
+ (1 - L\beta) \frac{e^{y_k(t)}}{e^{y_k(t)} + 1} \mathbb{E}\{u_k(t-1)|\mathcal{H}'(t-1)\} \\
= \mathbb{E}\{u_k(t)|\mathcal{H}'(t)\} = \text{Pr}\{u_k(t) = 1|\mathcal{H}'(t)\},
\]

where the third equality follows from (6.10.14). Hence, we have completed the proof of II(\(t\)).

**Proof of IV(\(t\))**

We prove IV(\(t\)) by enumerating all four cases of \(w_1, w_2\) in (6.10.9). Specifically, proving (6.10.9) is equivalent to proving, \(\forall i_1, i_2 \in \mathcal{L}, \forall j_1 \neq j_2\) with \(j_1 \in \mathcal{I}_{k_1}\) and \(j_2 \in \mathcal{I}_{k_2}\), \(\forall g_1, g_2 \in \{g^{(0)}, g^{(1)}\}\),

\[
\lim_{N \to \infty} \mathbb{E}\{g_1(\mu_{i_1j_1}(t))g_2(\mu_{i_2j_2}(t))\} \\
= \mathbb{E}\{g_1(u_{k_1}(t))\} \mathbb{E}\{g_2(u_{k_2}(t))\},
\] (6.10.22)

where \(g^{(0)}, g^{(1)}: \{0, 1\} \to \{0, 1\}\) are defined as \(g^{(1)}(x) = x\) and \(g^{(0)}(x) = 1 - x, \forall x \in \{0, 1\}\). The four cases are listed as follows:

- **Case 1:** \(w_1 = w_2 = 1\) in (6.10.9) corresponds to \(g_1 = g_2 = g^{(1)}\) in (6.10.22);
- **Case 2:** \(w_1 = 1, w_2 = 0\) in (6.10.9) corresponds to \(g_1 = g^{(1)}, g_2 = g^{(0)}\) in (6.10.22);
- **Case 3:** \(w_1 = 0, w_2 = 1\) in (6.10.9) corresponds to \(g_1 = g^{(0)}, g_2 = g^{(1)}\) in (6.10.22);
Case 4: \( w_1 = w_2 = 0 \) in (6.10.9) corresponds to \( g_1 = g_2 = g^{(0)} \) in (6.10.22).

We note that \( a_{ij}(t) \) are independent over \( i, j \) and \( p_{ij}(t) \) are independent over \( j \) by definition. From (6.10.5), for any given \( i_1, i_2 \in \mathcal{L} \) and \( j_1 \neq j_2 \), \( \mu_{i_1 j_1}(t) \) and \( \mu_{i_2 j_2}(t) \) are independent given

\[
X(t)
\]

\[
\Delta = (q_{i_1}(t - 1), q_{i_2}(t - 1), X_{i_1 k_1}(t - 1), X_{i_2 k_2}(t - 1),
\]

\[
(\mu_{ij}(t - 1))_{i \in L}^{j \neq j_1,j_2}, \mathcal{H}'(t)),
\]

where we note that \( S(t) \) is deterministic given \( \mathcal{H}(t) \). Thus, we have, \( \forall i_1, i_2 \in \mathcal{L}, \forall k_1, k_2, \forall j_1 \in \mathcal{L}_{k_1}, j_2 \in \mathcal{L}_{k_2} \) with \( j_1 \neq j_2 \),

\[
\mathbb{E}\{g_1(\mu_{i_1 j_1}(t))g_2(\mu_{i_2 j_2}(t))|X(t)\}
= \mathbb{E}\{g_1(\mu_{i_1 j_1}(t))|X(t)\}\mathbb{E}\{g_2(\mu_{i_2 j_2}(t))|X(t)\}
= [F_{g_{1,0}}(q_{i_1}(t - 1), X_{i_1 k_1}(t - 1), S_{k_1}(t))
+ \sum_{i_1 \neq i_1} F_{g_{1,2}}(q_{i_1}(t - 1), X_{i_1 k_1}(t - 1), S_{k_1}(t))\mu_{i_1 j_1}(t - 1)
+ F_{g_{1,1}}(q_{i_1}(t - 1), X_{i_1 k_1}(t - 1), S_{k_1}(t))\mu_{i_1 j_1}(t - 1)]
\times [F_{g_{2,0}}(q_{i_2}(t - 1), X_{i_2 k_2}(t - 1), S_{k_2}(t))
+ \sum_{i_2 \neq i_2} F_{g_{2,2}}(q_{i_2}(t - 1), X_{i_2 k_2}(t - 1), S_{k_2}(t))\mu_{i_2 j_2}(t - 1)
+ F_{g_{2,1}}(q_{i_2}(t - 1), X_{i_2 k_2}(t - 1), S_{k_2}(t))\mu_{i_2 j_2}(t - 1)],
\]

where we have used the following definitions

\[
F^{(1)}_{g_{1,0}}(z_1, z_2, S) \triangleq \beta \frac{e^{z_1 S - \gamma z_2 (1 - S)}}{e^{z_1 S - \gamma z_2 (1 - S)} + 1},
\]

\[
F^{(1)}_{g_{1,1}}(z_1, z_2, S) \triangleq (1 - \beta) \frac{e^{z_1 S - \gamma z_2 (1 - S)}}{e^{z_1 S - \gamma z_2 (1 - S)} + 1},
\]

\[
F^{(1)}_{g_{1,2}}(z_1, z_2, S) \triangleq -\beta \frac{e^{z_1 S - \gamma z_2 (1 - S)}}{e^{z_1 S - \gamma z_2 (1 - S)} + 1}.
\]

232
\[ F_{g_{0},0}(z_{1}, z_{2}, S) \equiv 1 - F_{g_{1},0}(z_{1}, z_{2}, S), \]
\[ F_{g_{0},1}(z_{1}, z_{2}, S) \equiv -F_{g_{1},1}(z_{1}, z_{2}, S), \]
\[ F_{g_{0},2}(z_{1}, z_{2}, S) \equiv -F_{g_{1},2}(z_{1}, z_{2}, S). \]

By taking the limit of \( N \) and the expectation on both sides of (6.10.23), we obtain (6.10.24) after rearranging terms and employing the induction hypothesis (II(\( t - 1 \)), II(\( t - 1 \)), and IV(\( t - 1 \)). Note the last equality in (6.10.24) follows (6.10.14). Since \( g_{1}, g_{2} \) are arbitrarily chosen from \( \{g^{(0)}, g^{(1)}\} \), we have (6.10.9) holds \( \forall w_{1}, w_{2} \in \{0, 1\} \), completing the proof of IV(\( t \)).
Proof of III(t)

For notational simplicity, all expectations that appear in the following proof are conditioned on \( H'(t) \). For any \( i \in \mathcal{L} \), taking the variance of \( \frac{1}{n_k N} \sum_{j \in I_k} \mu_{ij}(t) \) leads to

\[
\text{Var} \left\{ \frac{1}{n_k N} \sum_{j \in I_k} \mu_{ij}(t) \right\} = (n_k N)^{-2} \left[ \sum_{j_1, j_2 \in I_k : j_1 \neq j_2} \text{Cov} \{ \mu_{ij_1}(t), \mu_{ij_2}(t) \} \right.
\]
\[\left. + \sum_{j \in I_k} \text{Var} \{ \mu_{ij}(t) \} \right] \tag{6.10.25}
\]

\[= \frac{(a)n_k N - 1}{n_k N} \text{Cov} \{ \mu_{ij_1}(t), \mu_{ij_2}(t) \} + (n_k N)^{-1} \text{Var} \{ \mu_{ij_1}(t) \} \]
\[ \xrightarrow{(b)} 0, \text{ as } N \to \infty. \]

where \( j'_1, j'_2 \in I_k \) are any two given channels with \( j'_1 \neq j'_2 \). The equality \( (a) \) in (6.10.25) follows from the exchangeability of \( (\mu_{ij}(t))_{j \in I_k} \), and \( (b) \) in (6.10.25) follows the property of asymptotic mutual independence by IV(t) which has been proved in Section 6.10.5.

By employing Chebyshev’s inequality to the variance \( \text{Var} \{ \frac{1}{n_k N} \sum_{j \in I_k} \mu_{ij}(t) \} \), we obtain, for any arbitrarily small \( \eta > 0 \),

\[
\lim_{N \to \infty} \text{Pr} \left\{ \left| \frac{1}{n_k N} \sum_{j \in I_k} \mu_{ij}(t) - \mathbb{E} \left\{ \frac{1}{n_k N} \sum_{j \in I_k} \mu_{ij}(t) \right\} \right| \geq \eta \right\} = 0.
\]

Hence, we conclude

\[
\frac{1}{n_k N} \sum_{j \in I_k} \mu_{ij}(t) - \mathbb{E} \{ \mu_{ij_1}(t) \}
\]
\[= \frac{1}{n_k N} \sum_{j \in I_k} \mu_{ij}(t) - \mathbb{E} \{ \frac{1}{n_k N} \sum_{j \in I_k} \mu_{ij}(t) \} \xrightarrow{P} 0, \tag{6.10.26}
\]

234
where the first equality follows from the exchangeability of \((\mu_{ij}(t))_{j \in I_k}\). By II\((t)\) (which has been proved in Section 6.10.5), III\((t)\) follows (6.10.26).

### 6.10.6 Proof of Lemma 6.10.5

We introduce the following lemma to assist the proof of Lemma 6.10.5.

**Lemma 6.10.8.** For any stabilizable arrival rate \(\lambda' \in \Lambda\), there exists a randomized stationary algorithm STAT with schedules \((\mu_{ij}^{\text{STAT}}(t))\) dependent only on \(\mathcal{H}(t)\), such that for each time slot \(t\),

\[
\frac{K}{N} \mathbb{E} \left\{ \sum_{k=1}^{M} \sum_{j \in I_k} \mu_{ij}^{\text{STAT}}(t)(1 - C_k(t)) \right\} = \lambda',
\]

\[
\frac{1}{n_k N} \mathbb{E} \left\{ \sum_{j \in I_k} \mu_{ij}^{\text{STAT}}(t)C_k(t) \right\} \leq \rho_k, \forall i \in \mathcal{L}, \forall k.
\]

(6.10.27)

Similar formulation and proof have been provided in [102, 71], so the proof for Lemma 6.10.8 is omitted for brevity.

Consider any given \(\lambda' \in \Lambda\). We define random variables \(\mu_k^{\text{STAT}}(t)\) as follows:

\[
\mu_k^{\text{STAT}}(t)|(\mathcal{H}(t) = h) = \begin{cases} 
1, \text{ w.p. } \frac{1}{L} \sum_{i \in \mathcal{L}} \frac{1}{n_k N} \mathbb{E} \left\{ \sum_{j \in I_k} \mu_{ij}^{\text{STAT}}(t)|(\mathcal{H}(t) = h) \right\}, \\
0, \text{ otherwise,}
\end{cases}
\]

(6.10.28)

\(\forall k\), for all possible \(\mathcal{H}(t) = h\). Since \((\mu_{ij}^{\text{STAT}}(t))\) are feasible schedules, given any \(\mathcal{H}(t) = h\),

\[
\sum_{i \in \mathcal{L}} \mu_{ij}^{\text{STAT}}(t) \leq 1, \forall j,
\]

235
and hence,
\[
\mathbb{E}\{\mu_k^{\text{STAT}}(t)|\mathcal{H}(t) = h\} = \frac{1}{n_kN} \sum_{j \in \mathcal{I}_k} \frac{1}{L} \sum_{i \in \mathcal{I}} \mathbb{E}\{\mu_{ij}^{\text{STAT}}(t)|\mathcal{H}(t) = h\} \leq \frac{1}{L}.
\]

According to definition (6.10.28), we also have the following:
\[
\sum_{k=1}^{M} \mathbb{E}\{n_k K \mu_k^{\text{STAT}}(t) S_k(t)|\mathcal{H}(t) = h\} = \frac{K}{NL} \sum_{i \in \mathcal{L}} \mathbb{E}\{\sum_{k=1}^{M} \sum_{j \in \mathcal{I}_k} \mu_{ij}^{\text{STAT}}(t)(1 - C_k(t))|\mathcal{H}(t) = h\},
\]
\[
\mathbb{E}\{\mu_k^{\text{STAT}}(t)(1 - S_k(t))|\mathcal{H}(t) = h\} = \frac{1}{L} \sum_{i \in \mathcal{L}} \frac{1}{n_kN} \mathbb{E}\{\sum_{j \in \mathcal{I}_k} \mu_{ij}^{\text{STAT}}(t)C_k(t)|\mathcal{H} = h\},
\]
from which by employing Lemma 6.10.8, we conclude:
\[
\sum_{k=1}^{M} \mathbb{E}\{n_k K \mu_k^{\text{STAT}}(t) S_k(t)\} = \lambda',
\]
\[
\mathbb{E}\{\mu_k^{\text{STAT}}(t)(1 - S_k(t))\} \leq \rho_k,
\]
proving Lemma 6.10.5.

### 6.10.7 Proof of Theorem 6.10.6

For notational simplicity, we define
\[
\Delta(t) \triangleq \mathbb{E}\left\{ \frac{1}{2}[q(t)^2 - q(t-1)^2] + \frac{\gamma K}{2} \sum_{k=1}^{M} n_k[x_k(t)^2 - x_k(t-1)^2] \right\}.
\]

236
By squaring both sides of the queue dynamics (6.10.10)(6.10.11), we have

\[ \Delta(t) \leq B_1 + \lambda \mathbb{E}\{ q(t) \} - \gamma K \sum_{k=1}^{M} n_k \rho_k \mathbb{E}\{ x_k(t) \} \]

\[ - \mathbb{E}\{ \sum_{k=1}^{M} u_k(t) \} n_k K (1 - C_k(t)) q(t - 1) \]

\[ - \gamma n_k K C_k(t - 1) x_k(t - 1) \] \[ \leq B_1 + K \max\{ K, \gamma \} \]

\[ + \lambda \mathbb{E}\{ q(t - 1) \} - \gamma K \sum_{k=1}^{M} n_k \rho_k \mathbb{E}\{ x_k(t - 1) \} \]

\[ - \mathbb{E}\{ K \sum_{k=1}^{M} n_k u_k(t) \} (1 - C_k(t)) q(t - 2) \]

\[ - \gamma C_k(t - 1) x_k(t - 2) \] \[ \equiv B_1 + K \max\{ K, \gamma \} + \lambda \mathbb{E}\{ q(t - 1) \} \]

\[ - \gamma K \sum_{k=1}^{M} n_k \rho_k \mathbb{E}\{ x(t) \} - \mathbb{E}\{ K \sum_{k=1}^{M} n_k u_k(t - 1)y_k(t - 1) \}, \]

where \( B_1 \triangleq \frac{1}{2} \lambda^2 + \frac{1}{2} K^2 + \frac{1}{2} \gamma K + \frac{1}{2} \gamma K \sum_{k} \rho_k^2 n_k \). Note that (c) follows from the following equality

\[ \mathbb{E}\{ u_k(t - 1) \} (1 - C_k(t - 1)) q(t - 2) \]

\[ - \gamma C_k(t - 1) x_k(t - 2) \] \[ \mathcal{H}'(t - 1) \]

\[ = \mathbb{E}\{ u_k(t - 1) \} \mathcal{H}'(t - 1) y_k(t - 1), \]

where we utilized the fact that \( q(t - 2), x_k(t - 2), \) and \( u_k(t - 1) \) are independent of \( C_k(t - 1) \) given \( \mathcal{H}'(t - 1) \) by their dynamics (6.10.10)(6.10.11)(6.10.12).

Since \( \lambda \in \left( \frac{L - 1}{L} \right)^{L - 1} \Lambda \), there exists \( \epsilon_1 > 0 \) such that \( \lambda' \triangleq \frac{\lambda}{\beta L} + \epsilon_1 \in \Lambda \) by definition.
We define \( \delta \) in Lemma 6.10.4 as follows:

\[
0 < \delta \triangleq \frac{\beta L \epsilon_1}{2(\epsilon_1 + \frac{\delta}{L})} < 1.
\]

By Lemma 6.10.4, we have

\[
\mathbb{E}\{u_k(t-1)y_k(t-1) | y_k(t-1) \geq B_2\} \\
\geq (\beta - \frac{\delta}{L})\mathbb{E}\{y_k(t-1) | y_k(t-1) \geq B_2\}.
\]

Employing the above inequality to (6.10.29), we obtain

\[
\Delta(t) \\
\leq B_1 + K \max\{K, \gamma\} + \lambda \mathbb{E}\{q(t-1)\} \\
- \gamma K \sum_{k=1}^{M} n_k \rho_k \mathbb{E}\{x(t-1)\} \\
- K \sum_{k=1}^{M} n_k |\Pr\{y_k(t-1) \geq B_2\} \\
\times \mathbb{E}\{u_k(t-1)y_k(t-1) | y_k(t-1) \geq B_2\} \\
+ \Pr\{y_k(t-1) < B_2\} \\
\times \mathbb{E}\{u_k(t-1)y_k(t-1) | y_k(t-1) < B_2\}\}
\]

(6.10.30)

\[
\leq B_1 + K \max\{K, \gamma\} + \lambda \mathbb{E}\{q(t-1)\} \\
- \gamma K \sum_{k=1}^{M} n_k \rho_k \mathbb{E}\{x_k(t-1)\} \\
+ KB_2(\beta - \frac{\delta}{L}) - K(\beta - \frac{\delta}{L}) \sum_{k=1}^{M} n_k \mathbb{E}\{y_k(t-1)\}.
\]

Since \( \lambda' = \frac{\lambda}{\beta L} + \epsilon_1 \in \Lambda \), by Lemma 6.10.5, there exists \( (\mu_k^{STAT}(t)) \) such that (6.10.15) (6.10.16) (6.10.17) hold for each time slot \( t \) for this \( \lambda' \). According to (6.10.17), we have

\[
-\mathbb{E}\{\sum_{k=1}^{M} n_k y_k(t-1)\} \leq -\mathbb{E}\{L \sum_{k=1}^{M} n_k \mu_k^{STAT}(t-1) y_k(t-1)\}.
\]

(6.10.31)
By applying (6.10.31) to (6.10.30) and employing (6.10.15)(6.10.16), we obtain

\[ \Delta(t) \leq B_3 + \lambda \mathbb{E}\{q(t-1)\} - \gamma K \sum_{k=1}^{M} n_k \rho_k \mathbb{E}\{x_k(t-1)\} \]

\[ - K(\beta L - \delta) \mathbb{E}\{\sum_{k=1}^{M} n_k \mu_k^{SAT}(t-1)\} \]

\[ \times [S_k(t-1)q(t-1) - \gamma(1 - S_k(t-1))x_k(t-1)] \]

\[ = B_3 - \mathbb{E}\{q(t-1)\} \]

\[ \times [(\beta L - \delta) \sum_{k=1}^{M} n_k K \mu_k^{SAT}(t-1)S_k(t-1) - \lambda] \]

\[ - \gamma K \sum_{k=1}^{M} n_k \mathbb{E}\{x_k(t-1)\} \]

\[ \times [\rho_k - (\beta L - \delta)(1 - S_k(t-1))\mu_k^{SAT}(t-1)] \]

\[ \leq B_3 - \epsilon_2 \mathbb{E}\{q(t-1)\} + \sum_{k=1}^{M} x_k(t-1), \]

(6.10.32)

where \( B_3 \triangleq B_1 + K \max\{K, \gamma\} + KB_2(\beta - \frac{\delta}{L}) + K(\beta L - \delta) \sum_{k=1}^{M} n_k \max\{\lambda, \gamma \rho_k\} \), and

\[ \epsilon_2 \triangleq \min \left\{ \frac{1}{2} \epsilon_1 \beta L, \min_k \gamma K \rho_k (1 - \beta L + \delta) n_k \right\} > 0. \]

From (6.10.32), by taking the time-average over \( t = 0, 1, ..., T - 1 \) and taking limsup of \( T \), we can prove (6.10.18), completing the proof of Theorem 6.10.6.
CHAPTER 7
CONCLUSIONS

In this dissertation, we have proposed several algorithms/techniques to improve/study the queuing delay performance of scheduling in wireless networks. Via a product form of the virtual and actual queue backlogs, a centralized algorithm and a distributed random access solution have been proposed to improve delay performance and ensure finite buffer sizes in a cross-layer framework. A novel type of Lyapunov function is employed which multiplies the virtual queue backlog to the quadratic term of actual queue backlogs to prove the throughput/utility optimality of the algorithms. A power control component is proposed based on the centralized algorithm to optimize power allocation with finite buffers. A $v(t)$-regulated CSMA algorithm is proposed that significantly reduces the queuing delays and mitigates the temporal starvation experienced under regular throughput-optimal CSMA algorithms. Furthermore, in a fully-connected OFDM setting, a closed-form steady-state queue backlog is approximated via an equivalent queuing system that brings to light the congestion level of throughput-optimal CSMA algorithms under a many-channel regime. Via a similar equivalent queuing system analysis, a suboptimal SU scheduling algorithm is proposed for cognitive radio network that achieves at least a constant fraction of the capacity region.

7.1 Future Work

In Chapter 6 and our works [107][114], with a fully-connected topology, we have generalized regular throughput-optimal CSMA algorithms from a single-channel setting [83, 85, 28]
to an OFDM setting and analyzed the queuing behavior of these algorithms under the many-channel regime (where the number of channels grows asymptotically given a constant number of users/communication links). Looking forward, realtime wireless applications can be more general with, e.g., (i) CSMA algorithms (e.g., our proposed algorithm [117] with the employment of virtual queues) other than [83, 85, 28], (ii) more general interference models (e.g., multiple-cell topology, general interference graph model), (iii) the number of communication links scaling with the number of channels (e.g., a macro base station where there are hundreds of users [25]). As a guideline for future work, we extend the queuing behavior/delay analysis to the above scenarios in more detail and tentatively propose the technical approaches to solve the problems.

7.1.1 Queueing Behavior of CSMA Algorithm Employing Virtual Queues

The study of the queueing behavior of the CSMA algorithms (without utilizing a virtual queue concept, e.g., [83, 28, 48]) has been carried out in the many channel regime in [107]. One future research direction can be the asymptotic analysis of queuing behavior under our proposed CSMA algorithm proposed in [117], which has been shown to achieve comparatively superior delay performance with the employment of virtual queues in a single-channel setting.

Problem Description

In [117], we assume that the sources of communications links are constantly backlogged. Thus, a congestion controller is needed to admit $A_i(t)$ packets into the network at time slot $t$. In addition, our proposed algorithm utilizes a novel virtual queue for each link $i$ with $A_i(t)$ being the service rate and $R_i(t)$ the virtual input rate to the virtual queue $Q_i(t)$, where $R_i(t)$ is a control parameter used to regulate virtual queue evolutions. A congestion controller for $A_i(t)$ and a regulator for $R_i(t)$ have been introduced in detail in [117]. In addition, different from the class of CSMA algorithms studied in [107], the transmission weight $w_i(t)$ in [117] is a function of the product of the actual queue and the virtual queue.
The problem is to find the asymptotic queue length expectation (or the lower/upper-bound for queue length expectation) for $q_i(t)$ and $Q_i(t)$ under the proposed virtual-queue-based CSMA algorithm [117] under the many-channel regime. Note that actual packet queue lengths $q_i(t)$ correspond to the queuing delays, while the virtual queue lengths $Q_i(t)$ correspond to the convergence rate and stability of the system [117].

Challenges

The challenges for analyzing delay performance of the virtual-queue-based CSMA algorithm under the many-channel regime are two-fold. The first challenge is to establish an equivalent queuing system. Theorem 6.3.1 states the convergence results of the individual packet queues to a deterministic single-queue system. With the addition of a virtual queue maintained at the source of each communication link, we have to establish a two-queue system and the system’s evolution rule such that the original queuing system under the virtual-queue-based algorithm converges to this two-queue system asymptotically with the number of channels. In addition, with the addition of a virtual rate regulator and a congestion controller, we also have to establish Law-of-Large-Number results for the admitted rate of both virtual and actual queues.

The second challenge is to establish the asymptotic stability (over time) of the equivalent two-queue system. Note that the asymptotic stability of the single-queue system in [107] is proved via analyzing a two-by-two Jacobian matrix. For a two-queue system, the Jacobian matrix is five-by-five, adding to the difficulty of establishing stability.

Possible Technical Approaches

In conformance with the study in [107], we expect the system in the many-channel regime to converge to a simple two-queue virtual system $(q(t), u(t), s(t), v(t), r(t))$, according to the following: (i) Convergence results for the queue backlogs hold for $(q(t), v(t))$. (ii) The virtual queue backlogs converge to the deterministic queue backlog $u(t)$ in probability.
(iii) The normalized admitted arrival rates $A_i(t)$ and virtual input rates $R_i(t)$ converge in probability to the deterministic $s(t)$ and the deterministic $r(t)$, respectively:

$$A_i(t) \xrightarrow{P} N s(t), \text{ and } R_i(t) \xrightarrow{P} N r(t), \forall i.$$ 

The next step will be the derivation of simple updates of the virtual system

$$(q(t), u(t), s(t), v(t), r(t))$$

and the analysis of the corresponding steady state queue backlogs. Note that the asymptotic stability of the single-queue system in [107] is proved via analyzing a two-by-two Jacobian matrix. For the above two-queue non-linear system, the Jacobian matrix is five-by-five, adding to the difficulty of establishing stability. An alternative engineering solution to analyze the stability of the two-queue non-linear system can be done by drawing the nullclines and direction of the velocity vectors for the non-linear system via simulation. By this way, we can study numerically the stability/non-stability region for the equivalent system. When this is done, we can compare the delay performance both theoretically and numerically with respect to the CSMA algorithms not employing virtual queues (e.g., Q-CSMA [83]) under the many-channel regime.

### 7.1.2 Queueing Behavior of CSMA Algorithms with More General Interference Models

#### Problem Description

In [107], we considered an interference model based on a complete (interference) graph. A typical corresponding scenario is: A single cell of an uplink OFDM system consisting of orthogonal channels and users with a single base station as the common destination, as illustrated in Figure 7.1. Two users cannot be scheduled to transmit over a same channel in the model.

In this section, we discuss: (a) whether similar asymptotic analysis as in [107] holds
when the network topology is more general, and (b) how to prove (asymptotic) throughput optimality under modified CSMA algorithms under the many-fading channel regime. In the following, we discuss the challenges and propose possible technical approaches for the following interference models: (a) Degree-$d$ interference graph; (b) Multiple-cell topology; (c) General interference graph.

**Challenges**

For analytical simplicity, we first introduce the definition of a general interference graph. Consider a wireless network composed of a set $\mathcal{L}$ of single-hop directional transmission links. In an interference graph, we define an interference set $\mathcal{N}_i \subset \mathcal{L}$ for each link $i \in \mathcal{L}$, such that the transmission of SU link $i$ fails if and only if there is a simultaneous transmission over some link $l \in \mathcal{N}_i$.

Note that the complete graph model introduced in [107] is a special case of the interference graph model. Specifically, each user $i$ is equivalent to a link in the link set $\mathcal{L}$, with $\mathcal{L} = \{1, 2, ..., M\}$. Under the complete graph, we have the following symmetry property over links (users):

$$\mathcal{N}_i \cup \{i\} = \mathcal{L}, \forall i \in \mathcal{L}. \quad (7.1.1)$$
We recall that in [107], all packet queues converge (in the number of channels) to an equivalent queue. This convergence and the Law of Large Numbers (LLN) results in Theorem 2 and Theorem 3 in [107] rely on the property (7.1.1) under the complete graph model. In more general interference models where (7.1.1) no longer holds, the original wireless system may not converge to an equivalent single-queue system. Hence, subsequent results (i.e., the closed-form approximation on packet queue lengths and the asymptotic throughput optimality under the many-fading-channel regime) in [107] need a modified or different approach to obtain.

In the following sections, we propose possible technical approaches for more general interference models.

**Technical Approach for Degree-\(d\) Interference Graph Model**

We first consider a degree-\(d\) interference graph model. We say an interference graph is a degree-\(d\) interference graph model if

\[
|\mathcal{N}_i| = d, \forall i \in \mathcal{L}. \tag{7.1.2}
\]

Typical topologies of the degree-\(d\) interference graph model include complete graph, cycle (or infinite tandem), torus (or infinite grid), etc. Some exemplary topologies have been illustrated in Figure 7.2 under a node-exclusive model, where an SU \(i \in \mathcal{L}\) is represented by a link (a node pair) in the graph. Note that in a node-exclusive model, adjacent links (links sharing a same node) cannot transmit simultaneously.

Under the degree-\(d\) interference graph model, since the symmetry property (7.1.2) holds, we can still expect an asymptotic convergence (in number of channels) to an equivalent single-queue system, given the CSMA algorithm in [107] with contention probability \(\beta = \frac{1}{d+1}\). The closed-form approximation on packet queue backlogs and asymptotic throughput optimality follow similar analysis in [107] on the stability of the equivalent queue system.
Technical Approach for Multiple-Cell Topology

In this section, we consider a general cellular network represented by a multiple-cell topology, which is a generalization of the complete graph (i.e., a single-cell) model in [107].

As an exemplary illustration, we consider a two-cell topology in Figure 7.3, with two cells consisting of two non-overlapping areas $A$ and $B$ and an overlapping area $C$. Specifically, users from area $C$ cannot transmit over a same channel with users from either area $A$ or are $B$, while users from area $A$ can transmit simultaneously with users from area $B$ over a same channel. We assume both cells have a same set of $N$ channels. With $M_1$ denoting the set of users in area $A$, $M_2$ the set users in area $B$, and $M_3$ the set users in area $C$, we can find three formula for the dynamics of $(\mu_{ij}(t))_{j=1,...,N}$ for $i \in M_1$, $i \in M_2$, and $i \in M_3$, respectively, given the previous system state $((\mu_{ij}(t-1))_{i,j}, (q_i(t-1))_i$ and given the CSMA algorithm in [107] with modifications on contention probability $\beta$. Thus, we can expect the two-cell system converges to an equivalent three-queue system with deterministic updates as the number of channel becomes large, by extending the analysis in [107]. By studying the stability of the three-queue system, we can analyze the packet queue lengths and asymptotic throughput optimality of the original system under the many-fading-channel regime.

Figure 7.2: Sample topologies of degree-$d$ interference graph under a node-exclusive model
Technical Approach for General Interference Graph Model

We now consider the general interference graph model. We can modify the CSMA algorithm in [107] with contention probability $\beta = \frac{1}{|N_i| + 1}$ for each communication link $i \in \mathcal{L}$. Under the many-fading-channel regime, we can expect that the original system converges to an equivalent $|\mathcal{L}|$-queue system. That is, each packet queue $q_i(t)$ in the original system converges (in probability) to a (unique) equivalent queue, $i \in \mathcal{L}$. While a closed-form queue approximation is difficult to find for a general interference graph model, we can still analyze the throughput performance by applying a Lyapunov analysis to the equivalent system.

7.1.3 The Case of the Numbers of Users and Channels Scaling Together

Problem Description

In [107], we considered the wireless system composed of $M$ users/communication links when the number of fading channels $N$ grows asymptotically. That is, we assumed that the number of users does not scale with $N$. This assumption holds in real deployment such as WiMAX micro, pico and femto cells. However, this does not hold for macro base station where there are hundreds of users [25].

In this section, we consider the scenario where the number of users scales with the number of channels. The problem is to prove asymptotic throughput optimality and analyze
the packet queue lengths for the CSMA algorithm in [107] under this scenario. We discuss the challenges and the technical approach in the following.

Challenges

The convergence results in Theorem 6.3.1 is based on the schedule dynamics (6.2.2), which involves only finite summations when the number of users $M$ is finite. When $M$ scales with $N$, Theorem 6.3.1 does not hold since the schedule dynamics (6.2.2) now has an infinite summation. Thus, subsequent analysis on the closed-form queue length approximation and the asymptotic throughput optimality does not hold, which calls for a different technical approach to study the problem.

Technical Approaches

Since the convergence analysis in [107][114] no longer holds when $M$ scales with $N$, we consider a different approach to analyze the packet queue lengths. We plan to use the large deviation techniques, which have been applied to centralized wireless systems in [8, 9, 93]. Specifically, we let $Q(t) = \max \{q_i(t)\}$ be the maximum queue length over users at time slot $t > -\infty$. We are interested in the following metric, the rate function for buffer $D$,

$$I(D) = \lim_{N \to \infty} -\frac{1}{N} \log \mathbb{P}\{Q(0) > D\}, \quad (7.1.3)$$

where we can consider $\{Q(0) > D\}$ as the buffer overflow event. That is, with a large number of channels $N$, we can approximate $\mathbb{P}\{Q(0) > D\} \approx e^{-NI(D)}$. Note that it has been shown in [8] that the well-known centralized throughput-optimal MaxWeight algorithm [100] has a zero rate function, i.e., the queue-overflow performance is poor.

When $M$ scales with $N$, via a large deviation technique, we plan to analyze the rate function $I(D)$ for a set of transmission weight functions $h$. In addition, we plan to study under which weight functions the algorithm has positive rate function for arrival rates lying within the asymptotic capacity region. Note that with a positive rate function, the overflow probability decreases exponentially under the many-fading-channel regime.
It has been shown in [9] under i.i.d. arrival and channel processes: (i) The stationary distribution of $Q(t)$ is given by a stationary distribution of a one-dimensional Markov chain; (ii) If the transition probability of queue length increase in $Q(t)$ decays exponentially with the increase, then the algorithm has a positive rate function $I(D)$. Thus, the analysis of the rate function performance of the algorithm boils down to the study of the transition probabilities of a one-dimensional Markov chain.
BIBLIOGRAPHY


257
[101] Tsai, Y., and Zhang, G. Time and frequency synchronization for 3gpp long

[102] Urgaonkar, R., and Neely, M. Opportunistic scheduling with reliability
guarantees in cognitive radio networks. IEEE Transactions on Mobile Com-
puting 8, 6 (June 2009).

[103] Urgaonkar, R., and Neely, M. Opportunistic cooperation in cognitive
femtocell networks. IEEE Journal on Selected Areas in Communications 30, 3
(April 2012).

scheduling for minimizing end-to-end buffer usage over multihop wireless net-

greedy scheduling algorithms in wireless networks. IEEE Transactions on
Mobile Computing 6, 6 (June 2007).

[106] Xiong, H., Li, R., Eryilmaz, A., and Ekici, E. Delay-aware cross-layer
design for network utility maximization in multi-hop networks. IEEE Journal
on Selected Areas in Communications (2011).

[107] Xue, D., and Ekici, E. Queueing behavior of queue-backlog-based csma
scheduling under the many-channel regime. In WiOpt 2013.

[108] Xue, D., and Ekici, E. Guaranteed opportunistic scheduling in multi-hop

[109] Xue, D., and Ekici, E. Optimal power allocation in multi-hop wireless
networks with finite buffers. In IEEE International Conference on Communi-
cations (ICC) (June 2011).

[110] Xue, D., and Ekici, E. Delay-guaranteed cross-layer scheduling in multi-hop

[111] Xue, D., and Ekici, E. On reducing delay and temporal starvation of queue-
length-based csma algorithms. In 50th Annual Allerton Conference on Com-
munication, Control, and Computing (October 2012).

[112] Xue, D., and Ekici, E. Power optimal control in multi-hop wireless networks

[113] Xue, D., and Ekici, E. Cross-layer scheduling for cooperative multi-hop
cognitive radio networks. IEEE Journal on Selected Areas in Communications
31, 3 (2013).


