THE DELAY PERFORMANCE OF ADAPTIVE ROUTING AND SCHEDULING IN COMMUNICATION NETWORKS

BY

TIANXIONG JI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

ABSTRACT

Throughput and latency are two important QoS metrics in communication networks. Ideally, we would like to deliver a large amount of data from a source to its destination within a short time period. During the past decades, researchers have designed a number of network-layer routing algorithms and MAC-layer scheduling algorithms to deliver good QoS performance under various network conditions.

In this dissertation, we study the delay performance of routing and scheduling algorithms in communication networks. A collection of algorithms called MaxWeight Scheduling (MWS) algorithms is known to be throughput optimal. A particular algorithm in this class was conjectured to be delay-optimal as well. We disprove this conjectured by constructing a delay-optimal algorithm for a specific network and show that it outperforms the particular MWS algorithm.

Next, we propose a packet-by-packet adaptive routing and scheduling algorithm for multi-hop traffic which dramatically reduces delays in the network, while maintaining near throughput optimality. This algorithm uses a particular routing table, called a probabilistic routing table, to adaptively find a route for every packet. The probabilistic routing table is generated by running an emulated network, called the shadow network, which is essentially a model of the real network with a slightly higher traffic load. The scheduling decisions are also determined by the shadow system. Our joint routing and scheduling algorithm reduces the capacity region slightly, but provides low delay performance everywhere in this reduced region. In addition, the queueing and routing architecture is more consistent with the architecture of current routers and switches. We also extend our results to the case where network coding is used to improved the throughput in the network. Our algorithm provides a low-complexity solution to optimally exploit the routing-coding tradeoff.

Lastly, we consider wireless ad hoc networks in which each connection (file) traverses only one hop. We assume files arrive for service at each link and a file departs when all its packets have been transmitted. We consider two cases: one in which the file size distribution is arbitrary but has bounded support; another in which the file size distribution is a mixture of geometric distributions. The only assumption we make about the window flow control protocol is that the window size is always greater than zero. We show the following result: for an appropriately chosen MAC-layer scheduling algorithm, the network is stable for all file arrival rates within the capacity region. In other words, the MAC-layer scheduling algorithm, by only knowing the MAC-layer information, can achieve throughput optimality independently of the window flow control protocol used.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my adviser, Prof. Srikant, for his exceptional guidance and committed support during my journey of pursuing my Ph.D. He provides me a wonderful research atmosphere and an excellent experience in his laboratory. Without his guidance, support, caring and patience, I would never have been able to finish my dissertation.

I would like to thank my other committee members, Prof. Başar, Prof. Coleman, Prof. Sreenivas, and Prof. Vaidya. They spent their precious time to attend my preliminary and final examinations, and provided me valuable opinions which guided me during the writing of my dissertation.

I appreciate the guidance from, and interaction with, my collaborators Dr. Athanasopoulou, Dr. Bui, Dr. Stoylar and Javad Ghaderi. I am inspired by our warm discussions about interesting problems, and we are working together towards the same objective.

I would like to thank my great labmates, Eleftheria Athanasopoulou, Loc Bui, Javad Ghaderi, Juan Jose Jaramillo, Chong Jiang, Joohwan Kim, Jian Ni, Siva Theja, Bo Tan, and Rui Wu for accompanying me during my journey towards the Ph.D.

I would also like to thank my dearest parents for their precious support and encouragement every day.

Finally, I would like to give my special thanks to my wife, Siying Liu. She gives me her strongest support, encourages me in my hardest times, and is cheered by my every success.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Throughput and latency are two main QoS metrics in communication networks. Since the seminal work of Tassiulas and Ephremids [1], MaxWeight-type scheduling algorithms as well as backpressure-type adaptive routing algorithms for communication networks have been extensively studied under various network scenarios. The traditional backpressure algorithm in [1] provides throughput optimality, but has very bad delay performance under multi-hop network scenarios. Moreover, the computational complexity of the backpressure algorithm or the MaxWeight algorithm (i.e., the backpressure algorithm under one-hop traffic case) is extremely high.

There are two essential problems facing researchers: First, how do we reduce the computational complexity while providing good throughput performance? Second, how do we design algorithms to have good delay performance? These two problems are more difficult in the multi-hop traffic scenario because routing should be deployed as well as scheduling. Several scheduling algorithms are proposed to solve the first problem in a one-hop traffic scenario [2, 3, 4, 5, 6]. Fully distributed CSMA algorithms have been developed in recent years to make the scheduling algorithm completely distributed, and the throughput optimality is maintained under some conditions [7, 8, 9]. In a multi-hop traffic scenario, multi-path routing algorithms have been proposed to utilize network resources as much as possible [10, 11, 12, 13, 14, 15]. However, throughput optimality is not guaranteed in these routing algorithms. Backpressure-based routing algorithms can guarantee throughput optimality, but these algorithms are unrealistic due to bad delay performance or unrealistic queueing structure [1, 4, 7, 16]. The delay performance of algorithms is not well understood, especially in the multi-hop traffic scenario. The well-known shortest path routing is an intuitive way to provide good delay performance in light traffic, but it will lead to instability at moderate to high loads. In [17], the authors heuristically modified the backpressure algorithm to bias it towards taking shortest-path routes. ECMP (Equal-Cost Multi-Path) routing algorithm distributes traffic load equally onto multiple shortest paths to provide higher network throughput. However, it is far from optimal because it does not consider network traffic pattern. The delay performance has been more extensively studied in networks with single-hop traffic only. It is shown that an appropriate function of the workload in the system is minimized in heavy traffic region under a condition called *complete resource pooling* (CRP) condtion if MWS-α algorithm is used [18]. Shah and Wischik [19] studied the problem without the CRP condition in a fluid model. Keslassy and McKeown used simulations to study the delay performance of MWS-α algorithms in [20].

The packet is usually the basic unit when researchers analyze the queueing system of the network and design their algorithms. However, the file (or flow) is a more meaningful unit when we consider the latency of a group of information-related packets. It is well known that a small number of large flows dominate most of the traffic in the Internet. Stability of wireless networks under connection-level dynamics has been studied in, e.g., [21], [22], [23], [24]. Analyzing delay performance of files is attracting more attention. Good packet delay performance does not necessarily indicate good file delay performance.

In this dissertation, we discuss three problems and study these two essential QoS issues under various scenarios.

- In Chapter 2, we propose optimal scheduling algorithms for some small generalized switches (including wireless networks and switches with one-hop traffic) where the traffic load is close to the capacity boundary. We study the sum of queue lengths in the network for each sample-path under our scheduling algorithm and compare our algorithm using simulations with MWS-α algorithm where α is very small. It has been conjectured that MWS-0 minimizes the total queue length in a generalized network. Our results indicate that this conjecture is false.

- In Chapter 3, we consider a backpressure-based adaptive routing and scheduling algorithm for communication networks. Traditional backpressure routing has very poor delay performance. We systematically develop a routing algorithm using the backpressure idea, which avoids long routes and unnecessary loops in the network, and gets much better delay performance. We also reduce the queue complexity, which is an important issue for the scalability of the network. The queueing structure under our algorithm is widely used in commercial switches and routers. The extension of the routing algorithm into a simple network coding scenario is also described in detail.

- In Chapter 4, we study the scheduling algorithm in the scenario where files arrive into and depart from a wireless network in which each file traverses only one hop. We assume that a window flow control protocol regulates the injection of packets from each file into a MAC layer queue. We design a scheduling algorithm, which only uses the MAC-layer queue information, to guarantee throughput optimality. We also consider file latency instead of packet latency as a metric of the QoS performance. Simulation results indicate a significant file delay improvement under our scheduling algorithm compared to the original MaxWeight algorithm.

Our contributions are summarized as follows:

1. Optimal Scheduling Algorithms for Small Generalized Switches

- For some small networks, we have proposed optimal scheduling algorithms which minimize the number of backlogged packets in the network for each sample path in an arbitrary traffic region or in a heavy traffic region.

- There is a well-known conjecture that MWS-α scheduling algorithms with α going to zero are heavy-traffic optimal for scheduling in a generalized switch when the objective is to minimize the number of backlogged packets in the system. Our scheduling algorithm outperforms MWS-α in a heavy traffic region; thus, it is a counter-example to this conjecture.

2. Backpressure-based Packet-by-Packet Adaptive Routing in Communication Networks

- Using a novel concept of shadow queues, we decouple routing and scheduling such that the routing decision will no longer affect the scheduling decision. A shadow network is used to emulate the optimal traffic distribution and to update a probabilistic routing table which packets use to decide the next hop upon arrival at a node. The backpressure-based scheduling algorithm for the shadow network is used to serve real FIFO queues over each link.

- The routing algorithm is designed to minimize the average number of hops used by packets in the network. This idea, along with scheduling/routing decoupling, leads to delay reduction compared with the traditional backpressure algorithm, and the delay performance under our routing algorithm is very close to that of the shortest path routing algorithm in light traffic.

- We also extend our adaptive routing and scheduling algorithm into the scenario where a simple network coding scheme is employed. By employing network coding, the throughput region is expanded, but the queueing structure has to be modified correspondingly.

- Each node has to maintain counters, called shadow queues, per destination. This is very similar to the idea of maintaining a routing table per destination. But the real queues at each node are per-next-hop queues. When network coding is employed, real packets should not only remember the destination, but also keep the knowledge where they come from. Therefore, per-previous-hop-per-destination queues may also be necessary, but this is a requirement imposed by network coding, not by our algorithm.

- Our algorithm can be applied to wireline and wireless networks. Extensive simulations show dramatic improvement in delay performance compared to the backpressure algorithm.

3. Connection-Level Scheduling in Wireless Networks Using Only MAC-Layer Information

- Using the total queue length as the weight in MWS algorithms causes short files to experience high latencies. Instead, we use the MAC-layer queues in our algorithm and show that such an algorithm is still throughput optimal and its overall delay performance is better than traditional MWS in simulations.

- In reality, scheduling is performed at the MAC-layer and the scheduler does not have access to the total queue length information. Hence, our algorithm design is based on a more practical assumption.

In the next three chapters, we elaborate the problems, the key ideas and the proofs for the above three problems. We conclude this dissertation in the last chapter.

CHAPTER 2

OPTIMAL SCHEDULING IN SMALL GENERALIZED SWITCHES

Scheduling algorithms for high-speed switches and wireless networks (together called *generalized switches* in [18]) have been widely studied since the seminal work of Tassiulas and Ephremides [1]. Much of the work has focused on variations of the MaxWeight algorithm proposed in [1], usually with the goal of achieving 100% throughput. To motivate the problem considered in this chapter, we briefly describe the MaxWeight algorithm: for single-hop traffic scenario, each link is associated with one queue; a link is assigned a weight equal to the α^{th} power of its queue length, where $\alpha > 0$. A schedule is a set of links that can be enabled simultaneously without interfering with each other. The MaxWeight algorithm then picks the schedule with the largest weight, where the weight of a schedule is defined to be the sum of the weights of the links included in the schedule. To emphasize the role of α, the corresponding MaxWeight algorithm is sometimes called MWS-α, where MWS stands for MaxWeight scheduling. The original MaxWeight algorithm in [1] considers only the special case of $\alpha = 1$, i.e., MWS-1, but extensions to the case of general α are straightforward; see, for example, the paper by Eryilmaz et al. [25].

2.1 Related Works

Given that MWS-α achieves 100% throughput for any value of α greater than zero, the natural question to ask is whether α plays any role in network performance. Keslassy and McKeown studied this problem using simulations [20] and concluded that the average number of packets in a switch decreases as α decreases. In [18] Stolyar analyzed a much more general model which includes both high-speed switches and wireless networks with fading as special cases. Stolyar showed that $\sum_l q_l^{1+\alpha}$, where q_l is the queue length of link l in a switch, is minimized in a sample-path sense in heavy traffic by MWS-α under a condition called the *Complete Resource Pooling* (CRP) condition. Under the CRP condition, only one resource in the network can be heavily loaded. Relaxing this condition has been a subject of considerable interest. Shah and Wischik [19] studied the problem without the CRP condition; they characterized the dimension of the workload space when the switch is critically loaded and they provided further evidence to indicate that $\sum_l q_l^{1+\alpha}$ is minimized in an appropriate sense by MWS-α. However, the notions of optimality used in [18] and [19] appear to be different. We will comment on various notions of optimality later in this chapter. Related to the above work is the large deviations work of Venkataramanan and Lin [26] who show that the MWS-α algorithm maximizes the asymptotic decay-rate of the probability that the $L_{1+\alpha}$-norm of the queue length vector exceeds some threshold B. In [27] line networks under the 1-hop interference model are considered and it is shown that an optimal algorithm schedules the maximum number of queues (subject to interference constraints) so that the maximum number of packets are served at each slot. Among this class of algorithms, a further subset that does at least as well as any other given algorithm is identified. In one case (which we will mention later), the optimal algorithm has been derived. In this chapter, we derive heavy-traffic optimal algorithms for small generalized switches with the goal of comparing them to MWS-α algorithms. An earlier version of this work appears in [28].

Heavy-traffic analysis of generalized switches can be traced to Brownian models of stochastic networks, suggested by Harrison in [29]. The Brownian formulation approximates the original problem by replacing all random processes (such as arrival processes and potential service processes) by Brownian motions with a drift. In some cases, the resulting stochastic control problem can be reduced to a lower-dimensional problem (a phenomenon referred to as *state-space collapse* by Reiman in [30]) which sometimes renders the problem analytically tractable. When the state collapses to a single dimension, it means that there is a single critical resource in the network which determines the performance of the network. It was shown by Laws in [31] that the problem of identifying the appropriate critical resource (or

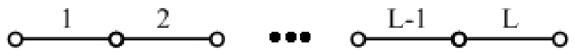

Figure 2.1: L-link line network.

resources), and the corresponding state-space collapse, can be formulated as a linear program. A number of examples illustrating the applicability of this result to scheduling and sequencing problems were provided by Kelly and Laws in [32]. Although the Brownian formulation suggests a solution for some network control problems, additional work is needed to show that the resulting solution is indeed near-optimal for the original model. Thus, a key contribution in [18] was to prove the heavy-traffic sample-path optimality of MWS-α algorithm. Related papers by Shakkottai et al. [33] and Bell and Williams [34] also use the CRP condition to establish heavy-traffic optimality of other resource allocation models. An introduction to fluid and Brownian models of stochastic networks can be found in the recent book by Meyn [35].

2.2 The Network Model

We consider networks consisting of L links arranged in a line as shown in Figure 2.1. Two adjacent nodes are connected by a link to denote that communication is possible between the pair. We assume that all traffic is single-hop, that time is slotted, and that packets are of equal size. To model interference in wireless networks, we consider two interference models: the 1-hop and 2-hop interference models. A valid schedule in a κ-hop interference model is a set of links such that no two links in the schedule are within κ hops of each other. Note that input-queued high-speed switches with special traffic patterns can also be described using the above model as shown in Figure 2.2. In Figure 2.2, input port A has traffic destined only to output port a, input port B has traffic destined for output ports a and b, and input port C has traffic destined for output ports b and c. It is easy to see that this switch can be

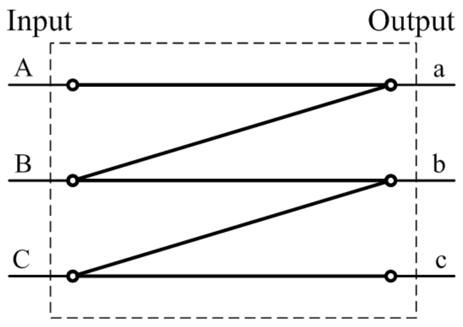

Figure 2.2: A 3×3 input-queued switch.

viewed as a line network under the 1-hop interference model.

Now we introduce some conventions and notation that will be used throughout the chapter. We assume that, during each time slot, arrivals occur first; then we observe the queue lengths to make the scheduling decision, and then departures occur. Let us associate a queue with each link and use $q_l[k]$ to denote the queue length of link l at the beginning of time slot k. Let $\alpha_l[k]$ denote the number of arrivals at link l at time slot k. We assume that the arrival processes to different links are mutually independent and stationary. The average arrival rate at link l is denoted by λ_l, i.e., $\lambda_l = E[\alpha_l[k]]$. We will make additional assumptions on the arrival processes in Section 2.5.

We say that link l or queue l is backlogged at time k if the queue length is greater than zero after the arrival epoch, i.e., if $q_l[k] + a_l[k] > 0$. If a backlogged link l is scheduled at time slot k, then one packet is removed from its queue. Let the indicator function $d_l[k]$ denote whether link l is scheduled at time slot k, i.e.,

$$d_l[k] := \begin{cases} 1, & \text{if link } l \text{ is scheduled} \\ 0, & \text{otherwise.} \end{cases}$$

Without loss of generality, we assume that each link has a unit link rate. Thus, $d_l[k]$ is the number of potential departures from link l at time slot k, i.e., the number of link l's packets that can be served at time slot k. The actual number of departures from link l can be less than $d_l[k]$ if link l is not backlogged. The dynamics of $q_l[k]$ are given by

$$q_l[k+1] \;=\; \big[q_l[k]+\alpha_l[k]-d_l[k]\big]^+,$$

where $\big[x\big]^+=\max(x,0)$. We define $u_l[k]$ to be the unused service on link l at time k, i.e.,

$$u_l[k]:=\begin{cases}1, & \text{if } q_l[k]+\alpha_l[k]=0 \text{ and } d_l[k]=1\\ 0, & \text{otherwise.}\end{cases}$$

Let λ_l be the arrival rate at link l and $\mathbf{\Lambda}:=(\lambda_1,\lambda_2,\cdots,\lambda_L)$. Denote each feasible schedule in the L-link network by an L-dimensional vector with the l^{th} element of the vector being 1 if link l is included in the schedule and 0 otherwise. The capacity region $\mathcal{C}$ is defined to be the convex hull of the feasible schedules. The set $\mathcal{C}$ is called the capacity region for the following reason: if we let ρ be the traffic load on the network, which is defined as $\rho=\inf\{\xi>0|\mathbf{\Lambda}/\xi\in\mathcal{C}\}$, then, for any set of arrival rates $\mathbf{\Lambda}\in\mathcal{C}$ with $\rho<1$, there exists a scheduling algorithm such that the network is stable, and if $\rho>1$, then there exists no scheduling algorithm that can stabilize the network [1]. When $\rho<1$, if the arrival process is such that the system can be described as a Markov chain, then there exists a scheduling algorithm which ensures that the Markov chain is positive recurrent. This is what we mean by "stability" above.

A scheduling algorithm which stabilizes the network for all arrival rates with $\rho<1$ is said to be *throughput optimal.* We say that the network is critically loaded if the traffic load $\rho=1$, which means that the set of arrival rates is on the boundary of the capacity region. We say that the traffic is heavy if ρ is very close to but less than 1.

We are interested in finding scheduling algorithms that minimize the total number of packets in the network. We will refer to such algorithms as being *workload optimal*, where workload refers to the total number of packets in the network.

2.3 Three-Link Line Network under the One-Hop Interference Model

2.3.1 Network Topology and the Interference Model

Consider a 3-link line network under the 1-hop interference model. As shown in Figure 2.3, there are two valid maximal schedules, one that includes link 2 and one that includes links 1 and 3; i.e., the feasible schedules are $a := \{2\}$ and $b := \{1, 3\}$. (Note that we have defined a schedule by the set of links included in the schedule instead of a vector of bits denoting whether each link is included in the schedule or not. This is a more convenient representation for exposition while the vector notation is more convenient to define the capacity region as in the previous section. We will abuse our definition in this manner for the sake of clarity of presentation.)

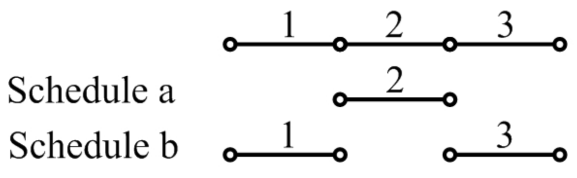

Figure 2.3: Possible maximal schedules for the 3-link line network under the 1-hop interference model.

It is not very difficult to see that the capacity region of this network is given by $\mathcal{C} = \{(\lambda_1, \lambda_2, \lambda_3) | \lambda_1 + \lambda_2 \leq 1, \lambda_2 + \lambda_3 \leq 1\}$. In the remainder of this section we propose a scheduling algorithm which minimizes the total number of packets in the network at each time slot.

2.3.2 Scheduling Algorithm

Scheduling Algorithm: At each time slot, observe the queue lengths after arrivals occur. Then, execute the following algorithm:

- if ($q_1 > 0$ and $q_3 > 0$) or if ($q_2 = 0$), use schedule $b = \{1, 3\}$,
- if ($q_1 = 0$ or $q_3 = 0$) and ($q_2 > 0$), use schedule $a = \{2\}$.

Notice that the scheduling decision is made based on whether the queues are backlogged or not and not based on exact queue lengths.

It should be clear that the main idea behind this algorithm is to drain as many packets as possible from the network; however, when both schedules are able to drain only one packet, it is important to choose the right schedule. In particular, if links 1 and 3 are backlogged, we use schedule b and remove two packets from the system. If only one of links 1 and 3 is backlogged, then we can at most remove one packet; in such a case, we choose to use schedule a. The intuition behind this choice is straightforward: we prefer to serve links 1 and 3 simultaneously since this would allow us to drain two packets in one time slot. If this is not possible, we give preference to link 2 and hope that both links 1 and 3 are backlogged sometime in the future. The scheduling algorithm that we have proposed here is similar in spirit to the optimal resource allocation algorithm for a 2-link, 3-flow connection-level model presented in [36].

2.3.3 Workload Optimality

In Proposition 2.3.2, we will show that our algorithm minimizes the total workload along every sample path. This automatically implies that our algorithm is throughput-optimal. Note that any naive algorithm may not be throughput optimal. Consider, for example, the following algorithm: at each time slot, observe the queue lengths after arrivals occur, and then

- use schedule $a = \{2\}$, if $q_1 = q_3 = 0$,
- use schedule $b = \{1, 3\}$, otherwise.

Queues q_1 and q_3 have the highest priority; i.e., they are always scheduled unless they are both zero. Queue q_1 behaves as a single-server queue with arrival rate λ_1 and potential departure rate 1. Similarly, queue q_3 behaves as a single-server queue with arrival rate λ_3 and potential departure rate 1. For simplicity, assume that the arrival processes to the links are independent, Bernoulli processes. In this case, the fraction of time we can use schedule a is $P(q_1 = q_3 = 0) = P(q_1 = 0)P(q_3 = 0) = (1 - \lambda_1)(1 - \lambda_3)$. Thus, q_2 is stable only if $(1 - \lambda_1)(1 - \lambda_3) > \lambda_2$. This set of link arrival rates is a strict subset of the link arrival rates that satisfy $\lambda_1 + \lambda_2 < 1$ and $\lambda_2 + \lambda_3 < 1$. Therefore, this algorithm is not throughput optimal. We note that this example is similar to the instability of priority algorithms in 2-link connection-level models of congestion control presented in [37].

Our scheduling algorithm is sample-path optimal, i.e., the total queue length is minimized at each step. Our algorithm is the same as the optimal algorithm derived in [27].

Lemma 2.3.1. *Let $q_l[k]$, $l = 1, 2, 3$ be the queue lengths at time slot k under our algorithm, and $\hat{q}_l[k]$, $l = 1, 2, 3$ be the queue lengths at time slot k under some other arbitrary scheduling algorithm. Assume that the initial queue lengths and the arrivals are the same under both algorithms. The following inequalities hold at all times:*

$$q_1[k] + q_2[k] \le \hat{q}_1[k] + \hat{q}_2[k]$$
$$q_2[k] + q_3[k] \le \hat{q}_2[k] + \hat{q}_3[k].$$

Proof. Recall that a maximum of one packet can be drained from links 1 and 2. Since our algorithm drains one such packet whenever possible, the sum of the queue lengths at links 1 and 2 under our algorithm must be less than or equal to that of any other algorithm. A similar observation holds for links 2 and 3. □

Proposition 2.3.2. *Our scheduling algorithm is sample-path optimal, i.e., at all times, we have*

$$q_1[k] + q_2[k] + q_3[k] \le \hat{q}_1[k] + \hat{q}_2[k] + \hat{q}_3[k].$$

Proof. We prove the proposition by contradiction. Let r be the first time when the inequality does not hold; i.e., $\sum_{l=1}^{3} q_l[r] > \sum_{l=1}^{3} \hat{q}_l[r]$. If $\sum_{l=1}^{3} q_l[r-1] = 0$, i.e., all queues are empty at time slot $r-1$, then it is easy to see that the inequality $\sum_{l=1}^{3} q_l[r] \le \sum_{l=1}^{3} \hat{q}_l[r]$ holds. So, we assume that $\sum_{l=1}^{3} q_l[r-1] > 0$. In this case, the following equality must hold: $\sum_{l=1}^{3} q_l[r-1] = \sum_{l=1}^{3} \hat{q}_l[r-1]$. To see this, note that, if $\sum_{l=1}^{3} q_l[r-1] < \sum_{l=1}^{3} \hat{q}_l[r-1]$, then since our algorithm will remove at least one packet whenever possible and any other algorithm can remove at most two packets, at time r, we cannot have the inequality $\sum_{l=1}^{3} q_l[r] > \sum_{l=1}^{3} \hat{q}_l[r]$. Thus, the following facts must be true at time instant $r-1$: $\sum_{l=1}^{3} q_l[r-1] = \sum_{l=1}^{3} \hat{q}_l[r-1]$, our algorithm serves one packet at $r-1$, and the arbitrary algorithm serves two packets at $r-1$. This further implies that the following condition must hold:

$$(\hat{q}_1[r-1] + \alpha_1[r-1] > 0 \text{ and } \hat{q}_3[r-1] + \alpha_3[r-1] > 0)$$
$$\text{and}$$
$$(q_1[r-1] + \alpha_1[r-1] = 0 \text{ or } q_3[r-1] + \alpha_3[r-1] = 0).$$

Without loss of generality, we assume the following:

$$(\hat{q}_1[r-1] + \alpha_1[r-1] > 0 \text{ and } \hat{q}_3[r-1] + \alpha_3[r-1] > 0)$$
$$\text{and}$$
$$(q_1[r-1] + \alpha_1[r-1] > 0 \text{ and } q_3[r-1] + \alpha_3[r-1] = 0).$$

From the fact that $q_3[r-1] + \alpha_3[r-1] = 0$ we get $q_3[r-1] = 0$ and $\alpha_3[r-1] = 0$. Combining this with the fact that $\hat{q}_3[r-1]+\alpha_3[r-1] > 0$, we have $\hat{q}_3[r-1] > 0$. Recalling that $\sum_{l=1}^{3} q_l[r-1] = \sum_{l=1}^{3} \hat{q}_l[r-1]$, we have the inequality $q_1[r-1]+q_2[r-1] > \hat{q}_1[r-1]+\hat{q}_2[r-1]$, which directly contradicts Lemma 2.3.1. Hence, there exists no time r such that $\sum_{l=1}^{3} q_l[r] > \sum_{l=1}^{3} \hat{q}_l[r]$. □

While this algorithm is workload optimal, it is not clear if MWS-α can perform as well as this algorithm. Later we will show through simulations that the optimal algorithm performs better than MWS-α.

2.4 Four-Link Line Network under the Two-Hop Interference Model

Consider a 4-link line network under the 2-hop interference model. There are three valid maximal schedules, i.e., schedule $a := \{2\}$, $b := \{3\}$, and $c := \{1, 4\}$, as shown in Figure 2.4.

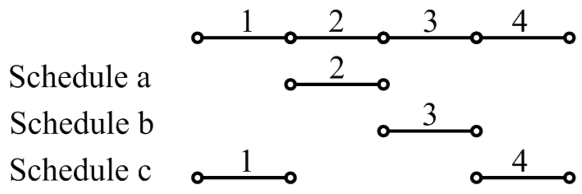

Figure 2.4: Possible maximal schedules for the 4-link line network under the 2-hop interference model.

In this network, links 1, 2, and 3 cannot be scheduled at the same time, neither can links 2, 3, and 4. The capacity region is given by $\mathcal{C} = \{(\lambda_1, \lambda_2, \lambda_3, \lambda_4) | \lambda_1+\lambda_2+\lambda_3 \leq 1, \lambda_2+\lambda_3+\lambda_4 \leq 1\}$.

We will show that the following scheduling algorithm for the 4-link line network under the 2-hop interference model is stable and workload optimal in a sample-path sense.

Scheduling Algorithm: At each time slot, observe the queue lengths after arrivals occur. Then

- if ($q_1 > 0$ and $q_4 > 0$) or ($q_2 = q_3 = 0$), use schedule $c = \{1, 4\}$,
- if ($q_1 = 0$ or $q_4 = 0$) and ($q_2 > 0$ and $q_3 = 0$), use schedule $a = \{2\}$,
- if ($q_1 = 0$ or $q_4 = 0$) and ($q_2 = 0$ and $q_3 > 0$), use schedule $b = \{3\}$,
- if ($q_1 = 0$ or $q_4 = 0$) and ($q_2 > 0$ and $q_3 > 0$), use schedule $a = \{2\}$ or $b = \{3\}$ arbitrarily.

Again, the main idea is to drain as many packets as possible from the network in each time slot, but to break ties carefully. If either link 1 or link 4 or both are not backlogged, we can at most remove one packet; in that case, we give higher priority to link 2 or link 3 if they are backlogged. This give us a better chance to serve two packets simultaneously in the future.

The throughput-optimality of our algorithm uses the same idea as in the stability proof for the 3-link line network under the 1-hop interference model. Let $q_{123} := q_1 + q_2 + q_3$ and $q_{234} := q_2 + q_3 + q_4$. The queues q_{123} and q_{234} can be thought of as two single-server queueing systems with arrival rates $\lambda_1 + \lambda_2 + \lambda_3$ and $\lambda_2 + \lambda_3 + \lambda_4$, respectively. The queues q_{123} and q_{234} are stable if $\lambda_1 + \lambda_2 + \lambda_3 < 1$ and $\lambda_2 + \lambda_3 + \lambda_4 < 1$, respectively. Therefore, (q_1, q_2, q_3, q_4) is stable and the proposed scheduling algorithm is throughput optimal.

Next, we prove that our proposed scheduling algorithm is sample-path optimal. Let $q_l[k]$, for $l = 1, 2, 3, 4$ be the queue lengths at time slot k under our algorithm, and $\hat{q}_l[k]$, for $l = 1, 2, 3, 4$ the queue lengths at time slot k under any arbitrary scheduling algorithm. Assume that the initial queue lengths are equal, i.e., $q_l[0] = \hat{q}_l[0]$, $\forall l$ and that the arrivals are exactly the same under both algorithms.

Lemma 2.4.1. *Let $q_l[k]$, $l = 1, 2, 3, 4$ be the queue lengths at time slot k under our algorithm, and $\hat{q}_l[k]$, $l = 1, 2, 3, 4$ be the queue lengths at time slot k under any arbitrary scheduling algorithm. Assume that the initial queue lengths are equal, i.e., $q_l[0] = \hat{q}_l[0]$, $\forall l$ and that the arrivals are exactly the same under both algorithms. The following inequalities hold at all times:*

$$q_1[k] + q_2[k] + q_3[k] \leq \hat{q}_1[k] + \hat{q}_2[k] + \hat{q}_3[k]$$
$$q_2[k] + q_3[k] + q_4[k] \leq \hat{q}_2[k] + \hat{q}_3[k] + \hat{q}_4[k].$$

Proof. Recall that a maximum of one packet can be drained from links 1, 2, and 3. Since our algorithm drains one such packet whenever possible, the sum of the queue lengths at links 1, 2, and 3 under our algorithm must be less than or equal to that of any other algorithm. A similar observation holds for links 2, 3, and 4. □

Proposition 2.4.2. *Our scheduling algorithm is sample-path optimal; i.e., the following inequality holds at all times:*

$$q_1[k] + q_2[k] + q_3[k] + q_4[k] \leq \hat{q}_1[k] + \hat{q}_2[k] + \hat{q}_3[k] + \hat{q}_4[k].$$

Proof. The proof is similar to the one of Proposition 2.3.2 and is therefore omitted. Note that the 4-link, 2-hop model can be thought as being equivalent to the 3-link 1-hop model as follows: the middle two links can be thought as a single link since only one of them can be scheduled at a time. □

2.5 Four-Link Line Network under the One-Hop Interference Model

In the previous two sections, we were able to derive sample-path optimal algorithms. In this section, we consider a model for which a simple sample-path optimal algorithm does not seem to exist. Instead, we present a class of scheduling algorithms which is sample-path optimal in a heavy-traffic sense.

2.5.1 Network Topology and the Interference Model

Consider a 4-link line network under the 1-hop (node-exclusive) interference model. There are three valid maximal schedules, namely schedules $a := \{2, 4\}$, $b := \{1, 3\}$, and $c := \{1, 4\}$, as shown in Figure 2.5.

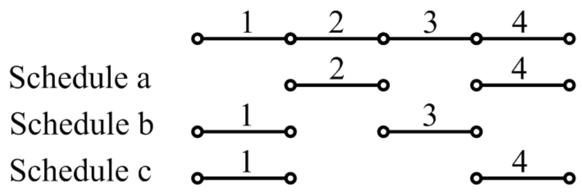

Figure 2.5: Possible maximal schedules for the 4-link line network under the 1-hop interference model.

The capacity region is given by $\mathcal{C} = \{(\lambda_1, \lambda_2, \lambda_3, \lambda_4) | \lambda_1 + \lambda_2 \leq 1, \lambda_2 + \lambda_3 \leq 1, \lambda_3 + \lambda_4 \leq 1\}$. Next, we propose a class of scheduling algorithms and show its throughput optimality and its sample-path optimality in a heavy-traffic setting.

2.5.2 The Scheduling Algorithms

A Class of Scheduling Algorithms: At each time slot, observe the queue lengths after arrivals occur. Then

- if $q_2 > 0$ and $q_3 > 0$, then
  - use schedule a if $q_1 = 0$ and $q_4 > 0$,
  - use schedule b if $q_1 > 0$ and $q_4 = 0$,
  - use either schedule a or b, otherwise.
- If $q_2 = 0$ and $q_3 > 0$, use schedule b.
- If $q_2 > 0$ and $q_3 = 0$, use schedule a.
- If $q_2 = 0$ and $q_3 = 0$, use schedule c.

This class of scheduling algorithms tries to remove two packets from the system whenever possible and, in addition, gives priority to schedules a and b over schedule c when all three are capable of serving the same number of packets. Observe that the pair of links 2 and 3 cannot be scheduled simultaneously, while the pair of links 1 and 4 can. In order to avoid situations where the only backlogged links are links 2 and 3, our algorithm gives the lowest priority to schedule c.

2.5.3 Throughput Optimality

To establish the optimality properties of the class of scheduling algorithms, we make the following additional assumptions on the arrival processes: the arrival process of each link is Bernoulli, i.e., at each time slot, there is exactly one packet arrival to link l with probability λ_l, and there is no arrival with probability $1 - \lambda_l$.

We provide an informal argument to show that our algorithms are throughput optimal. Notice that exactly one packet is removed from the pair of links 2 and 3 unless $q_2 = q_3 = 0$.

Let $q_{23} := q_2 + q_3$. The Markov chain describing the evolution of q_{23} is stable if $\lambda_2 + \lambda_3 < 1$. In this case, the fraction of time that schedule a is used (which is the same as the fraction of time that link 2 is backlogged) is equal to λ_2. Similarly, the fraction of time that schedule b is used is equal to λ_3. Since schedule c is used only when links 2 and 3 are not backlogged, the fraction of time that schedule c is used is equal to $1 - \lambda_2 - \lambda_3$. (This follows from a flow conservation argument or by analyzing the single-server queueing dynamics of q_{23}.) Thus, the available service to link 1 (i.e., the fraction of time that either schedule b or c is used) is equal to $1 - \lambda_2$. Thus, queue q_1 is stable if $\lambda_1 < 1 - \lambda_2 \Rightarrow \lambda_1 + \lambda_2 < 1$. Similarly, queue q_4 is stable if $\lambda_3 + \lambda_4 < 1$. In summary, all queues in the network are stable if

$$\lambda_1 + \lambda_2 < 1, \quad \lambda_3 + \lambda_4 < 1, \quad \lambda_2 + \lambda_3 < 1.$$

This establishes the throughput optimality of our algorithms. As mentioned earlier, the throughput optimality can be established under quite general assumptions on the arrival processes. However, our proof technique for establishing workload optimality under the heavy-traffic regime uses the Bernoulli arrival process assumption as will be seen in the next subsection.

2.5.4 Workload Optimality

Next, we will show that our algorithms are heavy-traffic optimal. Our heavy-traffic optimality result holds for every algorithm in our class of algorithms. In particular, to specify an algorithm, we have to specify which schedule a or b should be used when there is a choice between the two. Our results hold for any deterministic or randomized specification of an algorithm in such a case. For the rest of the chapter, we will not consider a particular algorithm, but the reader should assume that a particular randomized or deterministic algorithm in our class of algorithms has been specified.

In order to analyze the network in a heavy-traffic regime, we consider a sequence of networks indexed by parameter n where the n^{th} system has arrival rates given by

$$\lambda_l^{(n)} = \lambda_l - \frac{m_l}{\sqrt{n}}, \text{ for } l = 1, 2, 3, 4,$$
$$\lambda_1 + \lambda_2 = 1;\ \lambda_3 + \lambda_4 = 1;\ \lambda_2 + \lambda_3 < 1,$$

where $m_l \geq 0$ are constants. Let $\mathbf{\Lambda}^{(n)}$ denote the vector of arrival rates for the n^{th} system and let $\rho^{(n)}$ denote the traffic load in the n^{th} network. For simplicity, we assume that the arrival processes of all links are Bernoulli. To avoid further definitions, we remark that we will use the superscript (n) to denote the arrival and departures process of the n^{th} system.

Note that $\rho^{(n)} \to 1$ as $n \to \infty$ and the traffic load on the n^{th} network satisfies

$$n^{1/2}(1 - \rho^{(n)}) = \min\{m_1 + m_2, m_3 + m_4\}, \quad \forall n.$$

Further, in the limit as $n \to \infty$, the total arrival rates on links 1 and 2 and on links 3 and 4 approach their respective capacities, while the total arrival rate on links 2 and 3 is strictly less than its capacity. Thus, there are two critically loaded resources in the system: links 1 and 2 considered together and links 3 and 4 considered together; see [31, 19] for a precise definition. The third resource (links 2 and 3 considered together) is underloaded. In heavy traffic, one would expect the performance of a well-designed scheduling algorithm to be primarily constrained by the capacity of the two critically loaded resources, but not by the third resource in the network. The case where all three resources are critically loaded is harder to handle and we do not consider it here.

Let $q_{12}^{(n)}[k] := q_1^{(n)}[k] + q_2^{(n)}[k]$ be the sum of the queue lengths of links 1 and 2 under our algorithm for the n^{th} network at the beginning of time slot k. Let $\tilde{q}_{12}^{(n)}[k]$ be the queue length at the beginning of time slot k in a single-server queueing system which serves one packet in each time slot and whose arrival process is given by $\alpha_{12}^{(n)}[k] := \alpha_1^{(n)}[k] + \alpha_2^{(n)}[k]$. Thus, while $q_{12}^{(n)}[k]$ is not guaranteed to decrease by one when it is backlogged (for example, if $q_1^{(n)}[k] + \alpha_1^{(n)}[k] = 0$ and $q_2^{(n)}[k] + \alpha_2^{(n)}[k] > 0$ and schedule b is used), $\tilde{q}_{12}[k]$ is always guaranteed to decrease by one when it is backlogged. We assume that $\tilde{q}_{12}^{(n)}[0] = q_{12}^{(n)}[0]$. Since the arrival processes to both $q_{12}^{(n)}[k]$ and $\tilde{q}_{12}^{(n)}[k]$ are identical, it immediately follows that $\tilde{q}_{12}^{(n)}[k] \leq q_{12}^{(n)}[k]$. Similarly, we can construct process $\tilde{q}_{34}^{(n)}[k]$ such that $\tilde{q}_{34}^{(n)}[k] \leq q_{34}^{(n)}[k]$. Thus, $\tilde{q}_{12}^{(n)}[k] + \tilde{q}_{34}^{(n)}[k]$ is a lower bound on the workload in the system. In the remainder of this section, our goal is to show that there is a matched upper bound under appropriate scaling in heavy traffic.

Let $\tilde{u}_{12}^{(n)}[k]$ be the unused service of $\tilde{q}_{12}^{(n)}[k]$ and let $u_{12}^{(n)}[k]$ be the unused service of $q_{12}^{(n)}[k]$. Also, we define a quantity $w_{12}^{(n)}[k]$, which we call the *wasted service* of $q_{12}^{(n)}$, as

$$w_{12}^{(n)}[k] := \begin{cases} 1, & \text{if } q_1^{(n)}[k] + \alpha_1^{(n)}[k] = 0, \\ & q_2^{(n)}[k] + \alpha_2^{(n)}[k] > 0,\ d_1^{(n)}[k] = 1 \\ 0, & \text{otherwise.} \end{cases}$$

Note that there is wasted service on links $\{1, 2\}$ only if link 2 is backlogged and link 1 is not, and our algorithm chooses schedule b. Also define $w_{34}^{(n)}[k]$, $\tilde{u}_{34}^{(n)}[k]$, and $u_{34}^{(n)}[k]$ in a similar way. Let $\hat{i}^{(n)}[k] = \mathbb{1}\{q_1^{(n)}[k] = q_4^{(n)}[k] = 0\}$, where $\mathbb{1}$ is the indicator function. A little thought shows that, for our algorithm, we have $w_{12}^{(n)}[k] \leq \hat{i}^{(n)}[k]$ and $w_{34}^{(n)}[k] \leq \hat{i}^{(n)}[k]$. To see this, note that if q_1 and q_4 are empty at the beginning of a time slot, then $\hat{i}^{(n)}$ is equal to 1 in which case the upper bound on $w_{12}^{(n)}$ obviously holds. On the other hand, if either q_1 or q_4 is non-empty at the beginning of a time slot, then one of these queues will be backlogged and hence, there cannot be any wasted service.

The following lemma establishes an upper bound on $q_{12}^{(n)}[k]$ and $q_{34}^{(n)}[k]$.

Lemma 2.5.1. *The following inequalities hold at all times assuming that* $q_{12}^{(n)}[0] = \tilde{q}_{12}^{(n)}[0]$ *and* $q_{34}^{(n)}[0] = \tilde{q}_{34}^{(n)}[0]$:

$$q_{12}^{(n)}[k] \leq \tilde{q}_{12}^{(n)}[k] + \textstyle\sum_{r=0}^{k-1} \hat{i}^{(n)}[r],$$
$$q_{34}^{(n)}[k] \leq \tilde{q}_{34}^{(n)}[k] + \textstyle\sum_{r=0}^{k-1} \hat{i}^{(n)}[r].$$

Proof. Since $\tilde{d}_{12}^{(n)}[k] = d_{12}^{(n)}[k] = 1$, the unused services are given by

$$\begin{aligned} \tilde{u}_{12}^{(n)}[k] &= \left[-(\tilde{q}_{12}^{(n)}[k] + \alpha_{12}^{(n)}[k] - 1)\right]^+ \\ u_{12}^{(n)}[k] &= \left[-(q_{12}^{(n)}[k] + \alpha_{12}^{(n)}[k] - 1)\right]^+. \end{aligned}$$

Using the fact that $\tilde{q}_{12}^{(n)}[k] \leq q_{12}^{(n)}[k]$, we get $\tilde{u}_{12}^{(n)}[k] \geq u_{12}^{(n)}[k]$, $\forall k \geq 0$. And noting that $\tilde{q}_{12}^{(n)}[0] = q_{12}^{(n)}[0]$ and $w_{12}^{(n)}[k] \leq \hat{\imath}^{(n)}[k]$, we have the following result:

$$\begin{aligned}
\tilde{q}_{12}^{(n)}[k] + \textstyle\sum_{r=0}^{k-1} \hat{\imath}^{(n)}[r] &\geq \tilde{q}_{12}^{(n)}[k] + \textstyle\sum_{r=0}^{k-1} w_{12}^{(n)}[r] \\
&= \tilde{q}_{12}^{(n)}[0] + \textstyle\sum_{r=0}^{k-1} (\alpha_{12}^{(n)}[r] - 1 + \tilde{u}_{12}^{(n)}[r] + w_{12}^{(n)}[r]) \\
&\geq q_{12}^{(n)}[0] + \textstyle\sum_{r=0}^{k-1} (\alpha_{12}^{(n)}[r] - 1 + u_{12}^{(n)}[r] + w_{12}^{(n)}[r]) \\
&= q_{12}^{(n)}[k].
\end{aligned}$$

Similarly, $\tilde{q}_{34}^{(n)}[k] + \sum_{r=0}^{k-1} \hat{\imath}^{(n)}[r] \geq q_{34}^{(n)}[k]$. □

Next, we use the lower-bounding queueing systems to upper bound the probability of links 1 and 4 not being backlogged.

Lemma 2.5.2. *Let $q_{23}[k] := q_2[k] + q_3[k]$; the following inequality holds for the n^{th} network at any time slot k and for any integer $B > 0$:*

$$P(q_1^{(n)}[k] = q_4^{(n)}[k] = 0) \leq P(\tilde{q}_{12}^{(n)}[k] < B)\ P(\tilde{q}_{34}^{(n)}[k] < B) + 2P(q_{23}^{(n)}[k] \geq B).$$

Proof.

$$\begin{aligned}
& P(q_1^{(n)}[k] = q_4^{(n)}[k] = 0) \\
= \;& P(q_1^{(n)}[k] = q_4^{(n)}[k] = 0, q_2^{(n)}[k] < B) + P(q_1^{(n)}[k] = q_4^{(n)}[k] = 0, q_2^{(n)}[k] \geq B) \\
\leq \;& P(q_1^{(n)}[k] = q_4^{(n)}[k] = 0, q_2^{(n)}[k] < B) + P(q_2^{(n)}[k] \geq B) \\
\leq \;& P(q_1^{(n)}[k] = q_4^{(n)}[k] = 0, q_2^{(n)}[k] < B, q_3^{(n)}[k] < B) + P(q_2^{(n)}[k] \geq B) + P(q_3^{(n)}[k] \geq B) \\
\leq \;& P(q_{12}^{(n)}[k] < B, q_{34}^{(n)}[k] < B) + 2P(q_{23}^{(n)}[k] \geq B) \\
\leq \;& P(\tilde{q}_{12}^{(n)}[k] < B, \tilde{q}_{34}^{(n)}[k] < B) + 2P(q_{23}^{(n)}[k] \geq B) \\
= \;& P(\tilde{q}_{12}^{(n)}[k] < B) P(\tilde{q}_{34}^{(n)}[k] < B) + 2P(q_{23}^{(n)}[k] \geq B),
\end{aligned}$$

where the last line follows from the fact that the arrival processes to $\tilde{q}_{12}^{(n)}$ and $\tilde{q}_{34}^{(n)}$ are independent. □

Recall that our scheduling algorithm was designed to be throughput optimal. Thus, the Markov chain $\mathbf{q}^{(n)}[k] := (q_1^{(n)}[k], q_2^{(n)}[k], q_3^{(n)}[k], q_4^{(n)}[k])$ is positive recurrent and there exists a stationary distribution $\pi^{(n)}$ for the n^{th} network. The main result of this section establishes the sample-path optimality of our scheduling algorithm in the heavy-traffic regime provided that the initial queue lengths in the network are distributed according to the stationary distribution. Clearly the assumption on the initial condition is restrictive; however, the proof presented here contains the main ideas required to establish the result where the initial conditions are not assumed to be in steady state.

Proposition 2.5.3. *Assume that $\mathbf{q}^{(n)}[0]$ is distributed according to $\pi^{(n)}$. Then, our algorithm is sample-path optimal in heavy traffic; i.e., for any $c > 0$ and any fixed finite $T > 0$:*

$$\lim_{n\to\infty} P\Big(\sup_{t\in[0,T]} \frac{\sum_l q_l^{(n)}[\lfloor nt\rfloor] - \sum_l \tilde{q}_l^{(n)}[\lfloor nt\rfloor]}{\sqrt{n}} \geq c\Big) = 0.$$

Proof. From the lower bound on $q_{12}^{(n)}$ and Lemma 2.5.2, we get $\tilde{q}_{12}^{(n)}[\lfloor nt\rfloor] \leq q_{12}^{(n)}[\lfloor nt\rfloor] \leq \tilde{q}_{12}^{(n)}[\lfloor nt\rfloor] + \sum_{k=0}^{\lfloor nt\rfloor-1} \hat{\imath}^{(n)}[k]$. Thus,

$$\begin{aligned} P\Big(\sup_{t\in[0,T]} (q_{12}^{(n)}[\lfloor nt\rfloor] - \tilde{q}_{12}^{(n)}[\lfloor nt\rfloor]) \geq \frac{c\sqrt{n}}{2}\Big) &\leq P(\sup_{t\in[0,T]} \sum_{k=0}^{\lfloor nt\rfloor-1} \hat{\imath}^{(n)}[k] \geq \frac{c\sqrt{n}}{2}) \\ &= P(\sum_{k=0}^{\lfloor nT\rfloor-1} \hat{\imath}^{(n)}[k] \geq \frac{c\sqrt{n}}{2}) \qquad (2.1) \\ &\leq \frac{2}{c\sqrt{n}} \sum_{k=0}^{\lfloor nT\rfloor-1} \mathbb{E}(\hat{\imath}^{(n)}[k]) \leq \frac{2T\sqrt{n}}{c}\mathbb{E}(\hat{\imath}^{(n)}[j]), \end{aligned}$$

where the second inequality is the Markov inequality and the last inequality holds for all j due to the fact that the initial condition is distributed according to the stationary distribution. Noticing that for all j we have

$$\mathbb{E}(\hat{\imath}^{(n)}[j]) = P(q_1^{(n)}[j] = q_4^{(n)}[j] = 0),$$

we can use Lemma 2.5.2 and further bound (2.2) by upper-bounding $P(\tilde{q}_{12}^{(n)}[j] < B)$, $P(\tilde{q}_{34}^{(n)}[j] < B)$, and $P(q_{23}^{(n)}[j] \geq B)$ for a convenient choice of j. We will let $j \to \infty$; thus the upper-bound (2.2) would use the steady-state distributions of the Markov chains $\tilde{q}_{12}^{(n)}$, $\tilde{q}_{34}^{(n)}$ and $q_{23}^{(n)}$ which are easy to compute.

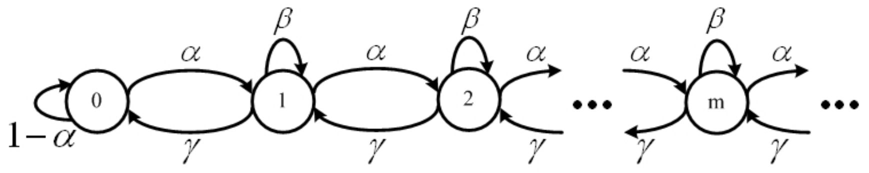

Figure 2.6: State transition diagram of a discrete-time single-server queue.

The state transition diagrams for the Markov chains $\tilde{q}_{12}^{(n)}$, $\tilde{q}_{34}^{(n)}$, and $q_{23}^{(n)}$ have the structure shown in Figure 2.6, but each one has different values for α, γ, and $\beta = 1 - \alpha - \gamma$. In fact, for $\tilde{q}_{12}^{(n)}$ we have $\alpha = \lambda_1^{(n)}\lambda_2^{(n)}$ and $\gamma = (1 - \lambda_1^{(n)})(1 - \lambda_2^{(n)})$; for $q_{23}^{(n)}$ we have $\alpha = \lambda_2^{(n)}\lambda_3^{(n)}$ and $\gamma = (1 - \lambda_2^{(n)})(1 - \lambda_3^{(n)})$; and for $q_{34}^{(n)}$ we have $\alpha = \lambda_3^{(n)}\lambda_4^{(n)}$ and $\gamma = (1 - \lambda_3^{(n)})(1 - \lambda_4^{(n)})$. To simplify notation, let $\nu_{12n} := \frac{\lambda_1^{(n)}\lambda_2^{(n)}}{(1-\lambda_1^{(n)})(1-\lambda_2^{(n)})}$, $\nu_{23n} := \frac{\lambda_2^{(n)}\lambda_3^{(n)}}{(1-\lambda_2^{(n)})(1-\lambda_3^{(n)}),}$ and $\nu_{34n} := \frac{\lambda_3^{(n)}\lambda_4^{(n)}}{(1-\lambda_3^{(n)})(1-\lambda_4^{(n)})}$.

Solving for the steady-state distribution of the Markov chain depicted in Figure 2.6, we can express the steady-state distributions $P(\tilde{q}_{12}^{(n)}[\infty] < B)$, $P(\tilde{q}_{34}^{(n)}[\infty] < B)$, and $P(q_{23}^{(n)}[\infty] \geq B)$ (we abuse notation a bit here and use $[\infty]$ to denote that the system is in steady-state) in terms of the ν's defined above as

$$\begin{aligned} P(\tilde{q}_{12}^{(n)}[\infty] < B) &= 1 - \nu_{12n}^B \\ P(\tilde{q}_{34}^{(n)}[\infty] < B) &= 1 - \nu_{34n}^B \\ P(q_{23}^{(n)}[\infty] \geq B) &= \nu_{23n}^B. \end{aligned}$$

Since $\lambda_1 + \lambda_2 = 1$ and $\lambda_3 + \lambda_4 = 1$, for large n, ν_{12n} and ν_{34n} are less than but very close to 1. To quantify how close they are, we bound ν_{12n} and ν_{34n} as follows:

$$\begin{aligned}
\nu_{12n} &= \frac{(\lambda_1 - \frac{m_1}{\sqrt{n}})(\lambda_2 - \frac{m_2}{\sqrt{n}})}{(1-\lambda_1+\frac{m_1}{\sqrt{n}})(1-\lambda_2+\frac{m_2}{\sqrt{n}})} \\
&\geq (1-\frac{m_1}{\lambda_1\sqrt{n}})(1-\frac{m_1}{\lambda_2\sqrt{n}})(1-\frac{m_2}{\lambda_2\sqrt{n}})(1-\frac{m_2}{\lambda_1\sqrt{n}}) \\
&= (1-\frac{m_1}{\lambda_1\lambda_2\sqrt{n}}+\frac{m_1^2}{\lambda_1\lambda_2 n})(1-\frac{m_2}{\lambda_1\lambda_2\sqrt{n}}+\frac{m_2^2}{\lambda_1\lambda_2 n}) \\
&\geq 1-\frac{m_1+m_2}{\lambda_1\lambda_2\sqrt{n}}+\frac{m_1 m_2}{\lambda_1^2\lambda_2^2 n}+\frac{m_1^2 m_2^2}{\lambda_1^2\lambda_2^2 n^2} \\
&\geq 1-\frac{(m_1+m_2)}{\lambda_1\lambda_2\sqrt{n}}. \qquad (2.2)
\end{aligned}$$

To get the second line in the previous sequence of expressions, we used the facts $(1+x)^{-1} \geq 1-x,\ \forall x \geq 0$, and $\lambda_1+\lambda_2 = 1$.

Using the fact that $(1-x)^B \geq 1-xB,$ for $0 \leq x < 1,\ B > 1$, we can bound $P(\tilde{q}_{12}^{(n)}[\infty] < B)$ as follows:

$$P(\tilde{q}_{12}^{(n)}[\infty] < B) \ \leq\ 1-\Big(1-\frac{m_1+m_2}{\lambda_1\lambda_2\sqrt{n}}\Big)^B \leq \frac{m_1+m_2}{\lambda_1\lambda_2\sqrt{n}}B,$$

where the above bound holds for sufficiently large n. In a similar fashion, we can show that

$$P(\tilde{q}_{34}^{(n)}[\infty] < B) \ \leq\ \frac{m_3+m_4}{\lambda_3\lambda_4\sqrt{n}}B, \qquad (2.3)$$

for sufficiently large n. In other words, there exists N such that, for all $n \geq N$, the bounds (2.3) and (2.3) hold.

Letting $\nu_{23} := \dfrac{\lambda_2\lambda_3}{(1-\lambda_2)(1-\lambda_3)}$, and noting that $\nu_{23n} \leq \nu_{23}$, we have $P(q_{23}^{(n)}[\infty] \geq B) =$

$\nu_{23n}^B \le \nu_{23}^B$. Choosing $B = n^{0.1}$, for $n \ge N$, we have

$$\begin{aligned}
& P\Big(\sup_{t\in[0,T]} (q_{12}^{(n)}[\lfloor nt\rfloor] - \tilde{q}_{12}^{(n)}[\lfloor nt\rfloor]) \ge \frac{c\sqrt{n}}{2}\Big) \\
\le\ & \frac{2T\sqrt{n}}{c} P(\tilde{q}_{12}^{(n)}[\infty] < B) P(\tilde{q}_{34}^{(n)}[\infty] < B) + \frac{4T\sqrt{n}}{c} P(q_{23}^{(n)}[\infty] \ge B) \\
=\ & \frac{2(m_1+m_2)(m_3+m_4)TB^2}{\lambda_1\lambda_2\lambda_3\lambda_4\sqrt{n}c} + \frac{4T}{c}\sqrt{n}\nu_{23}^B \\
=\ & \frac{2(m_1+m_2)(m_3+m_4)T}{\lambda_1\lambda_2\lambda_3\lambda_4 c} n^{-0.3} + \frac{4T}{c}\sqrt{n}\nu_{23}^{n^{0.1}}.
\end{aligned}$$

Noting that $\nu_{23} < 1$, we have

$$\lim_{n\to\infty} P\Big(\sup_{t\in[0,T]} \frac{q_{12}^{(n)}[\lfloor nt\rfloor] - \tilde{q}_{12}^{(n)}[\lfloor nt\rfloor]}{\sqrt{n}} \ge \frac{c}{2}\Big) = 0.$$

Similarly, we can show that

$$\lim_{n\to\infty} P\Big(\sup_{t\in[0,T]} \frac{q_{34}^{(n)}[\lfloor nt\rfloor] - \tilde{q}_{34}^{(n)}[\lfloor nt\rfloor]}{\sqrt{n}} \ge \frac{c}{2}\Big) = 0.$$

Observing that

$$\begin{aligned}
& P\Big(\sup_{t\in[0,T]} \frac{\sum_l q_l^{(n)}[\lfloor nt\rfloor] - \sum_l \tilde{q}_l^{(n)}[\lfloor nt\rfloor]}{\sqrt{n}} \ge c\Big) \\
\le\ & P\Big(\sum_{l=\{12\}}^{\{34\}} \sup_{t\in[0,T]} \frac{q_l^{(n)}[\lfloor nt\rfloor] - \tilde{q}_l^{(n)}[\lfloor nt\rfloor]}{\sqrt{n}} \ge c\Big) \\
\le\ & P\Big(\sup_{t\in[0,T]} \frac{q_{12}^{(n)}[\lfloor nt\rfloor] - \tilde{q}_{12}^{(n)}[\lfloor nt\rfloor]}{\sqrt{n}} \ge \frac{c}{2}\Big) + P\Big(\sup_{t\in[0,T]} \frac{q_{34}^{(n)}[\lfloor nt\rfloor] - \tilde{q}_{34}^{(n)}[\lfloor nt\rfloor]}{\sqrt{n}} \ge \frac{c}{2}\Big),
\end{aligned}$$

we get the desired result. □

2.6 Some Remarks on Various Notions of Optimality

In our analysis of scheduling algorithms for line networks, we have used two notions of workload optimality: *sample-path optimality* and *heavy-traffic sample-path optimality.* The notion of sample-path optimality is clear: a scheduling algorithm is sample-path optimal if it minimizes the total workload in the network at all times, i.e., it minimizes $\sum_l q_l[k],\ \forall k \geq 0$. This is the strongest notion of optimality and we were able to show this form of optimality for the 3-link network under the 1-hop interference model and the 4-link under the 2-hop interference model.

In general, it is unreasonable to expect to be able to derive scheduling algorithms that are sample-path optimal. A more reasonable goal is to try to minimize the steady-state expected workload $\mathbb{E}[\sum_l q_l[\infty]]$ or some related performance measure such as the average cost criterion [38]. The solution approach would be to formulate the problem as a Markov decision problem [38], but except for some special cases, the resulting dynamic programming equations are typically quite intractable. The scheduling problem under consideration seems to fall into the latter category.

A further relaxation is to consider the system when the traffic load on the system is close to its capacity. For this purpose, as in the previous section, we index the system by a parameter n and let $n \to \infty$ so that the corresponding traffic load $\rho^{(n)} \to 1$. From basic queueing theory, we know that the steady-state mean queue length would also go to infinity for all scheduling algorithms when $\rho^{(n)} \to 1$. Thus, to meaningfully compare scheduling algorithms, we have to scale the expected queue length in some manner to keep it finite. For this purpose, we first observe that, for a single-server queue, the expected queue length increases as $1/(1-\rho^{(n)})$ as $\rho^{(n)} \to 1$. Thus, it is reasonable to try to find a scheduling algorithm that minimizes $(1-\rho^{(n)})\mathbb{E}(\sum_l q_l^{(n)}[\infty])$ in the limit as $\rho^{(n)} \to 1$. By choosing the scaling parameter n such that $1-\rho^{(n)}$ is proportional to $1/\sqrt{n}$, the performance measure can be equivalently viewed as $\frac{\mathbb{E}(\sum_l q_l^{(n)}[\infty])}{\sqrt{n}}$. We will refer to an algorithm that minimizes the above objective as being *heavy-traffic expected workload optimal.* In the simulation section, we will use heavy-traffic expected workload as the metric to compare scheduling algorithms.

It turns out that deriving heavy-traffic expected workload optimal scheduling algorithms is also quite difficult; this leads us to the heavy-traffic sample-path optimality criterion used in the previous section. Recall that a scheduling algorithm is heavy-traffic workload optimal if $\lim_{n\to\infty} P\Big(\sup_{t\in[0,T]} \frac{\sum_l q_l^{(n)}[\lfloor nt\rfloor] - \sum_l \hat{q}_l^{(n)}[\lfloor nt\rfloor]}{\sqrt{n}} \geq c\Big) = 0, \forall\ c > 0$ for any fixed $T > 0$ where q_l is the queue length of the algorithm under consideration and $\hat{q}_l$ is the queue length of any other algorithm. Informally, this means that given any $\epsilon > 0$, for sufficiently large n, $\frac{\sum_l q_l^{(n)}[\lfloor nt\rfloor]}{\sqrt{n}} \leq \frac{\sum_l \hat{q}_l^{(n)}[\lfloor nt\rfloor]}{\sqrt{n}} + \epsilon$ with very high probability. If we are able to take expectations of the above inequality in steady state, then this condition would imply heavy-traffic expected workload optimality. However, such an operation is often difficult to justify and therefore, in the considerable literature on heavy-traffic analysis, heavy-traffic sample-path optimality is used as a surrogate for expected workload optimality. This is the sense in which the result in the previous section for the 4-link network under the 1-hop interference model should be interpreted.

In contrast to the above notion of heavy-traffic sample-path optimality which shows that the system behaves optimally over time intervals of the order of n, the result in [19] studies the system over much smaller time-scales of the order of $\sqrt{n}$ (the parameter r in [19] is equivalent to $\sqrt{n}$ here). Hence, it is difficult to directly compare the optimality result in [19, Theorem 10.2] to the notions of optimality mentioned here.

2.7 Simulation Results

Under some conditions, it has been shown that MWS-α minimizes $\sum_l q_l^{1+\alpha}$ in a heavy-traffic sense. Letting α go to zero, the following algorithm has been conjectured as the natural limit of the MWS-α algorithm in [19, Conjecture 10.1]: first, delete all links with zero queue lengths from the network graph, then find all maximum size schedules, and finally, among those pick the one which has the maximum weight breaking any ties arbitrarily, where the weight of link l is $\log q_l$. This algorithm is called the MWS-0 algorithm. In this section, we also compare our algorithms with MWS-α for small α. Specifically, we compare the scaled long-run average workload under the following three set of algorithms: the optimal algorithms derived in this chapter, MWS-0 and MWS-α for $\alpha = 0.01$.

We simulate the three network topologies considered earlier with the following traffic demands:

- 3-link, 1-hop interference model: $\lambda_1 = \lambda_2 = \lambda_3 = 0.5$; $m_1 = m_3 = 0, m_2 = 1$.
- 4-link, 2-hop interference model: $\lambda_1 = \lambda_4 = 0.4, \lambda_2 = \lambda_3 = 0.3$; $m_1 = m_4 = 0, m_2 = m_3 = 0.5$.
- 4-link, 1-hop interference model: $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0.5$; $m_1 = m_4 = 0, m_2 = m_3 = 1$.
- 4-link, 1-hop interference model: $\lambda_1 = \lambda_4 = 0.6, \lambda_2 = \lambda_3 = 0.4$; $m_1 = m_4 = 0, m_2 = m_3 = 1$.

We consider the following arrival process for all cases:

$$\alpha_l^{(n)}[k] := \begin{cases} 5, & \text{with probability } 0.1\lambda_l^{(n)} \\ 10, & \text{with probability } 0.05\lambda_l^{(n)} \\ 0, & \text{otherwise.} \end{cases}$$

For the 4-link, 1-hop interference models above, we have chosen the arrival rates such that all three pairs of consecutive links are in heavy traffic, and further, the arrival process is not Bernoulli. The purpose of the simulations is to test how well our algorithms perform when some of our assumptions do not hold. For simulation purposes, we pick an algorithm from our class of scheduling algorithms which chooses between alternatives with equal probability.

In Figures 2.7, 2.8, 2.9, and 2.10 we plot the scaled steady-state mean sum of queue lengths $(1-\rho^{(n)})\mathbb{E}(\sum_l q_l^{(n)})$ as a function of n, where the steady-state expectation is computed as a long-run average in the simulations. In our simulations, the scaled queue lengths under MWS-0.01 and MWS-0 appear to be nearly equal. However, in Figures 2.7, 2.8, and 2.9, as $n \to \infty$, i.e., as $\rho^{(n)} \to 1$, the difference between the scaled mean sum of queue lengths of our

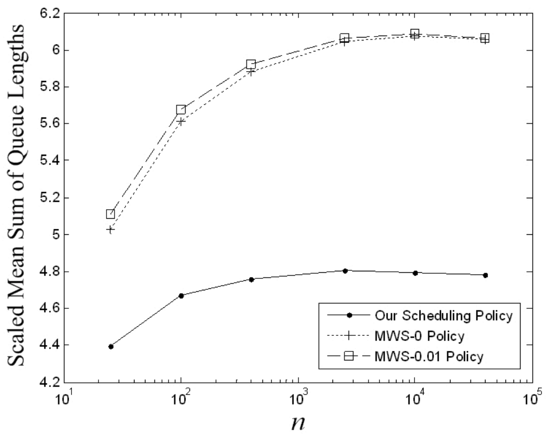

Figure 2.7: Scaled mean sum of queue lengths for the 3-link line network under the 1-hop interference model.

algorithms and that of the MWS-0 and MWS-0.01 scheduling algorithms becomes significant and approaches a constant. In Figure 2.10, the performances of the MWS-0 and MWS-0.01 algorithms are nearly identical to that of our algorithm. From these figures, we can conclude that, while MWS-α algorithms for small α perform nearly as well as our algorithm in some cases, in general this is not the case.

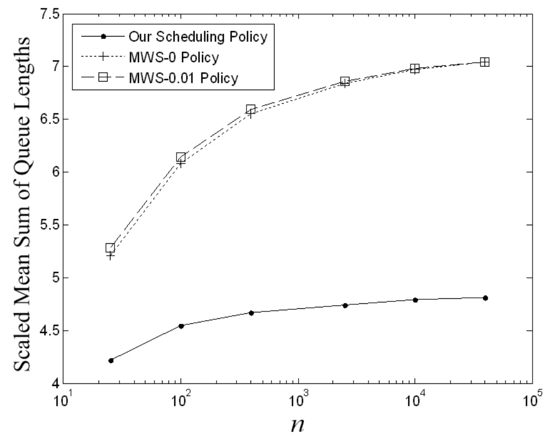

Figure 2.8: Scaled mean sum of queue lengths for 4-link line network under the 2-hop interference model.

We also simulate a MaxWeight scheduling algorithm proposed by Shah and Wischik where

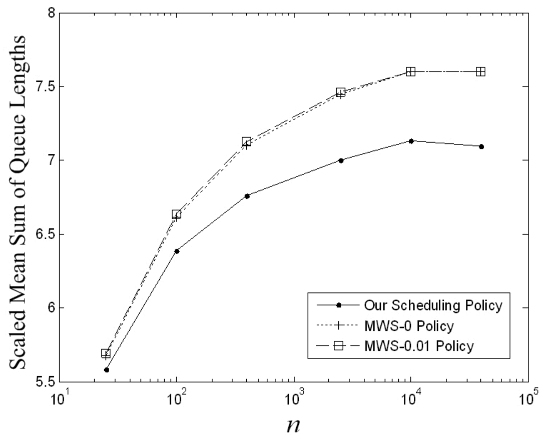

Figure 2.9: Scaled mean sum of queue lengths for 4-link line network under the 1-hop interference model, where $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0.5$.

$\log(1 + q_l)$ is used as the weight of each link l. The performance of such algorithm is very close to the performance of MWS-0 and MWS-0.01 in all of our simulations.

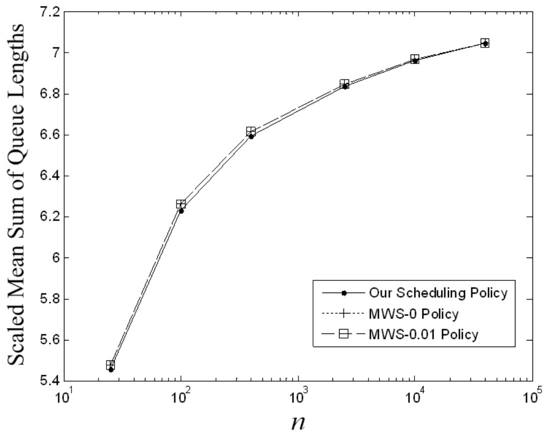

Figure 2.10: Scaled mean sum of queue lengths for 4-link line network under the 1-hop interference model, where $\lambda_1 = \lambda_4 = 0.6, \lambda_2 = \lambda_3 = 0.4$.

We note that the MWS-0 algorithm has not been proven to be optimal. In fact, for a specific model of a generalized switch (which does not include line networks or high-speed switches), Stolyar has shown that MWS-0 is not throughput-optimal [39]. The example in [39] is interesting in that it shows that max-size schedules need not be stabilizing, in general.

2.8 Heavy Traffic Workload Optimality Proof for the Four-Link Network under the One-Hop Interference Model

In Proposition 2.5.3, we proved optimality by assuming that the network starts from steady state. Here we assume that the initial queue lengths of links 1 and 4 in the n^{th} network are of order $\sqrt{n}$, and the initial queue lengths of links 2 and 3 are bounded (instead of assuming that the network starts from its steady state). More specifically, let $\mathcal{D}_1$ denote the initial condition for the n^{th} network; that is, $\mathcal{D}_1 := \{q_{12}^{(n)}[0] = q_0\sqrt{n}, q_{34}^{(n)}[0] = q_0'\sqrt{n}, q_{23}^{(n)}[0] \leq \hat{M}\}$, where q_0, q_0' and $\hat{M}$ are arbitrary positive number. The initial condition is the most likely initial state for the network if the resources are used "ideally": links 2 and 3 are underloaded so that their queues are small and links 1 and 2 (and 3 and 4) behave like a single-server queue in heavy traffic. We first state a number of lemmas and then state our final result in Proposition 2.8.7. In the proof of Proposition 2.5.3, one crucial step was to bound the probabilities $P(\tilde{q}_{12}^{(n)}[\infty] < B)$, $P(\tilde{q}_{34}^{(n)}[\infty] < B)$, and $P(q_{23}^{(n)}[\infty] \geq B)$. But since here the initial queueing system is not in steady state, we need to bound these transient probabilities at each time slot k, i.e., $P(\tilde{q}_{12}^{(n)}[k] < B)$, $P(\tilde{q}_{34}^{(n)}[k] < B)$, and $P(q_{23}^{(n)}[k] \geq B)$.

As before, the state transition diagrams of $\tilde{q}_{12}^{(n)}$, $\tilde{q}_{34}^{(n)}$, and $q_{23}^{(n)}$ are shown in Figure 2.6 with different values of α, γ, and $\beta = 1 - \alpha - \gamma$. We consider the transient behavior of queue $\tilde{q}_{12}^{(n)}$ first, where $\alpha = \lambda_1^{(n)}\lambda_2^{(n)}$ and $\gamma = (1 - \lambda_1^{(n)})(1 - \lambda_2^{(n)})$.

The work in [40] gives an expression for the transient probability distribution of a discrete-time queue conditioned on the initial state, with the state transition diagram as shown in Figure 2.6; i.e.,

$$P(\tilde{q}_{12}[k] = h|\tilde{q}_{12}[0] = p) = \nu^{-p-1}f(k, h+p+1) + f(k, h-p) + (1-\nu)\nu^h \sum_{j=h+p+2}^{k} f(k, -j),$$

where $\nu = \alpha/\gamma = \frac{\lambda_1^{(n)}\lambda_2^{(n)}}{(1-\lambda_1^{(n)})(1-\lambda_2^{(n)})} < 1$, and $f(k, h)$ is the probability distribution of a random walk which starts at the origin (at time 0) and ends at coordinate h at time slot k. According to the random walk, at each time slot the walker moves one step to the right with probability α, moves one step to the left with probability γ, and stays in the current position with probability β. In the random walk, a denotes the number of right steps, b denotes the number of slots that the walker does not move, and c denotes the number of left steps. Let $\mathcal{D} = \{a, b, c \in \mathbb{N} | \text{s.t. } a+b+c = k, a-c = h\}$, and $C(\theta) = \beta + 2\sqrt{\alpha\gamma}\cos\theta$. From [40], we get the following expression for the random walk distribution:

$$f(k,h) = \sum_{a,b,c\in\mathcal{D}} \Big(\begin{array}{c} k \\ a, b, c \end{array} \Big) \alpha^a \beta^b \gamma^c = \frac{\nu^{h/2}}{\pi} \int_0^\pi (\cos h\theta)[C(\theta)]^k d\theta, \text{ if } |h| \le k,$$

and $f(k,h) = 0$, if $|h| > k$.

To find expressions for α and γ as a function of n, recall that $\lambda_l^{(n)} = \lambda_l - \frac{m_l}{\sqrt{n}}$; the transition probabilities α and γ of queue $q_{12}^{(n)}$ are given by

$$\begin{aligned} \alpha &= \lambda_1^{(n)}\lambda_2^{(n)} = \lambda_1\lambda_2\big[1 - \frac{m_1}{\sqrt{n}\lambda_1}\big]\big[1 - \frac{m_2}{\sqrt{n}\lambda_2}\big], \\ \gamma &= (1-\lambda_1^{(n)})(1-\lambda_2^{(n)}) = \lambda_1\lambda_2\big[1 + \frac{m_1}{\sqrt{n}\lambda_2}\big]\big[1 + \frac{m_2}{\sqrt{n}\lambda_1}\big]. \end{aligned}$$

Next, we state some facts along with their proofs that will be used in the proofs of later lemmas.

Fact (i). For any positive and arbitrary small ϵ_1, there exists a positive number $\hat{N}$, so that if $n > \hat{N}$, we have $|\alpha - \lambda_1\lambda_2| \le \epsilon_1$, $|\gamma - \lambda_1\lambda_2| \le \epsilon_1$, $|\sqrt{\alpha\gamma} - \lambda_1\lambda_2| \le \epsilon_1$, and $\nu \ge 1 - \dfrac{m_1 + m_2}{\lambda_1\lambda_2\sqrt{n}}$.

Proof. From the expressions of α and γ, we know the limits of α and γ are $\lambda_1\lambda_2$ as n goes to ∞. Hence for any positive and arbitrary small ϵ_1, there exists a positive number $\hat{N}$, so that if $n > \hat{N}$, we have $|\alpha - \lambda_1\lambda_2| \le \epsilon_1$, $|\gamma - \lambda_1\lambda_2| \le \epsilon_1$, and $|\sqrt{\alpha\gamma} - \lambda_1\lambda_2| \le \epsilon_1$. From (2.2), we directly get $\nu \ge 1 - \dfrac{m_1 + m_2}{\lambda_1\lambda_2\sqrt{n}}$. □

Fact (ii). For any $0 \le \theta \le \pi$, we have $0 \le C(\theta) \le 1$; especially if $0 \le \theta \le \pi/3$ and n is sufficiently large so that *Fact (i)* is true, we have $C(\theta) \ge 3(\lambda_1\lambda_2 - \epsilon_1)$.

Proof. Noting that $C(\theta)$ is a decreasing function if θ is between 0 and π, and using the

Cauchy-Schwarz inequality, we have

$$\begin{aligned} C(\theta) &\geq C(\pi) \\ &= 1-\alpha-\gamma-2\sqrt{\alpha\gamma} \\ &= 1-\left(\sqrt{\lambda_1^{(n)}\lambda_2^{(n)}}+\sqrt{(1-\lambda_1^{(n)})(1-\lambda_2^{(n)})}\right)^2 \\ &\geq 1-\left(\lambda_1^{(n)}+(1-\lambda_1^{(n)})\right)\left(\lambda_2^{(n)}+(1-\lambda_2^{(n)})\right) \\ &= 0 \end{aligned}$$

and

$$C(\theta) \leq C(0) = 1-\alpha-\gamma+2\sqrt{\alpha\gamma} = 1-(\sqrt{\alpha}-\sqrt{\gamma})^2 \leq 1.$$

Especially if $0 \leq \theta \leq \pi/3$ and n is sufficiently large so that *Fact (i)* is satisfied, we have

$$C(\theta) \geq C(\pi/3) = C(\pi)+3\sqrt{\alpha\gamma} \geq 3\sqrt{\alpha\gamma} \geq 3(\lambda_1\lambda_2-\epsilon_1).$$

□

Fact (iii). $(1-x)^B \geq 1-xB,$ for $0 \leq x < 1,\ B > 1.$

Proof. Let $g(x) = (1-x)^B - 1 + xB$. Calculate the first derivative as

$$g'(x) = B\big(1-(1-x)^{B-1}\big).$$

Under the condition $0 \leq x < 1,\ B > 1$, we know $g(0) = 0$ and $g'(x) \geq 0$ with equality if and only if $x = 0$. Thus $g(x) = (1-x)^B - 1 + xB \geq 0$. □

Fact (iv). $1-\cos\theta \geq \theta^2/4$, for $0 \leq \theta \leq \pi/2$.

Proof. Let $g_1(\theta) = 1-\cos\theta-\theta^2/4$. The first order derivative is $g_1'(\theta) = \sin\theta-\theta/2 \geq 0$, for $0 \leq \theta \leq \pi/2$, with equality if and only if $\theta = 0$. Noting that $g_1(0) = 0$, the inequality holds. □

Now let us bound the probability $P(\tilde{q}_{12}^{(n)}[k] = h|\tilde{q}_{12}^{(n)}[0] = p)$ under the condition that the initial queue length $p = \lfloor q_0\sqrt{n}\rfloor \gg h$, and k satisfies $\lfloor n^{0.46}t\rfloor + 1 \le k \le \lfloor nt\rfloor$. In order to bound $P(\tilde{q}_{12}^{(n)}[k] = h|\tilde{q}_{12}^{(n)}[0] = p)$, we need to bound each of the terms in (2.4) separately. In Lemma 2.8.1, we will bound the second term, i.e., $f(k, h-p)$.

Lemma 2.8.1. *Assume $p = \lfloor q_0\sqrt{n}\rfloor \gg h$, and $\lfloor n^{0.46}t\rfloor + 1 \le k \le \lfloor nt\rfloor$, where q_0 and t are arbitrary positive numbers. There exists positive number N_1, so that if $n > N_1$, $f(k, h-p) \le M_1 \cdot n^{-0.29}$ for some positive number M_1 which is independent of h.*

Proof. We will consider two cases. For the first case, k is between $\lfloor n^{0.46}t\rfloor + 1$ and $\lfloor n^{0.6}t\rfloor$. For the second case, k is between $\lfloor n^{0.6}t\rfloor + 1$ and $\lfloor nt\rfloor$. Then we will combine these two cases and complete the lemma. Note that throughout the proof we assume that n is sufficiently large (at least greater than $\hat{N}$) so that *Facts (i)-(ii)* are true.

We bound $f(k, h-p)$ using the integral in (2.4), i.e., if $|h-p| \le k$,

$$f(k, h-p) = \frac{\nu^{\frac{h-p}{2}}}{\pi}\int_0^\pi \cos[(h-p)\theta][C(\theta)]^k d\theta,$$

and if $|h-p| > k$, $f(k, h-p) = 0$, which is always bounded by any positive number. So we do not care about this case. First let us bound the coefficient $\nu^{\frac{h-p}{2}}$. Using *Fact (i)*, $\nu < 1$, and $p = \lfloor q_0\sqrt{n}\rfloor$ we have

$$\nu^{\frac{h-p}{2}} \le \nu^{-p/2} \le \big(1 - \frac{m_1 + m_2}{\lambda_1\lambda_2\sqrt{n}}\big)^{-q_0\sqrt{n}/2}$$

and

$$\lim_{n\to\infty} \nu^{\frac{h-p}{2}} \le \lim_{n\to\infty}\big(1 - \frac{m_1 + m_2}{\lambda_1\lambda_2\sqrt{n}}\big)^{-q_0\sqrt{n}/2} = \exp\Big[-\frac{q_0(m_1 + m_2)}{2\lambda_1\lambda_2}\Big].$$

This means that for some fixed number $M > \exp\big[-\frac{q_0(m_1+m_2)}{2\lambda_1\lambda_2}\big]$, there exists a positive number N_{10}, so that if $n > N_{10}$, we have $\nu^{\frac{h-p}{2}} \le M$.

Case 1. Consider $\lfloor n^{0.46}t\rfloor + 1 \le k \le \lfloor n^{0.6}t\rfloor$.

Let $\phi_1 = n^{-0.2}$. We would like to bound the integrals $\int_{\phi_1}^{\pi}$ and $\int_0^{\phi_1}$ separately first.

Step 1. Bound $\dfrac{\nu^{\frac{h-p}{2}}}{\pi}\displaystyle\int_{\phi_1}^{\pi}\cos[(h-p)\theta][C(\theta)]^k d\theta$.
By *Facts (i), (ii), (iv)*, if θ satisfies $\phi_1 \le \theta \le \pi$, we have

$$\begin{aligned} C(\theta) \le C(\phi_1) &= C(0) - 2\sqrt{\alpha\gamma}(1-\cos\phi_1) \le 1 - 2(\lambda_1\lambda_2 - \epsilon_1)(1-\cos\phi_1) \\ &\le 1 - 2(\lambda_1\lambda_2-\epsilon_1){\phi_1}^2/4 = 1 - \frac{\lambda_1\lambda_2-\epsilon_1}{2n^{0.4}}. \end{aligned} \tag{2.4}$$

Because $k \ge n^{0.46}t$ and (2.4), for any small positive ϵ_2, there exists a positive number N_{11}, so that if $n > N_{11}$, we have the following inequality by using (2.4):

$$\begin{aligned} \frac{1}{\pi}\int_{\phi_1}^{\pi}\cos[(h-p)\theta][C(\theta)]^k d\theta &\le \frac{1}{\pi}\int_{\phi_1}^{\pi}[C(\theta)]^k d\theta \le [C(\phi_1)]^k \\ &\le \Big[\big(1-\frac{\lambda_1\lambda_2-\epsilon_1}{2n^{0.4}}\big)^{n^{0.4}}\Big]^{n^{0.06}t} \le \big[e^{-(\lambda_1\lambda_2-\epsilon_1)/2}+\epsilon_2\big]^{n^{0.06}t}. \end{aligned}$$

In summary, if $n > \max(N_{10}, N_{11})$, we have

$$\frac{\nu^{\frac{h-p}{2}}}{\pi}\int_{\phi_1}^{\pi}\cos[(h-p)\theta][C(\theta)]^k d\theta \le M\cdot\big[e^{-\frac{\lambda_1\lambda_2-\epsilon_1}{2}}+\epsilon_2\big]^{n^{0.06}t}.$$

Notice that $\dfrac{\nu^{\frac{h-p}{2}}}{\pi}\displaystyle\int_{\phi_1}^{\pi}\cos[(h-p)\theta][C(\theta)]^k d\theta$ goes to zero exponentially.
Step 2. Bound $\dfrac{\nu^{\frac{h-p}{2}}}{\pi}\displaystyle\int_{0}^{\phi_1}\cos[(h-p)\theta][C(\theta)]^k d\theta$.
Noting that $h \ll p$, we have

$$p - h = \lfloor q_0\sqrt{n}\rfloor - h \ge q_0\sqrt{n}/2.$$

Let $N_0 = \dfrac{2\pi}{p-h} = \dfrac{2\pi}{\lfloor q_0\sqrt{n}\rfloor - h} \le \dfrac{4\pi}{q_0\sqrt{n}}$, and $\theta_j = N_0 j$; we can bound the following integral:

$$\frac{1}{\pi}\int_0^{\phi_1}\cos[(h-p)\theta][C(\theta)]^k d\theta \le \Big|\frac{1}{\pi}\sum_{j=0}^{\lfloor\frac{\phi_1}{N_0}\rfloor-1}\int_{\theta_j}^{\theta_{j+1}}\cos(\frac{2\pi\theta}{N_0})[C(\theta)]^k d\theta\Big| + \frac{N_0}{\pi}.$$

Now we consider θ between θ_j and θ_{j+1}, i.e., $\theta = \theta_j + N_0 x$, where $0 \le x \le 1$. Because $C(\theta)$

is a decreasing function when θ is between 0 and ϕ_1, the difference between $C(\theta)$ and $C(\theta_j)$ is bounded as follows:

$$\begin{aligned}
0 &\geq C(\theta) - C(\theta_j) \\
&= 2\sqrt{\alpha\gamma}[\cos(\theta_j + N_0 x) - \cos\theta_j] \\
&\geq 2\sqrt{\alpha\gamma}[\cos(\theta_j + N_0) - \cos\theta_j] \\
&= 2\sqrt{\alpha\gamma}[-2\sin(\theta_j + N_0/2)\sin(N_0/2)] \\
&\geq 2\sqrt{\alpha\gamma}[-2(\theta_j + N_0/2)(N_0/2)] \\
&\geq -2(\lambda_1\lambda_2 + \epsilon_1)\phi_1 N_0. \qquad (2.5)
\end{aligned}$$

In (2.5), we used the fact that θ_j is small to go from line 4 to line 5, and *Fact (i)* to go from line 5 to line 6. Now we calculate $[C(\theta)]^k/[C(\theta_j)]^k$ using *Facts (ii) and (iii)* when $\theta_j \leq \theta \leq \theta_{j+1}$. Noting that $\lfloor n^{0.46}t \rfloor + 1 \leq k \leq \lfloor n^{0.6}t \rfloor$, $\epsilon_1 \ll \lambda_1\lambda_2$, and $C(\theta) > C(\pi/3)$ for $0 \leq \theta \leq \phi_1$, we have

$$\begin{aligned}
1 \geq \frac{[C(\theta)]^k}{[C(\theta_j)]^k} &= \left[1 + \frac{C(\theta) - C(\theta_j)}{C(\theta_j)}\right]^k \\
&\geq \left[1 + \frac{C(\theta) - C(\theta_j)}{C(\pi/3)}\right]^k \\
&\geq \left[1 - \frac{2(\lambda_1\lambda_2 + \epsilon_1)\phi_1 N_0}{3(\lambda_1\lambda_2 - \epsilon_1)}\right]^k \\
&\geq 1 - \frac{2(\lambda_1\lambda_2 + \epsilon_1)\phi_1 N_0}{3(\lambda_1\lambda_2 - \epsilon_1)}k \\
&\geq 1 - \frac{2(\lambda_1\lambda_2 + \epsilon_1)\ N_0 n^{0.4} t}{3(\lambda_1\lambda_2 - \epsilon_1)} \\
&\geq 1 - N_0 n^{0.4} t. \qquad (2.6)
\end{aligned}$$

From line 1 to line 2, we used the fact that $C(\theta)$ is positive and also a decreasing function when θ is between 0 to $\pi/3$. From line 2 to line 3, we used (2.5) and *Fact (ii)*. We used *Fact (iii)* to go from line 3 to line 4. By substituting for $\phi_1 = n^{-0.2}$ and considering $k \leq n^{0.6}t$, we go from line 4 to line 5. Because $[C(\theta_j)]^k \leq 1$, $\forall\ j$, we have

$$
\begin{aligned}
& \Big|\int_{\theta_j}^{\theta_{j+1}} \cos(\frac{2\pi\theta}{N_0})[C(\theta)]^k d\theta\Big| \\
= & \Big|\int_{\theta_j}^{\theta_{j+1}} \cos(\frac{2\pi\theta}{N_0})[C(\theta_j)]^k d\theta + \int_{\theta_j}^{\theta_{j+1}} \cos(\frac{2\pi\theta}{N_0})\Big[[C(\theta)]^k - [C(\theta_j)]^k\Big] d\theta\Big| \\
= & \Big|\ 0 + \int_{\theta_j}^{\theta_{j+1}} \cos(\frac{2\pi\theta}{N_0})\Big[[C(\theta)]^k - [C(\theta_j)]^k\Big] d\theta\Big| \\
\le & \Big|\int_{\theta_j}^{\theta_{j+1}} \cos(\frac{2\pi\theta}{N_0})\Big[\frac{[C(\theta)]^k}{[C(\theta_j)]^k} - 1\Big] d\theta\Big| \\
\le & \Big|\int_{\theta_j}^{\theta_{j+1}} \Big[\frac{[C(\theta)]^k}{[C(\theta_j)]^k} - 1\Big] d\theta\Big| \\
\le & \Big|\int_{\theta_j}^{\theta_{j+1}} N_0 n^{0.4} t \cdot d\theta\Big| \\
= & N_0^2 n^{0.4} t.
\end{aligned} \tag{2.7}
$$

In (2.7), we used (2.6) to go from line 5 to line 6. From (2.5) and (2.7), by substituting for $\phi_1 = n^{-0.2}$ and $N_0 \le \frac{4\pi}{q_0\sqrt{n}}$, we have

$$
\begin{aligned}
& \frac{1}{\pi}\int_0^{\phi_1} \cos[(h-p)\theta][C(\theta)]^k d\theta \\
\le & \frac{1}{\pi}\sum_{j=0}^{\lfloor \frac{\phi_1}{N_0}\rfloor - 1} \Big|\int_{\theta_j}^{\theta_{j+1}} \cos(\frac{2\pi\theta}{N_0})[C(\theta)]^k d\theta\Big| + N_0/\pi \\
\le & \frac{1}{\pi}\ N_0^2 n^{0.4} t \lfloor \frac{\phi_1}{N_0} \rfloor + N_0/\pi \le \frac{1}{\pi}\ N_0 n^{0.4} t \phi_1 + N_0/\pi \\
\le & \frac{1}{\pi}\ N_0 n^{0.2} t + N_0/\pi \le 4(n^{-0.3} t + n^{-0.5})/q_0.
\end{aligned}
$$

Combining the two steps, we can say that there exist positive numbers M_{11} and N_{12} so that if $n > N_{12}$ we have

$$\begin{aligned} & \frac{\nu^{\frac{h-p}{2}}}{\pi} \int_0^{\pi} \cos[(h-p)\theta][C(\theta)]^k d\theta \\ \leq \quad & M \cdot \left[e^{-(\lambda_1\lambda_2-\epsilon_1)/2} + \epsilon_2\right]^{n^{0.06}t} + M \cdot 4(n^{-0.3}t + n^{-0.5})/q_0 \\ \leq \quad & M_{11} \cdot n^{-0.3}. \end{aligned}$$

Notice that N_{12} needs to satisfy $N_{12} > \max(N_{10}, N_{11})$. Also notice that for large n, the rest of the terms go to zero much faster than the term $4Mtn^{-0.3}/q_0$ in the previous expression.

Case 2. Consider $\lfloor n^{0.6}t \rfloor + 1 \leq k \leq \lfloor nt \rfloor$.

Let $\phi_2 = n^{-0.29}$. As in the proof of case 1, we try to bound $f(k, h-p)$ using the integral expression. We would like to bound the integrals $\int_0^{\phi_2}$ and $\int_{\phi_2}^{\pi}$ separately first.

Step 1. Bound $\frac{\nu^{\frac{h-p}{2}}}{\pi} \int_0^{\phi_2} \cos[(h-p)\theta][C(\theta)]^k d\theta$.

$$\frac{1}{\pi} \int_0^{\phi_2} \cos[(h-p)\theta][C(\theta)]^k d\theta \leq \frac{1}{\pi} \int_0^{\phi_2} 1 \cdot d\theta = n^{-0.29}/\pi.$$

If $n > N_{10}$, we have

$$\frac{\nu^{\frac{h-p}{2}}}{\pi} \int_0^{\phi_2} \cos[(h-p)\theta][C(\theta)]^k d\theta \leq M \cdot n^{-0.29}/\pi,$$

because $\nu^{\frac{h-p}{2}} \leq M$ for large $n > N_{10}$.

Step 2. Bound $\frac{\nu^{\frac{h-p}{2}}}{\pi} \int_{\phi_2}^{\pi} \cos[(h-p)\theta][C(\theta)]^k d\theta$.

By *Facts (i), (ii), (iv)*, if θ satisfies $\phi_2 \leq \theta \leq \pi$, we have

$$\begin{aligned} C(\theta) \quad & \leq \quad C(\phi_2) = C(0) - 2\sqrt{\alpha\gamma}(1 - \cos\phi_2) \\ & \leq \quad 1 - 2(\lambda_1\lambda_2 - \epsilon_1)(1 - \cos\phi_2) \leq 1 - (\lambda_1\lambda_2 - \epsilon_1)\phi_2^2/2 \\ & = \quad 1 - \frac{\lambda_1\lambda_2 - \epsilon_1}{2n^{0.58}}. \end{aligned} \tag{2.8}$$

Because $k \geq n^{0.6}t$ and from (2.8), for any small positive ϵ_3, there exists a positive number

N_{13}, so that if $n > N_{13}$, we have

$$\begin{aligned} \frac{1}{\pi}\int_{\phi_2}^{\pi}\cos[(h-p)\theta][C(\theta)]^k d\theta &\leq \frac{1}{\pi}\int_{\phi_2}^{\pi}[C(\theta)]^k d\theta \\ &\leq [C(\phi_2)]^k \leq \Big[\big(1-\frac{\lambda_1\lambda_2-\epsilon_1}{2n^{0.58}}\big)^{n^{0.58}}\Big]^{n^{0.02}t} \\ &\leq \Big[e^{-(\lambda_1\lambda_2-\epsilon_1)/2}+\epsilon_3\Big]^{n^{0.02}t}. \end{aligned}$$

In summary, if $n \leq \max(N_{10}, N_{13})$, we have

$$\frac{\nu^{\frac{h-p}{2}}}{\pi}\int_{\phi_2}^{\pi}[\cos(h-p)\theta][C(\theta)]^k d\theta \leq M\cdot\big[e^{-(\lambda_1\lambda_2-\epsilon_1)/2}+\epsilon_3\big]^{n^{0.02}t}.$$

Combining the two steps, we can say that there exists positive number M_{12} and N_{14}, so that if $n > N_{14}$ we have

$$\begin{aligned} \frac{\nu^{\frac{h-p}{2}}}{\pi}\int_{0}^{\pi}\cos[(h-p)\theta][C(\theta)]^k d\theta &\leq M\cdot\big[e^{-(\lambda_1\lambda_2-\epsilon_1)/2}+\epsilon_2\big]^{n^{0.02}} + M\cdot n^{-0.29}/\pi \\ &\leq M_{12}\cdot n^{-0.29}. \end{aligned}$$

Notice that N_{14} needs to satisfy $N_{14} \geq \max(N_{10}, N_{13})$. Also notice that for large n, the rest of the terms go to zero much faster than the term $Mn^{-0.29}/\pi$ in the previous expression.

Combining two cases and choosing $M_1 = \max(M_{11}, M_{12})$ and $N_1 = \max(N_{12}, N_{14})$, we get our result. □

So far, we have bounded the second term $f(k, h-p)$ in (2.4); next, we try to bound the first term.

Lemma 2.8.2. *Assume that $p = \lfloor q_0\sqrt{n}\rfloor \gg h$, and $\lfloor n^{0.46}t\rfloor + 1 \leq k \leq \lfloor nt\rfloor$, where q_0 and t are arbitrary positive numbers. There exists positive number N_2, so that if $n > N_2$, we have $\nu^{-p-1}f(k, h+p+1) \leq M_2\cdot n^{-0.29}$ for some positive number M_2 which is independent of h.*

Proof.

$$\nu^{-p-1}f(k,h+p+1) = \frac{\nu^{\frac{h-p-1}{2}}}{\pi}\int_0^{\pi}\cos[(h+p+1)\theta][C(\theta)]^k d\theta$$

We can prove this lemma using similar ideas as in Lemma 2.8.1. More specifically, we can bound $\nu^{\frac{h-p-1}{2}}$ instead of $\nu^{\frac{h-p}{2}}$ in Lemma 2.8.1 and then we can consider $\cos[(h+p+1)\theta]$ instead of $\cos[(h-p)\theta]$. □

Next we focus on bounding the third term, i.e., the term $(1-\nu)\nu^h \sum_{j=h+p+2}^{k} f(k,-j)$.

Lemma 2.8.3. *Assume that $p = \lfloor q_0\sqrt{n}\rfloor \gg h$. There exists positive number N_3, so that if $n > N_3$, we have $(1-\nu)\nu^h \sum_{j=h+p+2}^{k} f(k,-j) \le M_3 \cdot n^{-0.5}$, $\forall k$ for some positive number M_3 which is independent of h.*

Proof. Because $\nu \le 1$ and $\sum_{j=h+p+2}^{k} f(k,-j) \le 1$, and by using *Fact (i)*, we have

$$\begin{aligned}(1-\nu)\nu^h \sum_{j=h+p+2}^{k} f(k,-j) &\le (1-\nu)\nu^h \le 1-\nu \\ &\le \frac{1}{\sqrt{n}}\left(\frac{m_1+m_2}{\lambda_1\lambda_2}+\epsilon_1\right) \le M_3 \cdot n^{-0.5}.\end{aligned}$$

□

We have bounded all three terms in (2.4); in Lemma 2.8.4, we show the upper bound of the probability $P(\tilde{q}_{12}^{(n)}[k] = h|\tilde{q}_{12}^{(n)}[0] = p)$.

Lemma 2.8.4. *Assume that $p = \lfloor q_0\sqrt{n}\rfloor \gg h$, and $\lfloor n^{0.46}t\rfloor + 1 \le k \le \lfloor nt\rfloor$, where q_0 and t are arbitrary positive numbers. There exists positive number $\bar{N}$, if $n > \bar{N}$, $P(\tilde{q}_{12}^{(n)}[k] = h|\tilde{q}_{12}^{(n)}[0] = p) \le M_4 n^{-0.29}$ for some positive M_4 which are independent of h.*

Proof. Let $M_4 = 3\cdot\max(M_1, M_2, M_3)$ and $\bar{N} = \max(N_1, N_2, N_3)$. Considering Lemmas 2.8.1, 2.8.2, 2.8.3 and (2.4), we know that there exists a constant M_4, so that for any sufficiently large $n > \bar{N}$, we have $P(\tilde{q}_{12}^{(n)}[k] = h|\tilde{q}_{12}^{(n)}[0] = p) \le M_4 n^{-0.29}$. □

Lemma 2.8.5. *Assume that $p = \lfloor q_0\sqrt{n}\rfloor$, $B = \lfloor n^{0.02}\rfloor$, and $\lfloor n^{0.46}t\rfloor + 1 \le k \le \lfloor nt\rfloor$. We have $P(\tilde{q}_{12}^{(n)}[k] < B|\tilde{q}_{12}^{(n)}[0] = p) \le M_4 n^{-0.27}$ for some positive number M_4 and $n > \bar{N}$.*

Proof. Because $B \ll p$ and due to Lemma 2.8.4, we have

$$\begin{aligned} P(\tilde{q}_{12}^{(n)}[k] < B|\tilde{q}_{12}^{(n)}[0] = p) &= \sum_{h=0}^{B-1} P(\tilde{q}_{12}^{(n)}[k] = h|\tilde{q}_{12}^{(n)}[0] = p) \\ &\leq BM_4 n^{-0.29} \leq M_4 n^{-0.27}. \end{aligned}$$

□

A similar result applies for queue $\tilde{q}_{34}^{(n)}$, for any sufficiently large $n > \bar{N}$, with parameters q_0' and M_5 instead, i.e.,

$$P(\tilde{q}_{34}^{(n)}[k] < B|\tilde{q}_{34}^{(n)}[0] = q_0'\sqrt{n}) \leq M_5 n^{-0.27}. \tag{2.9}$$

So far, we have bounded the transient probabilities of $\tilde{q}_{12}^{(n)}$ and $\tilde{q}_{34}^{(n)}$. Now we try to bound the transient probability of the queueing system $q_{23}^{(n)}$ with transition diagram as shown in Figure 2.6; to distinguish from $\tilde{q}_{12}^{(n)}$ and $\tilde{q}_{34}^{(n)}$ we use $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\gamma}$ instead of α, β and γ, where $\hat{\alpha} = \lambda_2^{(n)}\lambda_3^{(n)}$, $\hat{\gamma} = (1-\lambda_2^{(n)})(1-\lambda_3^{(n)})$, and $\hat{\beta} = 1 - \hat{\alpha} - \hat{\gamma}$. We also know that $\lambda_2^{(n)} = \lambda_2 - \dfrac{m_2}{\sqrt{n}}$, $\lambda_3^{(n)} = \lambda_3 - \dfrac{m_3}{\sqrt{n}}$ and $\lambda_2 + \lambda_3 < 1$. Let $\hat{C}(\theta) = \hat{\beta} + 2\sqrt{\hat{\alpha}\hat{\gamma}}\cos\theta$.

The transient probability distribution of $q_{23}^{(n)}$ is given by

$$\begin{aligned} &P(q_{23}^{(n)}[k] = h|q_{23}^{(n)}[0] = p) = \hat{\nu}^{-p-1}\hat{f}(k, h+p+1) \\ &+\hat{f}(k, h-p) + (1-\hat{\nu})\hat{\nu}^h \sum_{j=h+p+2}^{k} \hat{f}(k, -j), \end{aligned} \tag{2.10}$$

where $\hat{f}(k,h) = \dfrac{\hat{\nu}^{h/2}}{\pi}\displaystyle\int_0^{\pi}(\cos h\theta)[\hat{C}(\theta)]^k d\theta$, if $k \geq |h|$, and $\hat{f}(k,h) = 0$, if $k < |h|$, and $\hat{\nu} = \hat{\alpha}/\hat{\gamma} < 1$.

Lemma 2.8.6. *Assume that* $p \ll B$; *for any* k, *we can bound the probability that* $P(q_{23}^{(n)}[k] \geq B|q_{23}^{(n)}[0] = p) \leq \dfrac{3}{1-\sqrt{\hat{\nu}}}\hat{\nu}^{B/4}$.

Proof. The first step is to bound $P(q_{23}^{(n)}[k] = h|q_{23}^{(n)}[0] = p)$, where $p \ll h$ using (2.10).

Therefore we bound three terms in (2.10) one by one as follows:

$$\begin{aligned}
\hat{\nu}^{-p-1}\hat{f}(k, h+p+1) &= \frac{\hat{\nu}^{\frac{h-p-1}{2}}}{\pi}\int_0^{\pi}(\cos h\theta)(\hat{\beta}+2\sqrt{\hat{\alpha}\hat{\gamma}}\cos\theta)^k d\theta \\
&\leq \frac{\hat{\nu}^{\frac{h-p-1}{2}}}{\pi}\int_0^{\pi} 1\cdot d\theta = \hat{\nu}^{\frac{h-p-1}{2}},
\end{aligned}$$

$$\hat{f}(k, h-p) = \frac{\hat{\nu}^{\frac{h-p}{2}}}{\pi}\int_0^{\pi}(\cos h\theta)[\hat{C}(\theta)]^k d\theta \leq \frac{\hat{\nu}^{\frac{h-p}{2}}}{\pi}\int_0^{\pi} 1\cdot d\theta = \hat{\nu}^{\frac{h-p}{2}},$$

$$(1-\hat{\nu})\hat{\nu}^h \sum_{j=h+p+2}^{k} \hat{f}(k,-j) \leq (1-\hat{\nu})\hat{\nu}^h.$$

Because $\hat{\nu}^{\frac{h-p-1}{2}} \geq \hat{\nu}^{\frac{h-p}{2}}$ and $\hat{\nu}^{\frac{h-p-1}{2}} \geq (1-\hat{\nu})\hat{\nu}^h$, for $p \ll h$ and $\hat{\nu} < 1$, we have

$$\begin{aligned}
P(q_{23}^{(n)}[k] \geq B | q_{23}^{(n)}[0] = p) &= \sum_{h=B}^{\infty} P(q_{23}^{(n)}[k] = h | q_{23}^{(n)}[0] = p) \\
&\leq \sum_{h=B}^{\infty}[\hat{\nu}^{\frac{h-p}{2}} + \hat{\nu}^{\frac{h-p-1}{2}} + (1-\hat{\nu})\hat{\nu}^h] \leq \sum_{h=B}^{\infty} 3\hat{\nu}^{\frac{h-p-1}{2}} \\
&\leq \frac{3}{1-\sqrt{\hat{\nu}}}\hat{\nu}^{\frac{B-p-1}{2}} \leq \frac{3}{1-\sqrt{\hat{\nu}}}\hat{\nu}^{B/4}.
\end{aligned}$$

□

Having established Lemmas 2.8.4, 2.8.5 and 2.8.6, we can prove Proposition 2.8.7.

Proposition 2.8.7. *If the initial conditions of the* n^{th} *network are according to* $\mathcal{D}_1 := \{q_{12}^{(n)}[0] = q_0\sqrt{n}, q_{34}^{(n)}[0] = q_0'\sqrt{n}, q_{23}^{(n)}[0] \leq \hat{M}\}$*, where* q_0*,* q_0' *and* $\hat{M}$ *are arbitrary positive number, then our algorithm is sample-path optimal in heavy traffic; i.e., for any* $c > 0$ *and any fixed* $T > 0$*,*

$$\lim_{n\to\infty} P\Big(\sup_{t\in[0,T]} \frac{\sum_l q_l^{(n)}[\lfloor nt \rfloor] - \sum_l \tilde{q}_l^{(n)}[\lfloor nt \rfloor]}{\sqrt{n}} \geq c\Big) = 0.$$

Proof. Let $t_1 = \lfloor n^{0.46} t \rfloor$ and $B = \lfloor n^{0.02} \rfloor \gg \hat{M}$. From Lemma 2.8.5 and (2.9), if $n > \bar{N}$, we have

$$\begin{aligned}
& \sum_{r=0}^{\lfloor nt \rfloor -1} \frac{P(\tilde{q}_{12}^{(n)}[r] < B|\mathcal{D}_1) P(\tilde{q}_{34}^{(n)}[r] < B|\mathcal{D}_1)}{c\sqrt{n}/2} \\
\leq \ & \sum_{r=t_1+1}^{\lfloor nt \rfloor -1} \frac{P(\tilde{q}_{12}^{(n)}[r] < B|\mathcal{D}_1) P(\tilde{q}_{34}^{(n)}[r] < B|\mathcal{D}_1)}{c\sqrt{n}/2} + \frac{2(t_1+1)}{c\sqrt{n}} \\
\leq \ & \frac{n^{0.46}t + 1 + (nt - n^{0.46}t)[M_4 n^{-0.27}][M_5 n^{-0.27}]}{c\sqrt{n}/2} \\
\leq \ & \frac{2t}{c}(2 + M_4 M_5) n^{-0.04}.
\end{aligned}$$

We went from line 1 to lines 2 and 3 by dividing the sum into two parts and bounding the second part. From lines 2 and 3 to line 4, we used Lemma 2.8.5 and (2.9). From Lemma 2.8.6, we have

$$\sum_{r=0}^{\lfloor nt \rfloor -1} \frac{P(q_{23}^{(n)}[r] \geq B|\mathcal{D}_1)}{c\sqrt{n}/2} \leq 2nt\big[\frac{3}{1-\sqrt{\hat{\nu}}}\hat{\nu}^{B/4}\big]/c\sqrt{n} \leq \frac{6t\sqrt{n}}{(1-\sqrt{\hat{\nu}})c}\hat{\nu}^{n^{0.02}/4}.$$

Hence, using the Markov inequality and the previous expressions, we get

$$\begin{aligned}
& P\Big(\sup_{t\in[0,T]} \frac{q_{12}^{(n)}[\lfloor nt \rfloor] - \tilde{q}_{12}^{(n)}[\lfloor nt \rfloor]}{\sqrt{n}} \geq \frac{c}{2}\Big) \\
\leq \ & P(\sup_{t\in[0,T]} \sum_{r=0}^{\lfloor nt \rfloor -1} \hat{i}^{(n)}[r] \geq \frac{c\sqrt{n}}{2}) = P(\sum_{r=0}^{\lfloor nT \rfloor -1} \hat{i}^{(n)}[r] \geq \frac{c\sqrt{n}}{2}) \\
\leq \ & \frac{2}{c\sqrt{n}} \sum_{r=0}^{\lfloor nT \rfloor -1} E(\hat{i}^{(n)}[r]) \\
\leq \ & \sum_{r=0}^{\lfloor nT \rfloor -1} \frac{P(\tilde{q}_{12}^{(n)}[r] < B|\mathcal{D}_1) P(\tilde{q}_{34}^{(n)}[r] < B|\mathcal{D}_1)}{c\sqrt{n}/2} + \sum_{r=0}^{\lfloor nT \rfloor -1} \frac{2P(q_{23}^{(n)}[r] \geq B|\mathcal{D}_1)}{c\sqrt{n}/2} \\
\leq \ & \frac{2T}{c}(2 + M_4 M_5) n^{-0.04} + \frac{12T\sqrt{n}}{(1-\sqrt{\hat{\nu}})c}\hat{\nu}^{n^{0.02}/4}.
\end{aligned}$$

Thus, if the initial queue lengths are chosen according to $\mathcal{D}_1$, we have

$$\lim_{n\to\infty} P\Big(\sup_{t\in[0,T]} \frac{q_{12}^{(n)}[\lfloor nt\rfloor] - \tilde{q}_{12}^{(n)}[\lfloor nt\rfloor]}{\sqrt{n}} \geq \frac{c}{2}\Big) = 0.$$

Similarly, we can show that

$$\lim_{n\to\infty} P\Big(\sup_{t\in[0,T]} \frac{q_{34}^{(n)}[\lfloor nt\rfloor] - \tilde{q}_{34}^{(n)}[\lfloor nt\rfloor]}{\sqrt{n}} \geq \frac{c}{2}\Big) = 0.$$

Observing that

$$\begin{aligned} & P\Big(\sup_{t\in[0,T]} \frac{\sum_l q_l^{(n)}[\lfloor nt\rfloor] - \sum_l \tilde{q}_l^{(n)}[\lfloor nt\rfloor]}{\sqrt{n}} \geq c\Big) \\ \leq\ & P\Big(\sum_{l=\{12\}}^{\{34\}} \sup_{t\in[0,T]} \frac{q_l^{(n)}[\lfloor nt\rfloor] - \tilde{q}_l^{(n)}[\lfloor nt\rfloor]}{\sqrt{n}} \geq c\Big) \\ \leq\ & P\Big(\sup_{t\in[0,T]} \frac{q_{12}^{(n)}[\lfloor nt\rfloor] - \tilde{q}_{12}^{(n)}[\lfloor nt\rfloor]}{\sqrt{n}} \geq \frac{c}{2}\Big) + P\Big(\sup_{t\in[0,T]} \frac{q_{34}^{(n)}[\lfloor nt\rfloor] - \tilde{q}_{34}^{(n)}[\lfloor nt\rfloor]}{\sqrt{n}} \geq \frac{c}{2}\Big), \end{aligned}$$

we get the desired result. □

2.9 Summary

There has been much interest recently in understanding the optimality properties of the MWS-α algorithm. However, in general, optimal algorithms to minimize the total queue backlog (workload) were unknown. As a first step, in this chapter, we considered very small generalized switches and derived scheduling algorithms that are workload-optimal in a heavy-traffic sense. Since it is difficult to prove that our algorithms are the unique optimal algorithms, it is interesting to try to understand how well the MWS-α algorithm performs compared to our optimal algorithms. Our simulations show that, when the load on the system is close to its capacity, the proposed optimal algorithms perform better than the MWS-0 and MWS-α for small α in an appropriate heavy-traffic sense described in the chapter. Our results show that the conjecture of heavy-traffic workload optimality of MWS-α algorithms is not true in general.

CHAPTER 3

BACKPRESSURE-BASED PACKET-BY-PACKET ADAPTIVE ROUTING IN COMMUNICATION NETWORKS

The backpressure algorithm introduced in [1] has been widely studied in the literature. While the ideas behind scheduling using the weights suggested in that paper have been successful in practice in base stations and routers, the adaptive routing idea of the backpressure algorithm is rarely used. The main reason for this is that the routing algorithm can lead to poor delay performance due to routing loops. Additionally, the implementation of the backpressure algorithm requires each node to maintain per-destination queues which can be burdensome for a wireline or wireless router. Motivated by these considerations, we reexamine the backpressure routing algorithm in this chapter and design a new algorithm which has much superior performance and low implementation complexity.

Prior work in this area [17] has recognized the importance of doing shortest-path routing to improve delay performance and modified the backpressure algorithm to bias it towards taking shortest-hop routes. A part of our algorithm has similar motivating ideas, but we do much more. In addition to provably throughput-optimal routing which minimizes the number of hops taken by packets in the network, we decouple routing and scheduling in the network through the use of probabilistic routing tables and the so-called shadow queues. The min-hop routing idea was studied first in a conference paper [41] and shadow queues were introduced in [16] and [42], but the key step of decoupling the routing and scheduling which leads to both significant delay reduction and the use of per-next-hop queueing is original here. In [16], the authors introduced the shadow queue to solve a fixed routing problem. The min-hop routing idea is also studied in [43], but their solution requires even more queues than the original backpressure algorithm.

We also consider networks where simple forms of network coding are allowed [44]. In such networks, a relay between two other nodes XORs packets and broadcast them to decrease the number of transmissions. There is a tradeoff between choosing long routes to possibly increase network coding opportunities (see the notion of reverse carpooling in [45]) and choosing short routes to reduce resource usage. Our adaptive routing algorithm can be modified to automatically realize this tradeoff with good delay performance. In addition, network coding requires each node to maintain more queues [46] and our routing solution at least reduces the number of queues to be maintained for routing purposes, thus partially mitigating the problem. An offline algorithm for optimally computing the routing-coding tradeoff was proposed in [47]. Our optimization formulation bears similarities to this work, but our main focus is on designing low-delay on-line algorithms. Backpressure solutions to network coding problems have also been studied in [48, 49, 50], but the adaptive routing-coding tradeoff solution that we propose here has not been studied previously.

We summarize our main results below.

- Using the concept of shadow queues, we decouple routing and scheduling. A shadow network is used to update a probabilistic routing table which packets use upon arrival at a node. The backpressure-based scheduling algorithm is used to serve FIFO queues over each link.

- The routing algorithm is designed to minimize the average number of hops used by packets in the network. This idea, along with the scheduling/routing decoupling, leads to delay reduction compared with the traditional backpressure algorithm.

- Each node has to maintain counters, called shadow queues, per destination. This is very similar to the idea of maintaining a routing table per destination. But the real queues at each node are per-next-hop queues in the case of networks which do not employ network coding. When network coding is employed, per-previous-hop queues may also be necessary, but this is a requirement imposed by network coding, not by our algorithm.

- The algorithm can be applied to wireline and wireless networks. Extensive simulations show nice improvement in delay performance compared to the backpressure algorithm.

3.1 The Network Model

We consider a multi-hop wireline or wireless network represented by a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{L}$ is the set of directed links. A directed link that can transmit packets from node n to node j is denoted by $(nj) \in \mathcal{L}$. We assume that time is slotted and define the link capacity c_{nj} to be the maximum number of packets that link (nj) can transmit in one time slot.

Let $\mathcal{F}$ be the set of flows that share the network. Each flow is associated with a source node and a destination node, but no route is specified between these nodes. This means that the route can be quite different for packets of the same flow. Let $b(f)$ and $e(f)$ be source and destination nodes, respectively, of flow f. Let x_f be the rate (packets/slot) at which packets are generated by flow f. If the demand on the network, i.e., the set of flow rates, can be satisfied by the available capacity, there must exist a routing algorithm and a scheduling algorithm such that the link rates lie in the capacity region. To precisely state this condition, we define μ_{nj}^d to be the rate allocated on link (nj) to packets destined for node d. Thus, the total rate allocated to all flows at link (nj) is given by $\mu_{nj} := \sum\limits_{d \in \mathcal{N}} \mu_{nj}^d$. Clearly, for the network to be able to meet the traffic demand, we should have:

$$\{\mu_{nj}\}_{(nj)\in\mathcal{L}} \in \Lambda,$$

where Λ is the capacity region of the network for 1-hop traffic. The capacity region of the network for 1-hop traffic contains all sets of rates that are stabilizable by some kind of scheduling algorithm assuming all traffics are 1-hop traffic. As a special case, in the wireline network, the constraints are:

$$\mu_{nj} \leq c_{nj}, \qquad \forall (nj).$$

As opposed to Λ, let Υ denote the capacity region of the multi-hop network; i.e., for any set of flows $\{x_f\}_{f\in\mathcal{F}} \in \Upsilon$, there exists some routing and scheduling algorithms that stabilize the network.

In addition, a flow conservation constraint must be satisfied at each node; i.e., the total rate at which traffic can possibly arrive at each node destined to d must be less than or equal to the total rate at which traffic can depart from the node destined to d :

$$\sum_{f\in\mathcal{F}} x_f \mathbb{1}_{\{b(f)=n,e(f)=d\}} + \sum_{l:(ln)\in\mathcal{L}} \mu_{ln}^d \leq \sum_{j:(nj)\in\mathcal{L}} \mu_{nj}^d, \tag{3.1}$$

where $\mathbb{1}$ denotes the indicator function. Given a set of arrival rates $x = \{x_f\}_{f\in\mathcal{F}}$ that can be accommodated by the network, one version of the multi-commodity flow problem is to find the traffic splits μ_{nj}^d such that (3.1) is satisfied. However, finding the appropriate traffic split is computationally prohibitive and requires knowledge of the arrival rates. The backpressure algorithm to be described next is an adaptive solution to the multi-commodity flow problem.

3.2 Throughput-Optimal Backpressure Algorithm and Its Limitations

The backpressure algorithm was first described in [1] in the context of wireless networks and independently discovered later in [51] as a low-complexity solution to certain multi-commodity flow problems. This algorithm combines the scheduling and routing functions together. While many variations of this basic algorithm have been studied, they primarily focus on maximizing throughput and do not consider QoS performance. Our algorithm uses some of these ideas as building blocks, and therefore we first describe the basic algorithm, its drawbacks and some prior solutions.

The algorithm maintains a queue for each destination at each node. Since the number of destinations can be as large as the number of nodes, this per-destination queueing requirement can be quite large for practical implementation in a network. At each link, the algorithm assigns a weight to each possible destination which is called *backpressure.* Define the backpressure at link (nj) for destination d at slot t to be

$$w_{nj}^{d}[t] = Q_{nd}[t] - Q_{jd}[t],$$

where $Q_{nd}[t]$ denotes the number of packets at node n destined for node d at the beginning of time slot t. Under this notation, $Q_{nn}[t] = 0, \forall t$. Assign a weight w_{nj} to each link (nj), where w_{nj} is defined to be the maximum backpressure over all possible destinations, i.e.,

$$w_{nj}[t] = \max_{d} w_{nj}^{d}[t].$$

Let d_{nj}^{*} be the destination which has the maximum weight on link (nj),

$$d_{nj}^{*}[t] = \arg\max_{d}\{w_{nj}^{d}[t]\}. \tag{3.2}$$

If there are ties in the weights, they can be broken arbitrarily. Packets belonging to destination $d_{nj}^{*}[t]$ are scheduled for transmission over the activated link (nj). A schedule is a set of links that can be activated simultaneously without interfering with each other. Let Γ denote the set of all schedules. The backpressure algorithm finds an optimal schedule $\pi^{*}[t]$ which is derived from the optimization problem:

$$\pi^{*}[t] = \arg\max_{\pi\in\Gamma} \sum_{(nj)\in\pi} c_{nj} w_{nj}[t]. \tag{3.3}$$

Specially, if the capacity of every link has the same value, the chosen schedule maximizes the sum of weights in any schedule.

At time t, for each activated link $(nj) \in \pi^{*}[t]$ we remove c_{nj} packets from $Q_{nd_{nj}^{*}[t]}$ if possible, and transmit those packets to $Q_{jd_{nj}^{*}[t]}$. We assume that the departures occur first in a time slot, and external arrivals and packets transmitted over a link (nj) in a particular time slot are available to node j at the next time slot. Thus the evolution of the queue $Q_{nd}[t]$ is as follows:

$$\begin{aligned} Q_{nd}[t+1] &= Q_{nd}[t] - \sum_{j:(nj)\in\mathcal{L}} \mathbb{1}_{\{d^*_{nj}[t]=d\}}\, \hat{\mu}_{nj}[t] \\ &\quad + \sum_{l:(ln)\in\mathcal{L}} \mathbb{1}_{\{d^*_{ln}[t]=d\}}\, \hat{\mu}_{ln}[t] + \sum_{f\in\mathcal{F}} \mathbb{1}_{\{b(f)=n,e(f)=d\}}\, a_f[t], \end{aligned} \tag{3.4}$$

where $\hat{\mu}_{nj}[t]$ is the number of packets transmitted over link (nj) in time slot t and $a_f[t]$ is the number of packets generated by flow f at time t. It has been shown in [1] that the backpressure algorithm maximizes the throughput of the network.

A key feature of the backpressure algorithm is that packets may not be transferred over a link unless the backpressure over a link is non-negative and the link is included in the picked schedule. This feature prevents further congesting nodes that are already congested, thus providing the adaptivity of the algorithm. Notice that because all links can be activated without interfering with each other in the wireline network, Γ is the set of all links. Thus the backpressure algorithm can be localized at each node and operated in a distributed manner in the wireline network.

The backpressure algorithm has several disadvantages that prohibit practical implementation:

- The backpressure algorithm requires maintaining queues for each potential destination at each node. This queue management requirement could be a prohibitive overhead for a large network.

- The backpressure algorithm is an adaptive routing algorithm which explores the network resources and adapts to different levels of traffic intensity. However it might also lead to high delays because it may choose long paths unnecessarily. High delays are also a result of maintaining a large number of queues at each node, and each of those queues being large. The queues can be large because, under the backpressure algorithm, average size of a per-destination queue at a node can grow with the distance from the node to the destination. Furthermore, a large number of queues takes away statistical multiplexing advantage: since only one queue can be scheduled at a time, some of the allocated transmission capacity can be left unused if the scheduled queue is too short; this can contribute to high latency as well.

In this chapter, we address the high delay and queueing complexity issues. The computational complexity issue for wireless networks is not addressed here. We simply use the recently studied greedy maximal scheduling (GMS) algorithm. Here we call it the *largest-weight-first* algorithm, in short, the LWF algorithm. The LWF algorithm requires the same queueing structure that the backpressure algorithm uses. It also calculates the backpressure at each link using the same way. The difference between these two algorithms only lies in the methods to pick a schedule. Let $\mathcal{S}$ denote the set of all links initially. Let $\mathcal{N}_b(l)$ be the set of links within the interference range of link l including l itself. At each time slot, the LWF algorithm picks a link l with the maximum weight first, and removes links within the interference range of link l from $\mathcal{S}$, i.e., $\mathcal{S} = \mathcal{S} \backslash \mathcal{N}_b(l)$; then it picks the link with the maximum weight in the updated set $\mathcal{S}$, and so forth. It should be noticed that the LWF algorithm reduces the computational complexity at the cost of reduced network capacity. The LWF algorithm where the weights are queue lengths (not backpressures) has been extensively studied in [3, 52, 53, 54, 55]. While these studies indicate that there may be reduction in throughput due to LWF in certain special network topologies, it seems to perform well in practice and so we adopt it here for simulations.

In the rest of the chapter, we present our main results which eliminate many of the problems associated with the backpressure algorithm.

3.3 Min-Resource Routing Using Backpressure Algorithm

As mentioned in Section 3.2, the backpressure algorithm explores all paths in the network and as a result may choose paths which are unnecessarily long and which may even contain loops, thus leading to poor delay performance. Loc et al. address this problem by introducing a cost function which measures the total amount of resources used by all flows in the network [41]. Specifically, they add up traffic loads on all links in the network and use this as their cost function. The goal then is to minimize this cost subject to network capacity constraints.

Given a set of packet arrival rates that lie within the capacity region, the goal is to find the routes for flows so that the minimum amount of resources is used in the network if possible. Thus, the problem is formulated as the following optimization problem:

$$\begin{aligned} \min \quad & \sum_{(nj)\in\mathcal{L}} \mu_{nj} \\ s.t. \quad & \sum_{f\in\mathcal{F}} x_f \mathbb{1}_{\{b(f)=n,e(f)=d\}} + \sum_{(ln)\in\mathcal{L}} \mu_{ln}^d \ \le \ \sum_{(nj)\in\mathcal{L}} \mu_{nj}^d, \forall d\in\mathcal{N}, n\in\mathcal{N}, \\ & \{\mu_{nj}\}_{(nj)\in\mathcal{L}} \ \in \ \Lambda. \end{aligned} \tag{3.5}$$

We now show how a modification of the backpressure algorithm can be used to solve this min-resource routing problem. (Note that similar approaches have been used in [56, 57, 58, 59, 60] to solve related resource allocation problems.)

Let $\{q_{nd}\}$ be the Lagrange multipliers corresponding to the flow conservation constraints in problem (3.5). Appending these constraints to the objective, we get

$$\begin{aligned} & \min_{\boldsymbol{\mu}\in\Lambda} \sum_{(nj)\in\mathcal{L}} \mu_{nj} + \sum_{n,d} q_{nd}\Big(\sum_{f\in\mathcal{F}} x_f \mathbb{1}_{\{b(f)=n,e(f)=d\}} + \sum_{(ln)\in\mathcal{L}} \mu_{ln}^d - \sum_{(nj)\in\mathcal{L}} \mu_{nj}^d\Big) \\ = \ & \min_{\boldsymbol{\mu}\in\Lambda}\Big(-\sum_{(nj)\in\mathcal{L}} \sum_{d} \mu_{nj}^d\big(q_{nd}-q_{jd}-1\big) - \sum_{n,d} q_{nd} \sum_{f\in\mathcal{F}} x_f \mathbb{1}_{\{b(f)=n,e(f)=d\}}\Big). \end{aligned} \tag{3.6}$$

If the Lagrange multipliers are known, then the optimal $\boldsymbol{\mu}$ can be found by solving

$$\max_{\boldsymbol{\mu}\in\Lambda} \sum_{(nj)\in\mathcal{L}} \mu_{nj} w_{nj}$$

where $w_{nj} = \max_d (q_{nd} - q_{jd} - 1)$. The form of the constraints in (3.5) suggests the following update algorithm to compute q_{nd} :

$$q_{nd}[t+1] = \Big[q_{nd}[t] + \frac{1}{M}\Big(\sum_{f\in\mathcal{F}} x_f \mathbb{1}_{\{b(f)=n,e(f)=d\}} + \sum_{(ln)\in\mathcal{L}} \mu_{ln}^d - \sum_{(nj)\in\mathcal{L}} \mu_{nj}^d\Big)\Big]^+, \tag{3.7}$$

where $\dfrac{1}{M}$ is a step-size parameter. See [61] for details. Notice that $Mq_{nd}[t]$ looks very much like a queue update equation, except for the fact that arrivals into Q_{nd} from other links may be smaller than μ_{ln}^{d} when Q_{ld} does not have enough packets. This suggests the following algorithm.

Min-resource routing by backpressure: At time slot t,

- Each node n maintains a separate queue of packets for each destination d; its length is denoted $Q_{nd}[t]$. Each link is assigned a weight

$$w_{nj}[t] = \max_{d} \left(Q_{nd}[t] - Q_{jd}[t] - M \right), \tag{3.8}$$

  where $M > 0$ is a parameter.

- Scheduling/routing rule: find an optimal schedule $\pi^*[t]$ such that

$$\pi^*[t] \in \arg\max_{\pi \in \Gamma} \sum_{(nj) \in \pi} c_{nj} w_{nj}[t]. \tag{3.9}$$

- For each activated link $(nj) \in \pi^*[t]$ we remove c_{nj} packets from $Q_{nd^*_{nj}[t]}$ if possible, and transmit those packets to $Q_{jd^*_{nj}[t]}$, where $d^*_{nj}[t]$ achieves the maximum in (3.8).

Compared with the traditional backpressure scheduling/routing, the only difference is that each link weight is equal to the maximum differential backlog *minus parameter* M. ($M = 0$ reverts the algorithm to the traditional one.) For simplicity, we call this algorithm the *M-backpressure algorithm.*

The performance of the stationary process which is "produced" by the algorithm with fixed parameter M is within $o(1)$ of the optimal as M goes to ∞ (analogous to the proofs in [57, 58]; see also the related proof in [59, 60]):

$$\left| \mathbb{E}\left[\sum_{(nj) \in \mathcal{L}} \mu_{nj}[\infty] \right] - \sum_{(nj) \in \mathcal{L}} \mu^*_{nj} \right| = o(1),$$

where μ^* is an optimal solution to (3.5).

Figure 3.1 illustrates an example of how the M-backpressure algorithm works in a simple wireline network. All links can be activated simultaneously without interfering with each other. Notice that the backlog difference of route 1 is 6 and the backlog difference of route 2 is 4. M is chosen to be 5 in this example. Because the backlog difference of route 2 is smaller than M, route 2 is blocked at current traffic load. The M-backpressure algorithm will automatically choose route 1 which is shorter. Therefore, a proper M can avoid long routes in when the traffic is not close to capacity. Throughput optimality is still achieved by using M-backpressure algorithm. In this example, route 2 will be enabled if the traffic load is close to the boundary of the capacity region due to large backlog difference which is greater than M.

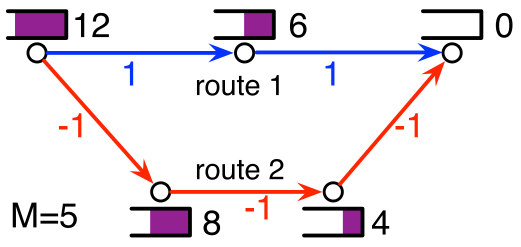

Figure 3.1: Link weights under the M-backpressure algorithm.

Although the M-backpressure algorithm could reduce the delay by forcing flows to go through shorter routes, simulations indicate a significant problem with the basic algorithm presented above. A link can be scheduled only if the backpressure of at least one destination is greater than or equal to M. Thus, at light to moderate traffic loads, the delays could be high since the backpressure may not build up sufficiently fast. Also, the upper-stream queue backlogs could be huge if the number of hops between the source and the destination is large. Huge upstream backlogs are detrimental to delay QoS. In practice, the residual packets in queues can not be removed with $M > 0$ at the end of the transmission stage. Like the traditional backpressure algorithm, the M-backpressure algorithm also maintains per-destination queues at each node. The queueing complexity is still high, and there are scalability issues.

Furthermore, the queueing structure is not compatible with today's switches and routers. In order to overcome all these adverse issues, we develop a new routing algorithm in the following section. The solution also simplifies the queueing structure to be maintained at each node.

3.4 PARN: Packet-by-Packet Adaptive Routing and Scheduling Algorithm for Networks

In this section, we present our adaptive routing and scheduling algorithm. We will call it PARN (Packet-by-Packet Adaptive Routing for Networks) for ease for repeated reference later. First, we introduce the queueing structure that is used in PARN.

In the traditional backpressure algorithm, each node n has to maintain a queue q_{nd} for each destination d. Let $|\mathcal{N}|$ and $|\mathcal{D}|$ denote the number of nodes and the number of destinations in the network, respectively. Each node maintains $|\mathcal{D}|$ queues. Generally, each pair of nodes can communicate along a path connecting them. Thus, the number of queues maintained at each node can be as high as one less than the number of nodes in the network, i.e., $|\mathcal{D}|$=$|\mathcal{N}| - 1$.

Instead of keeping a queue for every destination, each node n maintains a queue q_{nj} for every neighbor j, which is called a *real queue*. Notice that real queues are per-neighbor queues. Let J_n denote the number of neighbors of node n, and let $J_{max} = \max_n J_n$. The number of queues at each node is no greater than J_{max}. Generally, J_{max} is much smaller than $|\mathcal{N}|$. Thus, the number of queues at each node is much smaller compared with the case using the traditional backpressure algorithm.

In addition to real queues, each node n also maintains a counter, which is called a *shadow queue*, p_{nd}, for each destination d. Unlike the real queues, counters are much easier to maintain even if the number of counters at each node grows linearly with the size of the network. A backpressure algorithm run on the shadow queues is used to decide which links to activate. The statistics of the link activation are further used to route packets to the per-next-hop neighbor queues mentioned earlier. The details are explained next.

3.4.1 Shadow Queue Algorithm – M-Backpressure Algorithm

The shadow queues are updated based on the movement of fictitious entities called shadow packets in the network. The movement of the fictitious packets can be thought of as an exchange of control messages for the purposes of routing and scheduling. Just like real packets, shadow packets arrive from outside the network and eventually exit the network. The external shadow packet arrivals are generated as follows: when an exogenous packet arrives at node n to the destination d, the shadow queue p_{nd} is incremented by 1, and is further incremented by 1 with probability ε in addition. Thus, if the arrival rate of a flow f is x_f, then the flow generates "shadow traffic" at a rate $x_f(1+\varepsilon)$. In words, the incoming shadow traffic in the network is $(1+\varepsilon)$ times the incoming real traffic.

The backpressure for destination d on link (nj) is taken to be

$$w_{nj}^{d}[t] = p_{nd}[t] - p_{jd}[t] - M,$$

where M is a properly chosen parameter. The choice of M will be discussed in the simulations section. The evolution of the shadow queue $p_{nd}[t]$ is

$$\begin{aligned} p_{nd}[t+1] &= p_{nd}[t] - \sum_{j:(nj)\in\mathcal{L}} \mathbb{1}_{\{d_{nj}^{*}[t]=d\}}\,\hat{\mu}_{nj}[t] \\ &\quad + \sum_{l:(ln)\in\mathcal{L}} \mathbb{1}_{\{d_{ln}^{*}[t]=d\}}\,\hat{\mu}_{ln}[t] + \sum_{f\in\mathcal{F}} I_{\{b(f)=n,e(f)=d\}}\,\hat{a}_f[t], \end{aligned} \tag{3.10}$$

where $\hat{\mu}_{nj}[t]$ is the number of shadow packets transmitted over link (nj) in time slot t, $d_{nj}^{*}[t]$ is the destination that has the maximum weight on link (nj), and $\hat{a}_f[t]$ is the number of shadow packets generated by flow f at time t. The number of shadow packets scheduled over the links at each time instant is determined by the backpressure algorithm in (3.9).

From the above description, it should be clear that the shadow algorithm is the same as the traditional backpressure algorithm, except that it operates on the shadow queueing system with an arrival rate slightly larger than the real external arrival rate of packets. Note the shadow queues do not involve any queueing structure at each node; there are no packets to maintain in a FIFO order in each queue. The shadow queue is simply a counter which is incremented by 1 upon a shadow packet arrival and decremented by 1 upon a departure.

The backpressure algorithm run on the shadow queues is used to activate the links. In other words, if $\pi^*_{nj} = 1$ in (3.9), then link (nj) is activated and packets are served from the real queue at the link in a first-in, first-out fashion. This is, of course, very different from the traditional backpressure algorithm where a link is activated to serve packets to a particular destination. Thus, we have to develop a routing scheme that assigns packets arriving to a node to a particular next-hop neighbor so that the system remains stable.

3.4.2 Adaptive Routing Algorithms

Now we discuss how a packet is routed once it arrives at a node. Let us define a variable $\sigma^d_{nj}[t]$ to be the number of shadow packets "transferred" from node n to node j for destination d during time slot t by the shadow queue algorithm. Let us denote by $\bar{\sigma}^d_{nj}$ the expected value of $\sigma^d_{nj}[t]$, when the shadow queueing process is in a stationary regime; let $\hat{\sigma}^d_{nj}[t]$ denote an estimate of $\bar{\sigma}^d_{nj}$, calculated at time t. (In the simulations we use the exponential averaging, as specified in the next section.)

At each time slot, the following sequence of operations occurs at each node n. A packet arriving at node n for destination d is inserted in the real queue q_{nj} for next-hop neighbor j with probability

$$P^d_{nj}[t] = \frac{\hat{\sigma}^d_{nj}[t]}{\sum_{k:(nk)\in\mathcal{L}} \hat{\sigma}^d_{nk}[t]}. \tag{3.11}$$

Thus, the estimates $\hat{\sigma}^d_{nj}[t]$ are used to perform routing operations: in today's routers, based on the destination of a packet, a packet is routed to its next hop based on routing table entries. Instead, here, the $\bar{\sigma}$'s are used to probabilistically choose the next hop for a packet. Packets waiting at link (nj) are transmitted over the link when that link is scheduled (see

Figure 3.2).

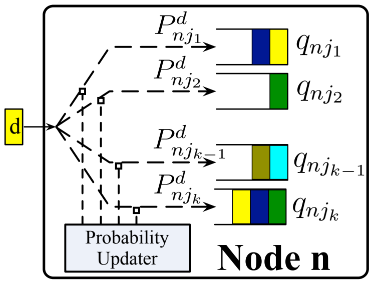

Figure 3.2: Probabilistic splitting algorithm at Node n.

The first question that one must ask about the above algorithm is whether it is stable if the packet arrival rates from flows are within the capacity region of the multi-hop network. This is a difficult question, in general. Since the shadow queues are positive recurrent, "good" estimates $\hat{\sigma}^d_{nj}[t]$ can be maintained by simple averaging (e.g. as specified in the next section), and therefore the probabilities in (3.11) will stay close to their "ideal" values

$$\bar{P}^d_{nj} = \frac{\bar{\sigma}^d_{nj}}{\sum_{k:(nk)\in\mathcal{L}} \bar{\sigma}^d_{nk}}.$$

The following theorem asserts that the real queues are stable under the additional assumption that the routing probabilities P^d_{nj} are fixed at their ideal values $\bar{P}^d_{nj}$ (as opposed to being updated via (3.11), which is what the actual algorithm does).

Theorem 3.4.1. *Suppose the routing probabilities are fixed at $\bar{P}^d_{nj}$. Assume that there exists a delta such that $\{x_f(1+\epsilon+\delta)\}$ lies in Γ. Let $a_f[t]$ be the number of packets arriving from flow f at time slot t, with $E(a_f[t]) = x_f$ and $E(a_f[t]) < \infty$. Assume that the arrival process is independent across time slots and flows (this assumption can be considerably relaxed). Then, the Markov chain, jointly describing the evolution of shadow queues and real FIFO queues (whose state include the destination of the real packet in each position of each FIFO queue), is positive recurrent.*

Proof. The key ideas behind the proof are outlined. The details are similar to the proof in [16] and are omitted.

- The average rate at which packets arrive to link (nj) is strictly smaller than the capacity allocated to the link by the shadow process if $\varepsilon > 0$.
- It follows that the fluid limit of the real-queue process is same as that of the networks in [62]. Such fluid limit is stable [62], which implies the stability of our process as well. See Section 3.8 for details.

$\square$

3.5 Implementation Details

In this section, we discuss a number of enhancements to the basic algorithm to improve performance.

3.5.1 Exponential Averaging

In order to take most recent statistics with larger weight, one possible solution is exponential averaging. Exponential averaging fades old values exponentially with weight $1-\beta$ and takes new statistics into account with weight β. To compute $\hat{\sigma}^d_{nj}[t]$ we use the following iterative exponential averaging algorithm:

$$\hat{\sigma}^d_{nj}[t] = (1-\beta)\ \hat{\sigma}^d_{nj}[t-1] + \beta\ \sigma^d_{nj}[t], \tag{3.12}$$

where $0 < \beta < 1$. By choosing β large or small, we can adjust how important the latest statistics is. Another way to take recent statistics into account while fading the old ones is to use moving averaging. However, to suppress the randomness, we should keep a large amount of previous data. For simplicity, we consider exponential averaging in our implementation.

3.5.2 Extra Link Activation

Under the shadow backpressure algorithm, only links with backpressure greater than or equal to M can be activated. The stability theory ensures that this is sufficient to render the real queues. On the other hand, the delay performance can still be unacceptable. Recall that the parameter M was introduced to discourage the use of unnecessarily long paths. However, under light and moderate traffic loads, the shadow backpressure at a link may be frequently less than M, and thus, packets at such links may have to wait a long time before they are processed. One way to remedy the situation is to activate additional links beyond those activated by the shadow backpressure algorithm.

The basic idea is as follows: in each time slot, first run the shadow backpressure algorithm. Then, add additional links to make the schedule maximal. If the extra activation procedure depends only on the state of shadow queues (but beyond that, can be random and/or arbitrarily complex), then the stability result of Theorem 3.4.1 still holds (with essentially same proof). Informally, the stability prevails, because the shadow algorithm alone provides sufficient average throughput on each link, and adding extra capacity "does not hurt"; thus, with such extra activation, a certain degree of "decoupling" between routing (totally controlled by shadow queues) and scheduling (also controlled by shadow queues, but not completely) is achieved.

For example, in the case of wireline networks, by the above arguments, all links can be activated all the time. The shadow routing algorithm ensures that the arrival rate at each link is less than its capacity. In this case the *complete* decoupling of routing and scheduling occurs.

In practice, activating extra links which have large queue backlogs leads to better performance than activating an arbitrary set of extra links. However, in this case, the extra activation procedure depends on the state of real queues, which makes the issue of validity of an analog of Theorem 3.4.1 much more subtle. We believe that the argument in this subsection provides a good motivation for our algorithm, which is confirmed by simulations.

3.5.3 The Choice of the Parameter ε

From basic queueing theory, we expect the delay at each link to be inversely proportional to the mean capacity minus the arrival rate at the link. In a wireless network, the capacity at a link is determined by the shadow scheduling algorithm. This capacity is guaranteed to be at least equal to the shadow arrival rate. The arrival rate of real packets is of course smaller. Thus, the difference between the link capacity and arrival rate could be proportional to epsilon. Thus, epsilon should be large enough to ensure small delays and small enough to ensure that the capacity region is not diminished significantly. In our simulations, we found that choosing $\varepsilon = 0.1$ provides a good tradeoff between delay and network throughput.

In the case of wireline networks, recall from the previous subsection that all links are activated. Therefore, the parameter epsilon plays no role here.

3.6 Extension to the Network Coding Case

In this section, we extend our approach to consider networks where network coding is used to improve throughput. We consider a simple form of network coding illustrated in Figure 3.3. When i and j each have a packet to send to the other through an intermediate relay n, traditional transmission requires the following set of transmissions: send a packet a from i to n, then n to j, followed by j to n and n to i. Instead, using network coding, one can first send from i to n, then j to n, XOR the two packets and broadcast the XORed packet from n to both i and j. This form of network coding reduces the number of transmissions from four to three. However, the network coding can improve throughput only if such coding opportunities are available in the network. Routing plays an important role in determining whether such opportunities exist. In this section, we design an algorithm to automatically find the right tradeoff between using possibly long routes to provide network coding opportunities and the delay incurred by using long routes.

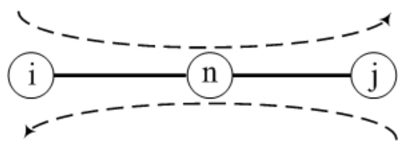

Figure 3.3: Network coding opportunity.

3.6.1 System Model

We still consider the wireless network represented by the graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$. Let x_f be the rate (packets/slot) at which packets are generated by flow f. To facilitate network coding, each node must not only keep track of the destination of the packet, but also remember the node from which a packet was received. Let μ_{lnj}^{d} be the rate at which packets received from either node l or external incoming flow l, destined for node d, are scheduled over link (nj). Note that, for compactness of notation, we allow l in the definition of μ_{lnj}^{d} to denote either a flow or a node. We assume μ_{lnj}^{d} is zero when such a transmission is not feasible, i.e., when n is not the source node or d is not the destination node of flow l, or if (ln) or (nj) is not included in $\mathcal{L}$. At node n, the network coding scheme may generate a coded packet by "XORing" two packets received from previous-hop nodes l and j destined for the destination nodes d and d' respectively, and broadcast the coded packet to nodes j and l. Let $\mu_{n|jl}^{d,d'}$ denote the rate at which coded packets can be transferred from node n to nodes j and l destined for nodes d and d', respectively. Notice that, due to symmetry, the following equality holds: $\mu_{n|jl}^{d,d'} = \mu_{n|lj}^{d',d}$. Assume $\mu_{n|jl}^{d,d'}$ to be zero if at least one of $(nl), (ln), (nj)$ and (jn) does not belong to $\mathcal{L}$. Note that $\mu_{lnj}^{d} = 0$ when $d = l$ or $d = n$, and $\mu_{n|jl}^{d,d'} = 0$ when $d = n$ or $d' = n$.

There are two kinds of transmissions in our network model: point-to-point transmissions and broadcast transmissions. The total point-to-point rate at which packets received externally or from a previous-hop node are scheduled on link (nj) and destined to d is denoted by

$$\mu_{nj,pp}^{d} = \sum_{l:l\in\mathcal{F}} \mu_{lnj}^{d} + \sum_{l:l\in\mathcal{N}} \mu_{lnj}^{d},$$

and the total broadcast rate at which packets are scheduled on link (nj) and destined to d is denoted by

$$\mu_{nj,broad}^{d} = \sum_{d'} \sum_{l:l \neq j} \mu_{n|jl}^{d,d'}.$$

The total point-to-point rate on link (nj) is denoted by

$$\mu_{nj,pp} = \sum_{d} \mu_{nj,pp}^{d}$$

and the total broadcast rate at which packets are broadcast from node n to nodes j and l is denoted by

$$\mu_{n|jl} = \sum_{d'} \sum_{d} \mu_{n|jl}^{d,d'}.$$

Let $\boldsymbol{\mu}$ be the set of rates including all point-to-point transmissions and broadcast transmissions, i.e.,

$$\boldsymbol{\mu} \quad = \quad \{\{\mu_{nj,pp}\}_{(nj)}, \{\mu_{n|jl}\}_{(n|jl)}\}.$$

The multi-hop traffic should also satisfy the flow conservation constraints.
Flow conservation constraints: For each node n, each neighbor j, and each destination d, we have

$$\mu_{nj,pp}^{d} + \mu_{nj,broad}^{d} \leq \sum_{k} \mu_{njk}^{d} + \sum_{d'} \sum_{k:k \neq n} \mu_{j|kn}^{d,d'}, \tag{3.13}$$

where the left-hand side denotes the total incoming traffic rate at link nj destined to d, and the right-hand side denotes the total outgoing traffic rate from link nj destined to d. For each node n and each destination d, we have

$$\sum_{f \in \mathcal{F}} x_f \mathbb{1}_{\{b(f)=n, e(f)=d\}} \leq \sum_{f \in \mathcal{F}} \sum_{j \in \mathcal{N}} \mu^d_{fnj}, \tag{3.14}$$

where $\mathbb{1}$ denotes the indicator function.

3.6.2 Links and Schedules

We allow broadcast transmission in our network model. In order to define a schedule, we first define two kinds of "links:" the point-to-point link and the broadcast link. A point-to-point link (nj) is a link that supports point-to-point transmission, where $(nj) \in \mathcal{L}$; a broadcast link $(n|lj)$ is a "link" which contains links (nl) and (nj) and supports broadcast transmission. Let $\mathcal{B}$ denote the set of all broadcast links, thus $(n|lj) \in \mathcal{B}$. Let $\bar{\mathcal{L}}$ be the union of the set of the point-to-point links $\mathcal{L}$ and the set of the broadcast links $\mathcal{B}$, i.e., $\bar{\mathcal{L}} = \mathcal{L} \cup \mathcal{B}$.

We let Γ' denote the set of links that can be activated simultaneously. By abusing notation, Γ' can be thought of as a set of vectors where each vector is a list of 1's or 0's where a 1 corresponds to an active link and a 0 corresponds to an inactive link. Then, the capacity region of the network for 1-hop traffic is the convex hull of all schedules, i.e., $\Lambda' = co(\Gamma')$. Thus, $\mu \in \Lambda'$.

3.6.3 Queueing Structure and Shadow Queue Algorithm

Each node n maintains a set of counters, which are called *shadow queues,* p_{lnd}, for each previous hop l and each destination d, and p_{0nd} for external flows destined for d at node n. Each node n also maintains a real queue, denoted by q_{lnj}, for each previous hop l and each next-hop neighbor j, and q_{0nj} for external flows with their next hop j.

By solving the optimization problem with flow conservation constraints, we can work out the backpressure algorithm for the network coding case (see the brief description in Appendix 3.9). More specifically, for each link $(nj) \in L$ in the network and for each destination d, define the *backpressure* at every slot to be

$$\begin{aligned} & w_{nj}^d[t] = \max_{l:(ln)\in\mathcal{L} \text{ or } l=0} w_{lnj}^d[t] \\ & \text{where } w_{lnj}^d[t] = p_{lnd}[t] - p_{njd}[t] - M, \\ & \text{and } l_{nj}^*[t] = \arg\max_{l:(ln)\in\mathcal{L} \text{ or } l=0} w_{lnj}^d[t]. \end{aligned} \tag{3.15}$$

For each broadcast at node n to nodes j and l destined for d and d', respectively, define the *backpressure* at every slot to be

$$w_{n|jl}^{d,d'}[t] = w_{lnj}^d[t] + w_{jnl}^{d'}[t]. \tag{3.16}$$

The weights associated with each point-to-point link $(nj) \in \boldsymbol{\mathcal{L}}$ and each broadcast link $(n|jl)$ are defined as follows:

$$\begin{aligned} & w_{nj}[t] = \max_d\{w_{nj}^d[t]\}, \\ & w_{n|jl}[t] = \max_{d,d'}\{w_{n|jl}^{d,d'}[t]\}, \\ & \text{with } d_{nj}^*[t] = \arg\max_d\{w_{nj}^d[t]\}, \\ & \{d,d'\}_{n|jl}^*[t] = \arg\max_{d,d'}\{w_{n|jl}^{d,d'}[t]\}. \end{aligned} \tag{3.17}$$

The rate vector $\tilde{\boldsymbol{\mu}}^*[t]$ at each time slot is chosen to satisfy

$$\tilde{\boldsymbol{\mu}}^*[t] \in \arg\max_{\tilde{\boldsymbol{\mu}}\in\Gamma'} \Big\{ \sum_{(nj)\in\mathcal{L}} \tilde{\mu}_{nj,pp} w_{nj}[t] + \sum_{(n|jl)\in\mathcal{B}} \tilde{\mu}_{n|jl} w_{n|jl}[t] \Big\}.$$

By running the shadow queue algorithm in network coding case, we get a set of activated links in $\bar{\boldsymbol{\mathcal{L}}}$ at each slot.

Next we describe the evolution of the shadow queue lengths in the network. Notice that the shadow queues at each node n are distinguished by their previous hop l and their destination d, so p_{lnd} only accepts the packets from previous hop l with destination d. A similar rule should be followed when packets are drained from the shadow queue p_{lnd}. We assume the departures occur before arrivals at each slot, and the evolution of queues is given by

$$\begin{aligned} p_{lnd}[t+1] &= \Big[p_{lnd}[t] - \sum_{j\in\mathcal{N}} \tilde{\mu}^*_{nj,pp}[t]\mathbb{1}_{\{l=l^*_{nj},d=d^*_{nj}\}} - \sum_{d'\in\mathcal{N}}\sum_{j\in\mathcal{N}}\tilde{\mu}^*_{n|jl}[t]\mathbb{1}_{\{\{d,d'\}=\{d,d'\}^*_{n|jl}\}}\Big]^+ \\ &+ \sum_{k\in\mathcal{N}} \hat{\mu}^d_{kln}[t]\mathbb{1}_{\{k=l^*_{ln},d=d^*_{ln}\}} + \sum_{k\in\mathcal{N}}\sum_{d'\in\mathcal{N}} \hat{\mu}^{d,d'}_{l|nk}[t]\mathbb{1}_{\{\{d,d'\}=\{d,d'\}^*_{l|nk}\}} \\ &+ \sum_{f\in\mathcal{F}} \hat{a}_f[t]\mathbb{1}_{\{b(f)=n,e(f)=d,l=0\}}, \end{aligned} \tag{3.18}$$

where $\hat{\mu}^d_{kln}[t]$ is the actual number of shadow packets scheduled over link (ln) and destined for d from the shadow queue p_{kld} at slot t, $\hat{\mu}^{d,d'}_{l|nk}[t]$ is the actual number of coded shadow packets transfered from node l to nodes n and k destined for nodes d and d' at slot t, and $\hat{a}_f$ denotes the actual number of shadow packets from external flow f received at node n destined for d.

3.6.4 Implementation Details

The implementation details of the joint adaptive routing and scheduling algorithm with network coding are similar to the case without using network coding, but the notation is more cumbersome. We briefly describe it here.

The probabilistic splitting algorithm chooses the next hop of the packet based on the probabilistic routing table. Let $P^d_{lnj}[t]$ be the probability of choosing node j as the next hop once a packet destined for d arrives at node n from previous hop l or from external flows; i.e., $l=0$ at slot t. Assume that $P^d_{lnj}[t]=0$ if $(nj)\notin\mathcal{L}$. Obviously, $\sum_{j\in\mathcal{N}} P^d_{lnj}[t]=1$. Let $\sigma^d_{lnj}[t]$ denote the number of potential shadow packets "transferred" from node n to node j destined for d whose previous hop is l during time slot t. Notice that the packet comes from an external flow if $l=0$. Also notice that $\sigma^d_{lnj}[t]$ is contributed by shadow traffic point-to-point transmission as well as shadow traffic broadcast transmission, i.e.,

$$\sigma^d_{lnj}[t] \;= \mu^*_{nj,pp}[t]\mathbb{1}_{\{l=l^*_{nj}[t],d=d^*_{nj}[t]\}} + \sum_{d'\in\mathcal{N}} \mu^*_{n|jl}[t]\mathbb{1}_{\{\{d,d'\}=\{d,d'\}^*_{n|jl}[t]\}}.$$

We keep track of the the average value of $\sigma^d_{lnj}[t]$ across time by using the following exponential averaging updating process:

$$\hat{\sigma}^d_{lnj}[t] = (1-\beta)\hat{\sigma}^d_{lnj}[t-1] + \beta\sigma^d_{lnj}[t], \tag{3.19}$$

where $0 < \beta < 1$. The splitting probability $P^d_{lnj}[t]$ is expressed as follows:

$$P^d_{lnj}[t] = \frac{\hat{\sigma}^d_{lnj}[t]}{\sum_{k\in\mathcal{N}}\hat{\sigma}^d_{lnk}[t]}. \tag{3.20}$$

3.6.5 Extra Link Activation

Like the case without network coding, extra link activation can reduce delays significantly. As in the case without network coding, we add additional links to the schedule based on the queue lengths at each link. For extra link activation purposes, we only consider point-to-point links and not broadcast. Thus, we schedule additional point-to-point links by giving priority to those links with larger queue backlogs.

3.7 Simulations

We consider two types of networks in our simulations: wireline and wireless. Next, we describe the topologies and simulation parameters used in our simulations, and then present our simulation results.

3.7.1 Simulation Settings

Wireline Setting

The network shown in Figure 3.4 has 31 nodes and represents the GMPLS network topology of North America [63]. Each link is assume to be able to transmit one packets in each slot. We assume that the arrival process is a Poisson process with parameter λ, and we consider the arrivals that come within a slot are considered for service at the beginning of the next slot. Once a packet arrives from an external flow at a node n, the destination is decided by probability mass function $\hat{P}_{nd}, d = 1, 2, ...N$, where $\hat{P}_{nd}$ is the probability that a packet is received externally at node n destined for d. Obviously, $\sum_{d:d\neq n} \hat{P}_{nd} = 1$, and $\hat{P}_{nn} = 0$. The probability $\hat{P}_{nd}$ is calculated by

$$\hat{P}_{nd} = \frac{J_d + J_n}{\sum\limits_{k:k\neq n} (J_k + J_n)},$$

where J_n denotes the number of neighbors of node n. Thus, we use $\hat{P}_{nd}$ to split the incoming traffic to each destination based on the degrees of the source and the destination.

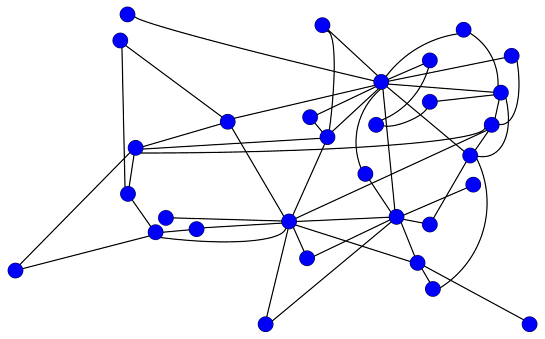

Figure 3.4: Sprint GMPLS network topology of North America with 31 nodes [63].

Wireless Setting

We generated a random network with 30 nodes which resulted in the topology in Figure 3.5. We used the following procedure to generate the random network: 30 nodes are placed uniformly at random in a unit square; then starting with a zero transmission range, the transmission range was increased till the network was connected. We assume that each link can transmit one packet per time slot. We assume a 2-hop interference model in our simulations. By a k-hop interference model, we mean a wireless network where a link activation silences all other links which are k hops from the activated link. The packet arrival processes are generated using the same method as in the wireline case. We simulate two cases given the network topology: the no coding case and the network coding case. In both wireline and wireless simulations, we chose β in (3.12) to be 0.02, and we use probabilistic splitting algorithm for simulations.

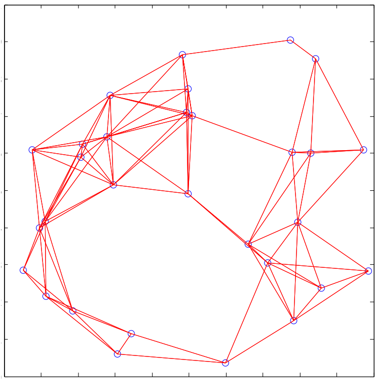

Figure 3.5: Wireless network topology with 30 nodes.

3.7.2 Simulation Results

Wireline Setting

First, we compare the performance of three algorithms: the traditional backpressure algorithm, the basic shadow queue routing/scheduling algorithm without the extra link activation enhancement and PARN. Without extra link activation, to ensure that the real arrival rate at each link is less than the link capacity provided by the shadow algorithm, we choose $\varepsilon = 0.02$. Figure 3.6 shows delay as a function of the arrival rate lambda for the three algorithms. As can be seen from the figure, simply using a value of $M > 0$ does not help to reduce delays without extra link activation. The reason is that, while $M > 0$ encourages the use of shortest paths, links with backpressure less than M will not be scheduled and thus can contribute to additional delays. Because we exaggerate the shadow traffic by a factor of ε, the throughput region of the algorithm without extra link activation is smaller than the throughput region of the traditional backpressure algorithm.

We also compare the delay performance of PARN with that of the shortest path routing in Figure 3.7. For each pair of source and destination, we find a shortest path between them by using Dijkstra's algorithm. When the arrival rate $\lambda < 0.38$, the difference between the

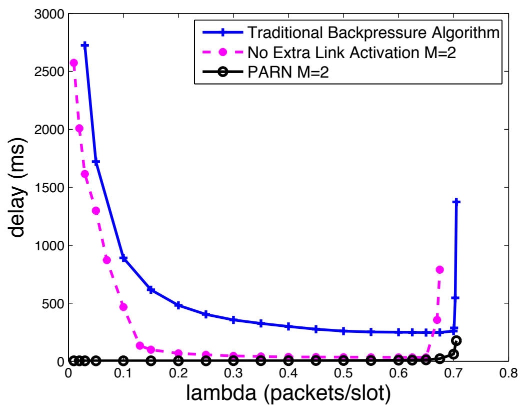

Figure 3.6: The impact of the parameter M in Sprint GMPLS network topology.

average packet delays of PARN and the shortest path routing is very small. This implies that PARN can obtain similar delay performance as the shortest path routing at light traffic. However, the shortest path routing can only achieve about 60% of the capacity region of the network.

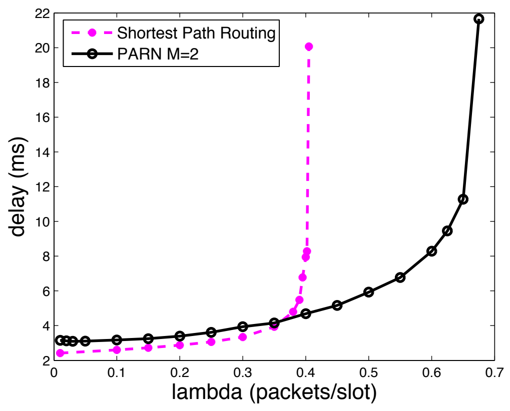

Figure 3.7: The delay performance of PARN and shortest path routing.

Next, we study the impact of M on the performance on PARN. Figure 3.8 shows the delay performance for various M with extra link activation in the wireline network. The delays for different values of M (except $M = 0$) are almost the same in the light traffic region. Once M is sufficiently larger than zero, extra link activation seems to play a bigger role than the choice of the value of M in reducing the average delays. Once we choose M to be large, the simulation takes longer to converge. This phenomenon is not captured in our plots, but it is very important in the practical implementation. A good choice of M should also depend on the network topology and the traffic pattern. Under our simulation setting, choosing M between 2 and 10 leads to good performance.

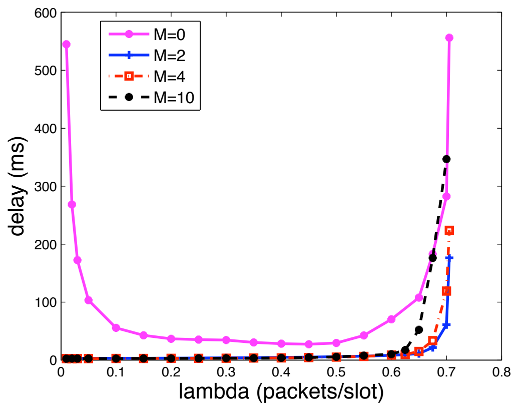

Figure 3.8: Packet delay as a function of λ under PARN in Sprint GMPLS network topology.

The wireline simulations show the usefulness of the PARN algorithm for adaptive routing. However, a wireline network does not capture the scheduling aspects inherent to wireless networks, which is studied next.

Wireless Setting

In the case of wireless networks, even with extra link activation, to ensure stability even when the arrival rates are within the capacity region, we need $\varepsilon > 0$. We chose $\varepsilon = 0.1$ in our simulations for the reasons mentioned in Section 3.5.

In Figure 3.9, we study wireless networks without network coding. From the figure, we see that the delay performance is relatively insensitive to the choice of M as long as it is sufficiently greater than zero. However, M does play an important role because it suppresses the search of long paths when the traffic load is not high. Extra link activation can be used to decrease delays significantly for $M > 0$ especially in light traffic region.

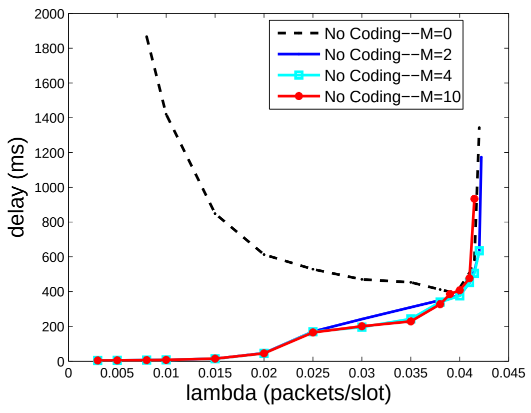

Figure 3.9: Packet delay as a function of λ under PARN in the wireless network under 2-hop interference model without network coding.

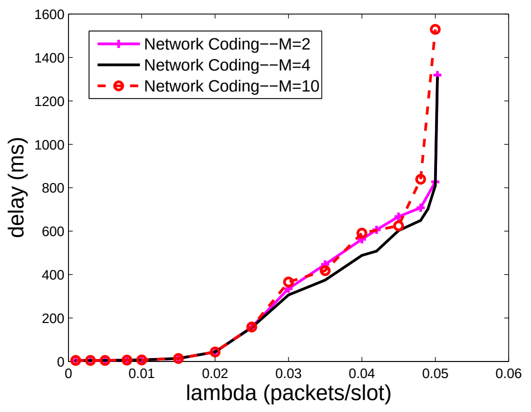

Figure 3.10: Packet delay as a function of λ under PARN for $M > 0$ in the wireless network under 2-hop interference model with network coding.

In Figures 3.10 and 3.11, we show the corresponding results for the case where both adaptive routing and network coding are used. Comparing Figures 3.9 and 3.10, we see that, when used in conjunction with adaptive routing, network coding can increase the capacity region. We make the following observation regarding the case $M = 0$ in Figure 3.11: in this case, no attempt is made to optimize routing in the network. As a result, the delay performance is very bad compared to the cases with $M > 0$ (Figure 3.10). In other

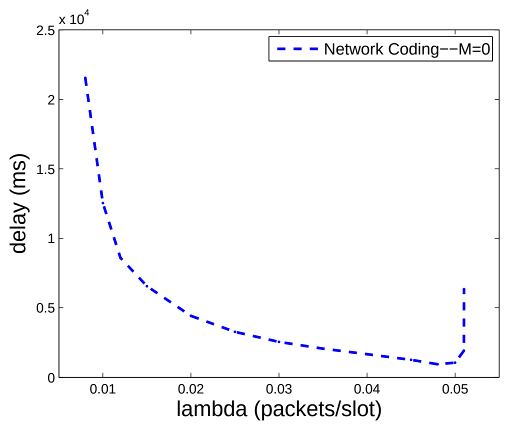

Figure 3.11: Packet delay as a function of λ under PARN for $M = 0$ in the wireless network under 2-hop interference model with network coding.

words, network coding alone does not increase capacity sufficiently to overcome the effects of backpressure routing. On the other hand, PARN with $M > 0$ harnesses the power of network coding by selecting routes appropriately.

Next, we make the following observation about network coding. Comparing Figures 3.10 and 3.11, we notice that at moderate to high loads (but when the load is within the capacity region of the no coding case), network coding increases delays slightly. We believe that this is due to fact that packets are stored in multiple queues under network coding at each node: for each next-hop neighbor, a queue for each previous-hop neighbor must be maintained. This seems to result in slower convergence of the routing table.

3.8 The Stability of the Network Under PARN

Our stability result uses the result in [62] and relies on the fact that the arrival rate on each link is less than the available capacity of the link.

We will now focus on the case of wireless networks without network coding. All variables in this section are assumed to be average values in the stationary regime of the corresponding variables in the shadow process. Let $\bar{\sigma}_{nj}^d$ denote the mean shadow traffic rate at link (nj) destined to d. Let $\bar{\mu}_{nj}$ and $\alpha_n^d(1 + \varepsilon)$ denote the mean service rate of link (nj) and the exogenous shadow traffic arrival rate destined to d at node n. Notice that ε comes from our strategy on shadow traffic. The flow conservation equation is as follows:

$$\alpha_n^d(1+\varepsilon) + \sum_{l:(ln)\in\mathcal{L}} \bar{\sigma}_{ln}^d = \sum_{j:(nj)\in\mathcal{L}} \bar{\sigma}_{nj}^d, \forall n, d \in \mathcal{N}. \tag{3.21}$$

The necessary condition on the stability of shadow queues is as follows:

$$\sum_{d\in\mathcal{N}} \bar{\sigma}_{nj}^d \leq \bar{\mu}_{nj}. \tag{3.22}$$

Since we know that the shadow queues are stable under the shadow queue algorithm, the expression (3.22) should be satisfied.

Now we focus on the real traffic. Suppose the system has an equilibrium distribution and let λ_{nj}^d be the mean arrival rate of real traffic at link (nj) destined to d. The splitting probabilities are expressed as follows:

$$\bar{P}_{nj}^d = \frac{\bar{\sigma}_{nj}^d}{\sum_{k\in\mathcal{N}} \bar{\sigma}_{nk}^d}, \text{where } \ d \neq n. \tag{3.23}$$

Thus, the mean arrival rates at a link satisfy the traffic equation:

$$\lambda_{nj}^d = \alpha_n^d \bar{P}_{nj}^d + \sum_{l:(ln)\in\mathcal{L}} \lambda_{ln}^d \bar{P}_{nj}^d, \forall (nj) \in \mathcal{L}, d \in \mathcal{N}, \tag{3.24}$$

where $d \neq n$.

The traffic intensity at link (nj) is expressed as

$$\rho_{nj} = \frac{1}{\bar{\mu}_{nj}} \sum_{d\in\mathcal{N}} \lambda_{nj}^d. \tag{3.25}$$

Now we will show $\rho_{nj} < 1$ for any link $(nj) \in \mathcal{L}$. Let $\lambda_{nj}^d = \bar{\sigma}_{nj}^d/(1+\varepsilon)$ for every $(nj) \in \mathcal{L}$, and substitute it into expression (3.24). It is easy to check that the candidate solution is valid by using expression (3.21). From (3.22), the traffic intensity at link (nj) is strictly less than 1 for any link $(nj) \in \mathcal{L}$:

$$\rho_{nj} = \frac{1}{\bar{\mu}_{nj}} \sum_{d \in \mathcal{N}} \lambda_{nj}^d = \frac{1}{(1+\varepsilon)\bar{\mu}_{nj}} \sum_{d \in \mathcal{N}} \bar{\sigma}_{nj}^d < 1. \tag{3.26}$$

Thus we have shown that the traffic intensity at each link is strictly less than 1.

The wireline network is a special case of a wireless network. Substitute the link capacity c_{nj} for $\bar{\mu}_{nj}$ and set ε to be zero, and stability follows directly. The stability of wireless networks with network coding is similar to the case of wireless network with no coding.

3.9 To Derive the Backpressure Algorithm in the Network Coding Case

Given a set of packet arrival rates that lie in the capacity region, our goal is to find routes for flows that use as few resources as possible. Thus, we formulate the following optimization problem for the network coding case.

$$\begin{aligned} \min \quad & \sum_{(nj) \in \bar{\mathcal{L}}} \mu_{nj,pp} + \sum_{(n|jl) \in \bar{\mathcal{L}}} \mu_{(n|jl)} \\ s.t. \quad & \mu_{nj,pp}^d + \mu_{nj,broad}^d \leq \sum_k \mu_{njk}^d + \sum_{d'} \sum_{k:k \neq n} \mu_{j|kn}^{d,d'} \\ & \sum_{f \in \mathcal{F}} x_f I_{\{b(f)=n, e(f)=d\}} \leq \sum_{f \in \mathcal{F}} \sum_{j \in \mathcal{N}} \mu_{fnj}^d \end{aligned} \tag{3.27}$$

Let $\{q_{njd}\}$ and $\{q_{0nd}\}$ be the Lagrange multipliers corresponding to the flow conservation constraints in problem (3.27). Appending the constraints to the objective, we get

$$
\begin{aligned}
&\min_{\boldsymbol{\mu}\in\Lambda'} \sum_{(nj)\in\bar{\mathcal{L}}} \mu_{nj,pp} + \sum_{(n|jl)\in\bar{\mathcal{L}}} \mu_{n|jl} + \sum_{n,d} q_{0nd}\Big[\sum_{f\in\mathcal{F}} x_f \mathbb{1}_{\{b(f)=n,e(f)=d\}} - \sum_{f\in\mathcal{F}}\sum_{j\in\mathcal{N}} \mu^d_{fnj}\Big] \\
&\quad + \sum_d \sum_{(nj)\in\bar{\mathcal{L}}} q_{njd}\Big[\mu^d_{nj,pp} + \mu^d_{nj,broad} - \sum_k \mu^d_{njk} - \sum_{d'}\sum_{k:k\neq n} \mu^{d,d'}_{j|kn}\Big] \\
&= \min_{\boldsymbol{\mu}\in\Lambda'} \Big(- \sum_{(nj)\in\bar{\mathcal{L}}} \sum_{l:(ln)\in\bar{\mathcal{L}}} \sum_d \mu^d_{lnj}\big(q_{lnd} - q_{njd} - 1\big) \qquad (3.28) \\
&\quad - \sum_{(n|jl)\in\bar{\mathcal{L}},j<l} \sum_{d,d'} \mu^{d,d'}_{n|jl}\big(q_{lnd} - q_{njd} + q_{jnd'} - q_{nld'} - 2\big) \\
&\quad - \sum_{(nj)\in\bar{\mathcal{L}}} \sum_d \sum_{f\in\mathcal{F}} \mu^d_{fnj}\big(q_{0nd} - q_{njd} - 1\big) + \sum_{n,d} q_{0nd} \sum_{f\in\mathcal{F}} x_f \mathbb{1}_{\{b(f)=n,e(f)=d\}}\Big).
\end{aligned}
$$

If the Lagrange multipliers are known, then the optimal $\boldsymbol{\mu}$ can be found by solving

$$
\max_{\boldsymbol{\mu}\in\Lambda'} \sum_{(nj)\in\bar{\mathcal{L}}} \mu_{nj,pp} w_{nj} + \sum_{(n|jl)\in\bar{\mathcal{L}},j<l} \mu_{n|jl} w_{n|jl} \qquad (3.29)
$$

where

$$
\begin{aligned}
w_{nj} &= \max_d \{w^d_{nj}\}, \\
w_{n|jl} &= \max_{d,d'} \{w^{d,d'}_{n|jl}\}, \\
w^{d,d'}_{n|jl} &= w^d_{lnj} + w^{d'}_{jnl} \\
w^d_{nj} &= \max_{l:(ln)\in\mathcal{L} \text{ or } l=0} w^d_{lnj} \\
w^d_{lnj} &= q_{lnd} - q_{njd} - 1.
\end{aligned}
$$

Similar to the update algorithm of q_{nd} in (3.7), we can derive the update algorithm to compute q_{njd} :

$$
\begin{aligned}
q_{njd}[t+1] \;=\; &\Big[q_{njd}[t] + \frac{1}{M}\big(\mu^d_{nj,pp} + \mu^d_{nj,broad} - \sum_k \mu^d_{njk} - \sum_{d'}\sum_{k:k\neq n} \mu^{d,d'}_{j|kn}\big) \qquad (3.30) \\
&+ \frac{1}{M}\big(\sum_{f\in\mathcal{F}} x_f I_{\{b(f)=n,e(f)=d\}} - \sum_{f\in\mathcal{F}}\sum_{j\in\mathcal{N}} \mu^d_{fnj}\big)\Big]^+.
\end{aligned}
$$

By choosing $1/M$ to be the step-size parameter, Mq_{njd} looks very much like a queue update equation. Replacing Mq_{njd} by p_{njd}, we get (3.15)-(3.18). It can be shown using the results in [57, 58] that the stochastic versions of the above equations are stable and that the average rates can approximate the solution to (3.27) arbitrarily closely.

3.10 Summary

The backpressure algorithm, while being throughput-optimal, is not useful in practice for adaptive routing since the delay performance can be very bad. In this chapter, we have presented an algorithm that routes packets on shortest hops when possible, and decouples routing and scheduling using a probabilistic splitting algorithm built on the concept of shadow queues introduced in [42, 41]. By maintaining a probabilistic routing table that changes slowly over time, real packets do not have to explore long paths to improve throughput; this functionality is performed by the shadow "packets." Our algorithm also allows extra link activation to reduce delays. The algorithm has also been shown to reduce the queueing complexity at each node and can be extended to optimally trade off between routing and network coding.

CHAPTER 4

CONNECTION-LEVEL SCHEDULING IN WIRELESS NETWORKS USING ONLY MAC-LAYER INFORMATION

In order to operate wireless systems efficiently, scheduling algorithms are needed to facilitate simultaneous transmissions of different users. Scheduling algorithms for wireless networks have been widely studied since Tassiulas and Ephremides [1] proposed the *MaxWeight* algorithm. The MaxWeight Scheduling (MWS) algorithm is throughput optimal in the sense that it can stabilize the queues of the network for the largest set of arrival rates possible without knowing the actual arrival rates. Max Weight works under very general conditions, but it does not consider *connection-level dynamics*. It considers *packet-level dynamics* assuming that there is a fixed set of users and that packets are generated by each user according to some stochastic process. Moreover, there is no notion of congestion control while most modern communication networks use some congestion control mechanism for fairness purposes or to avoid excessive congestion inside the network [64]. There is a rich body of literature on the packet-level stability of scheduling algorithms. Stability of wireless networks under connection-level dynamics has been studied in, e.g., [21], [22], [23], [24]. The implicit assumption in these works is that the algorithm can fully observe the queue lengths for different connections while, in practice, the scheduler is implemented as part of the MAC layer and thus can use only the MAC-layer queue lengths.

In this chapter, we are interested in the scenario where files/connections arrive into and depart from an ad hoc wireless network. Upon arrival of a file, a TCP connection is established which regulates the injection of packets to the MAC layer. The scheduling algorithm must determine which links can transmit packets at each time instant. When the transmission of a file ends, its corresponding TCP connection is closed and the file departs the system. If the scheduler has access to the total queue length at Transport and MAC layers, then it can use MaxWeight algorithm to achieve throughput optimality. However, this would lead to a poor delay performance because large files might get priority over many small files for long periods of time. An alternative is to implement a weight-based MAC algorithm as in [1], but use the MAC-layer queue; then large files will not dominate the service because files are stored at the Transport layer and only a few packets are released at a time to the MAC layer. However, it is not obvious that such a system will be throughput-optimal.

For the connection-level model, we show that by appropriately choosing weights which are functions of the MAC-layer queue and using the MaxWeight-type scheduling algorithms, throughput optimality is achieved. The only assumption about protocols that we make is that each TCP window size is at least one and that there is a maximum window size, both of which are true for all implementations of TCP. We make no other assumptions on the dynamics of the TCP window flow control mechanism. On the other hand, the fact that our scheduler uses only the MAC-layer information is consistent with the actual implementation, because in reality, the scheduler is part of the MAC layer and might not have access to the Transport layer. We will present simulations that verify the fact that our scheduling algorithm improves the delay performance.

We comment on the differences between our model and a related model considered in [65]. In [65], throughput-optimal scheduling algorithms have been derived for a connection-level model of a wireless network assuming that each link has access to the number of files waiting at the link. Here, we make no such assumption and use only MAC-layer queue information which is readily available.

4.1 System Model

4.1.1 Model of Wireless Network

Consider a wireless network consisting of a set of nodes where each node could be a source and/or a destination for another node. We assume single-hop communications, and time is slotted. Files arrive at each link according to an i.i.d. process with some file size distribution.

File arrival processes at different links are also independent. We consider two cases of file size distribution throughout this chapter.

- *Case 1 (Arbitrary Distribution with Bounded File Size):* File size distribution can be arbitrary, but any file size is bounded by a positive integer σ_{max}. We say a file is type-i file if the file initially contains i packets. The number of file types is bounded because the file size is bounded.

- *Case 2 (Mixed Geometric Distribution):* The detailed description of the mixed geometric distribution is as follows. There are K possible file types where files of type i are geometrically distributed with mean $1/\eta_i$ packets. A new file arrival at link l belongs to type i with probability p_{li}, $i = 1, 2, .., K$, where $\sum_{i=1}^{K} p_{li} = 1$. It is obvious that the file size distribution of a particular file is geometric if the file type is known. Our motivation for selecting such a file size distribution is to model the large variance in the file sizes in the Internet. It is believed, see e.g., [66], that most of the bytes are generated by long files while most of the flows are short flows. By controlling the probabilities p_{li}, for the same average file size, we can obtain distributions with a large range of variances.

Let λ_l denote the file arrival rate at link l. For simplicity, we can assume that files arrive according to an i.i.d Bernoulli process with rate λ_l. Furthermore, we assume that each link has a unit service rate, i.e., in each time slot, one packet could be successfully transmitted over a link. We use the notion of the *conflict graph* to capture the interference constraints. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote the conflict graph of the wireless network, where each vertex in $\mathcal{V}$ is a communication link in the wireless network. There is an edge $(l_1, l_2) \in \mathcal{E}$ between vertices l_1 and l_2 if simultaneous transmissions over communication links l_1 and l_2 are not successful. Therefore, at each time slot, the active links should form an independent set of $\mathcal{G}$, i.e., no two scheduled vertices can share an edge in $\mathcal{G}$. Let $N = |\mathcal{V}|$ denote the number of communication links in the wireless network.

Formally, a schedule can be represented by a vector $X = [x_l : l = 1, ..., N]$ such that $x_l \in \{0, 1\}$ and $x_i + x_j \leq 1$ for all $(i, j) \in \mathcal{E}$. A schedule can also be represented by a set s

of links such that $l \in s$ if $x_l = 1$ (and $l \notin s$ if $x_l = 0$). Let $\mathcal{M}$ denote the set of all feasible schedules. At each time slot, a feasible schedule is chosen by the scheduling algorithm based on the current network information. Let m_l denote the mean file size of files arriving at link l, and define the workload at link l by $\rho_l = \lambda_l m_l$. For example, if the file size has a mixed geometric distribution, the mean file size is $m_l = \sum_{i=1}^{K} p_{li}/\eta_i$. Then, the capacity region of the network is the set of all load vectors $\boldsymbol{\rho} = (\rho_1, \cdots, \rho_N)$ that make the network queues stable. It is well known, e.g., see [1], that, under our model, the capacity region is given by

$$\mathcal{C} = \Big\{ \boldsymbol{\rho} : \exists \boldsymbol{\mu} \in Co(\mathcal{M}) \text{ s.t. } \boldsymbol{\rho} < \boldsymbol{\mu} \Big\},$$

where $Co(\cdot)$ is the convex hull operator. When dealing with vectors, inequalities should be interpreted component-wise. A scheduling algorithm is called throughput-optimal if it stabilizes the queues in the network for any load vector inside the capacity region $\mathcal{C}$.

4.1.2 Models of Transport and MAC Layers

Upon arrival of a file at a source, a TCP-connection is established that regulates the injection of packets into the MAC layer. The transmission of MAC-layer packets is itself controlled by the scheduling algorithm. Once transmission of a file ends, the file departs and the corresponding TCP-connection will be closed.

At each link, we index the files according to their arrival order such that the index 1 is given to the earliest file. This means that once transmission of a file ends, the indices of the remaining files are updated such that indices again start from 1 and are consecutive. Note that the indexing rule is not part of the algorithm implementation and it is used here only for the purpose of analysis.

Let $W_{li}[t]$ denote the congestion window size of file i at link l at time t. In general, W_{li} is a time-varying sequence which changes as a result of TCP congestion control. But, we can see it later, to simplify the analysis we also consider that the congestion window size is constant. It is worth noting that $W_{li}[t]$ is still time-varying because the subscript i refers to

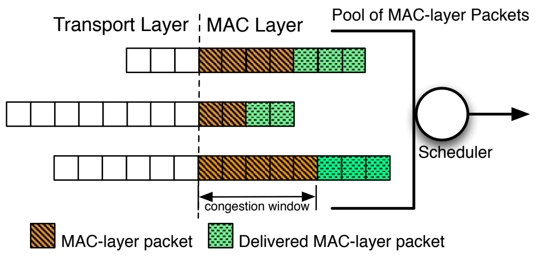

Figure 4.1: Transport/MAC layers: the packets at the MAC layer need not be in separate queues as shown above, they can be in a single queue.

a different file once a file with a smaller index departs. If the congestion window of file i is not full, TCP will continue injecting packets from the remainder of file i to the congestion window until file i has no packets remaining at the Transport layer (see Figure 4.1). It is important to note that the MAC layer does not know the number of remaining packets at the Transport layer, so any scheduling decision has to be made based on the MAC-layer information only. It is reasonable to assume that $W_{li}[t] \geq 1$.

Let $q_{li}[t]$ and $q_{li}^{mac}[t]$ denote the number of remaining packets and the number of MAC-layer packets of file i at link l at time t respectively. Let $n_l[t]$ denote the number of files at link l at time t. Therefore, the total number of packets at link l is $q_l[t] = \sum_{i=1}^{n_l[t]} q_{li}[t]$, and the total number of MAC-layer packets at link l is $q_l^{mac}[t] = \sum_{i=1}^{n_l[t]} q_{li}^{mac}[t]$. Based on our model so far, we have $n_l[t] \leq q_l^{mac}[t] \leq q_l[t]$, where the lower bound follows from the fact that each window size is at least one. Let $\boldsymbol{q}_l[t] = \{q_{li}[t]\}_i$ denote the vector of queue lengths of files at link l, and $\boldsymbol{W}_l = \{W_{li}[t]\}_i$ denote the vector of congestion window sizes of files at link l.

4.1.3 The State of the Network

We aim to find a scheduling algorithm which only uses the MAC-layer information to stabilize the system when the traffic loads are within the capacity region. We not only allow the scheduling algorithm to choose the set of active links but also allow the algorithm to determine which MAC-layer packets to serve at each time instant. For example, our analysis is general enough to allow even service disciplines that are not FIFO (*first-in, first-out*) at each MAC-layer queue. However, we assume that the scheduling algorithm is such that the network state can be described by a Markov chain as we will specify now. To understand the state of the network, we analyze several cases under various file size distributions.

First, we assume that the file size distribution follows *Case 1*, i.e., *arbitrary distribution with bounded file size.* To simplify the analysis under *Case 1*, we make the assumption that packets will be injected into the MAC-layer immediately as long as the congestion window is not full and there are remaining packets at the Transport layer.

We describe the network state in detail under the following three scenarios.

1) The congestion window size for a file remains constant throughout the sojourn of the file in the network. The initial congestion window size is determined by the file type. At the MAC layer, packets are scheduled in random order.

The network state should keep track of the status of files at each link, including the number of remaining packets $q_{li}[t]$, the number of MAC-layer packets $q_{li}^{mac}[t]$ and the congestion window size $W_{li}[t]$. Due to the immediate injection assumption that we make, we have $q_{li}^{mac}[t] = q_{li}[t]$ if $q_{li}[t] \leq W_{li}[t]$ or $q_{li}^{mac}[t] = W_{li}[t]$ if $q_{li}[t] > W_{li}[t]$. Therefore, $q_{li}^{mac}[t]$ is known if $q_{li}[t]$ and $W_{li}[t]$ are known. Therefore, the state of link l is $\mathcal{S}_l[t] = \{\boldsymbol{q}_l[t], \boldsymbol{W}_l[t]\}$, and the network state is $\mathcal{S}[t] = \{\mathcal{S}_l[t]\}_l$. At each time instant, the network state changes because $\boldsymbol{q}_l[t]$ changes and the reindexing happens if any file departs the network. Upon file arrival, the network state changes because the number of files changes, and new elements will be appended to $\boldsymbol{q}_l[t]$ and $\boldsymbol{W}_l[t]$.

2) The congestion window size for a file remains constant throughout the sojourn of the file in the network. The initial congestion window size is determined by the file type. Packets at the MAC layer are scheduled in FIFO order.

Similar to the previous scenario, the state of link l should include $\boldsymbol{q}_l[t]$ and $\boldsymbol{W}_l[t]$. But in order to schedule MAC-layer packets in FIFO order, we need to maintain the identity of MAC-layer packets too (see Figure 4.2). Let $\mathcal{I}_l[t] = \{f_{id}^1, f_{id}^2, f_{id}^3, f_{id}^4, \cdots\}$ denote which MAC-layer packets belong to which file in FIFO sequence; e.g., the head of line MAC-layer packets belong to file f_{id}^1. Therefore, the state of link l is $\mathcal{S}_l[t] = \{\boldsymbol{q}_l[t], \boldsymbol{W}_l[t], \mathcal{I}_l[t]\}$, and the

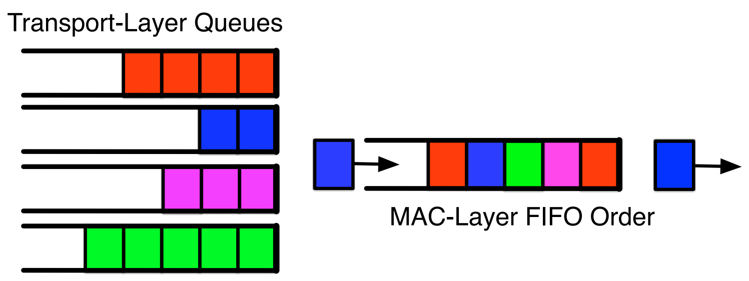

Figure 4.2: FIFO scheduling.

network state is $\mathcal{S}[t] = \{\mathcal{S}_l[t]\}_l$. Upon packet departures and file arrivals, the MAC-layer packet ordering $\mathcal{I}_l[t]$ is updated.

3) The congestion window size is time-varying. MAC-layer packets can be scheduled either in random order or FIFO order.

Obviously, the network state should include $\boldsymbol{q}_l[t]$ and $\boldsymbol{W}_l[t]$ as before. The network state should also include $\mathcal{I}_l[t]$ if the MAC-layer scheduling order is FIFO order. In reality, the congestion window size decreases if no ACK is received, and the congestion window size increases if all packets in the window are successfully transmitted. Window sizes are updated as a function of the congestion level in the network. Congestion information can be extracted from $\boldsymbol{q}_l[t]$ and $\boldsymbol{W}_l[t]$. If we use FIFO order as the MAC-layer scheduling order, the network state is $\mathcal{S}[t] = \{\mathcal{S}_l[t]\}_l$, where $\mathcal{S}_l[t] = \{\boldsymbol{q}_l[t], \boldsymbol{W}_l[t], \mathcal{I}_l[t]\}$. Let $\boldsymbol{A}_l[t]$ denote the vector of files arriving at link l at time t. The dynamics of the congestion window sizes can be written as $\boldsymbol{W}_l[t] = h_l(\mathcal{S}[t-1], \boldsymbol{A}_l[t])$, where $h_l(\cdot)$ is a corresponding mapping function to update the congestion window size based on the previous network state and new file arrivals.

From the above three scenarios, we can see that, under each specific scheduling algorithm which only uses the MAC-layer information, we can define a network state such that this state evolves according to a Markov chain.

Similarly, we show examples to indicate how the network state is defined when the file size distribution follows *Case 2*, i.e., the mixed geometric distribution. Let $\xi_{li}[t]$ be an indicator function which indicates whether there are packets of file i at the Transport layer or not; i.e., if $\xi_{li}[t] = 1$, the last packet of file i has not been injected to the MAC layer; if $\xi_{li}[t] = 0$, then there is no remaining packet of file i at the Transport layer. Let $\tau_i[t] \in \{1, ..., K\}$ denote the type of file with index i at time t. It is worth mentioning that the number of remaining packets of file i at the Transport layer follows a geometric distribution with mean $\sigma_{li}[t] = 1/\eta_{\tau_i[t]}$, as long as $\xi_{li}[t] = 1$, due to the memoryless property of the geometric distribution. Note that $\sigma_{li}[t]$ is a function of time only because of re-indexing since a file might change its index from slot to slot. The network states are slightly different under the following cases. Under all cases, we assume the congestion window size $W_{li} \geq 1$.

1) The congestion window size for a file remains constant throughout the sojourn of the file in the network. The initial congestion window size is determined by the file type. At the MAC layer, packets are scheduled in random order.

The network state should keep track of the status of files at each link, including the Transport layer information $\xi_{li}[t]$, the number of MAC-layer packets $q_{li}^{mac}[t]$, the congestion window size $W_{li}[t]$ and file type $\tau_i[t]$. Actually, keeping $\sigma_{li}[t]$ is equivalent to keeping $\tau_i[t]$ because they are one-to-one mapping. Therefore, the state of link l is $\mathcal{S}_l[t] = \{\xi_l[t], \boldsymbol{q}_l^{mac}[t], \boldsymbol{W}_l[t], \boldsymbol{\sigma}_l[t]\}$, and the network state is $\mathcal{S}[t] = \{\mathcal{S}_l[t]\}_l$.

2) The congestion window size for a file remains constant throughout the sojourn of the file in the network. The initial congestion window size is determined by the file type. Packets at the MAC layer are scheduled in FIFO order.

Similar to the previous scenario, the network state should keep track of the status of each file, but it is also necessary to maintain the identity of MAC-layer packets. Let $\mathcal{I}_l[t] = \{f_{id}^1, f_{id}^2, f_{id}^3, f_{id}^4, \cdots\}$ denote which MAC-layer packets belong to which file in FIFO sequence. The network state is $\mathcal{S}[t] = \{\mathcal{S}_l[t]\}_l$, where $\mathcal{S}_l[t] = \{\xi_l[t], \boldsymbol{q}_l^{mac}[t], \boldsymbol{W}_l[t], \boldsymbol{\sigma}_l[t], \mathcal{I}_l[t]\}$.

3) The congestion window size is time-varying. MAC-layer packets can be scheduled either in random order or FIFO order.

Similarly, the network state is $\mathcal{S}[t] = \{\mathcal{S}_l[t]\}_l$, where $\mathcal{S}_l[t] = \{\xi_l[t], \boldsymbol{q}_l^{mac}[t], \boldsymbol{W}_l[t], \boldsymbol{\sigma}_l[t], \mathcal{I}_l[t]\}$. The dynamics of the congestion window sizes can be written as $\boldsymbol{W}_l[t] = h_l(\mathcal{S}[t-1], \boldsymbol{A}_l[t])$, where $h_l(\cdot)$ is a corresponding mapping function to update the congestion window size based on the previous network state and new file arrivals.

Notice that the way to schedule MAC-layer packets is not limited to either random order scheduling or FIFO order scheduling. The MAC-layer packet scheduling method could be arbitrary as long as the corresponding Markov chain is well defined.

4.2 A MaxWeight-Type Scheduling Algorithm

4.2.1 Algorithm Description

Define a function $f(x)$ as

$$f(x) := \frac{\log(1+x)}{g(x)}, \tag{4.1}$$

where $g(x)$ is an arbitrary increasing function which makes $f(x)$ an increasing concave function. Assume that $g(0) > 0$ and $f(x)$ is a continuously differentiable function on $[0, \infty)$. Our *scheduling algorithm* is as follows:

- We assign a weight of $f(q_l^{mac}[t])$ to each link l. At each time instant t, the algorithm picks a schedule $\tilde{s}[t]$ such that

$$\tilde{s}[t] \in \arg\max_{s \in \mathcal{M}} \sum_{l \in s} f(q_l^{mac}[t]). \tag{4.2}$$

- If link l is scheduled, we choose a MAC-layer packet at link l to transmit. The scheduling decision within link l can be based on some arbitrary service discipline, for example, FIFO or random selection.

Recall that $q_l^{mac}[t]$ is the total number of packets at the MAC-layer of link l at time t. Therefore, the scheduling decision is only based on MAC-layer information.

Theorem 4.2.1. *For any $\varepsilon > 0$, the MaxWeight-type algorithm can stabilize the network for all $\boldsymbol{\rho} \in \mathcal{C}/(1+\varepsilon)$, independent of the Transport layer algorithm (as long as the minimum* *window size is one and the window sizes are bounded) and the (non-idling) service discipline used to transmit packets from active links.*

In particular, one can implement service disciplines that give priority to packets from short files if such information can be made available to the MAC layer. Such algorithms are often used in practice to reduce file transfer delays of short files. Notice that the purpose of choosing weight functions as in (4.1) is to achieve the throughput optimality by only using the MAC-layer information. Furthermore, such weight functions enable us to implement a fully distributed version of the algorithm using CSMA. But this is not considered in my dissertation.

4.3 Proof of Throughput-Optimality under Case 1

Throughout this section, we assume the file size distribution can be arbitrary but bounded. We will show throughput optimality of our scheduling algorithm which only needs to observe the MAC-layer information. By choosing the weight function $f(x)$ properly as in (4.1), we argue that the difference between the link weight using complete queue length information and the weight using queue length information only at the MAC-layer is sufficiently small. Therefore, when we show the stability using Lyapunov stability analysis, we will find the negative drift induced by our scheduling algorithm is close to that using the traditional MaxWeight algorithm.

Lemma 4.3.1. *The number of packets $q_l[t]$ and the number of files $n_l[t]$ at link l satisfy the following inequality:*

$$0 \leq f(q_l[t]) - f(n_l[t]) \leq c_0, \forall l, t,$$

where $c_0 = \log \sigma_{max}/g(0) > 0$.

Proof. Because any file size σ satisfies the inequality $1 \leq \sigma \leq \sigma_{max}$, we have

$$n_l[t] \leq q_l[t] \leq \sigma_{max} n_l[t].$$

a) Since $f(x)$ is an increasing function, we have $f(n_l[t]) \leq f(q_l[t])$.

b) Also because $g(x)$ is an increasing function, we have

$$\begin{aligned} f\big(q_l[t]\big) &\leq f\big(\sigma_{max} n_l[t]\big) = \frac{\log\big(1+\sigma_{max} n_l[t]\big)}{g(\sigma_{max} n_l[t])} \\ &\leq \frac{\log\big(\sigma_{max}+\sigma_{max} n_l[t]\big)}{g(n_l[t])} \\ &\leq f\big(n_l[t]\big) + \frac{\log \sigma_{max}}{g(0)}. \end{aligned}$$

Letting $c_0 = \log \sigma_{max}/g(0)$, we have proved the lemma.

□

Lemma 4.3.2. *The number of packets $q_l[t]$ and the number of MAC-layer packets $q_l^{mac}[t]$ at link l satisfy the following ineqaulity:*

$$0 \leq f(q_l[t]) - f(q_l^{mac}[t]) \leq c_0, \forall l, t,$$

where $c_0 = \log \sigma_{max}/g(0) > 0$.

Proof. This lemma directly follows Lemma 4.3.1. Because $n_l[t] \leq q_l^{mac}[t] \leq q_l[t]$, we have $0 \leq f(q_l[t]) - f(q_l^{mac}[t]) \leq f(q_l[t]) - f(n_l[t]) \leq c_0$. □

Recall that the state of link l is $\mathcal{S}_l[t] = \{\boldsymbol{q}_l[t], \boldsymbol{q}_l^{mac}[t], \boldsymbol{W}_l[t], \mathcal{I}_l[t]\}$ where $\boldsymbol{q}_l[t] = \{q_{li}[t]\}_i$, $\boldsymbol{q}_l^{mac}[t] = \{q_{li}^{mac}[t]\}_i$, and $\boldsymbol{W}_l[t] = \{W_{li}[t]\}_i$. And the network state $\mathcal{S}[t]$ is the set of all link states, i.e., $\mathcal{S}[t] = \{\mathcal{S}_l[t]\}_l$. $\mathcal{S}[t]$ is the state of a discrete-time Markov chain of the network, and we will now show that the network is stable, i.e., this Markov chain with state $\mathcal{S}[t]$ is positive recurrent.

Let $F(q) = \int_0^q f(x)dx$. We define the Lyapunov function $V(\mathcal{S})$ as follows:

$$V(\mathcal{S}) = \sum_{l=1}^{L} F(q_l). \tag{4.3}$$

Notice that the number of packets at link l is determined by the state $\mathcal{S}$. Recall the queue evolution at link l:

$$q_l[t+1] \;=\; \big[q_l[t]-x_l[t]\big]^+ + a_l[t] = q_l[t]-x_l[t]+a_l[t]+u_l[t], \tag{4.4}$$

where $[x]^+ = \max\{x,0\}$, $a_l[t]$ is the number of packets that arrives at link l at time t, $x_l[t]$ is the service rate provided by link l determined by the corresponding schedule and $u_l[t]$ is the wasted service at time t where $u_l[t] = \max\{x_l[t]-q_l[t],0\}$. The mean packet arrival rate is the workload, i.e., $\mathbb{E}(a_l[t]) = \rho_l = \lambda_l\sigma_l$. For simplicity of notation, we denote $\mathbb{E}_{\mathcal{S}}[\cdot]$ to be the expectation conditioned on the network state $\mathcal{S}[t]$, i.e., $\mathbb{E}_{\mathcal{S}}[\cdot] = \mathbb{E}[\cdot|\mathcal{S}[t]]$.

Now we try to calculate the Lyapunov drift. Define the 1-step Lyapunov drift to be $\Delta V(\mathcal{S}[t]) = V(\mathcal{S}[t+1]) - V(\mathcal{S}[t])$. From the mean-value theorem, we have

$$\begin{aligned}
\mathbb{E}_{\mathcal{S}}\big[\Delta V(\mathcal{S}[t])\big] &= \mathbb{E}_{\mathcal{S}}\big[V(\mathcal{S}[t+1])\big] - V(\mathcal{S}[t]) = \sum_{l=1}^{N}\Big(E_{\mathcal{S}}\big[F(q_l[t+1])\big] - F(q_l[t])\Big)\\
&= \sum_{l=1}^{N}\mathbb{E}_{\mathcal{S}}\big[f(y_l[t])(a_l[t]-x_l[t]+u_l[t])\big]\\
&= \sum_{l=1}^{N}\mathbb{E}_{\mathcal{S}}\Big[\big(f(y_l[t])-f(q_l[t])\big)(a_l[t]-x_l[t])\Big] + \sum_{l=1}^{N}\mathbb{E}_{\mathcal{S}}\Big[f(q_l[t])(a_l[t]-x_l[t])\Big]\\
&\quad + \sum_{l=1}^{N}\mathbb{E}_{\mathcal{S}}\big[f(y_l[t])u_l[t]\big],
\end{aligned} \tag{4.5}$$

where $y_l[t]$ is some value between $q_l[t]$ and $q_l[t+1]$. The following two lemmas provide upper bounds on the third term and the first term in the last step of (4.5).

Lemma 4.3.3. $\sum_{l=1}^{N}\mathbb{E}_{\mathcal{S}}\big[f(y_l[t])u_l[t]\big] \le c_1,\ \forall t, \mathcal{S}[t]$, *for some positive constant* c_1.

Proof. Let μ_l^c denote the link capacity of link l. Under our assumption in Section 4.1.1, $\mu_l^c = 1$. If $x_l[t] = \mu_l^c$, link l is scheduled at time t, and if $x_l[t] = 0$, link l is not scheduled. Because $u_l[t] = \max\{x_l[t]-q_l[t],0\}$, then

$$f(y_l[t])u_l[t] = \begin{cases} f(y_l[t])(x_l[t]-q_l[t]), & \text{if } x_l[t] > q_l[t]\\ 0, & \text{else.}\end{cases}$$

From the queue dynamics equation (4.4), $q_l[t+1] = a_l[t]$ if $x_l[t] > q_l[t]$. Therefore, $y_l[t]$ is somewhere between $q_l[t]$ and $a_l[t]$ if $x_l[t] > q_l[t]$. Because $f(x)$ is a nonnegative increasing function, we have $f(y_l[t]) \leq f(a_l[t]) + f(q_l[t]) \leq f(a_l[t]) + f(\mu_l^c)$ when $x_l[t] > q_l[t]$. Therefore, $f(y_l[t])u_l[t] \leq \big(f(a_l[t]) + f(\mu_l^c)\big)\mu_l^c$ if $x_l[t] > q_l[t]$. Now we bound $\mathbb{E}_{\mathcal{S}}\Big[f(y_l[t])u_l[t]\Big]$. Using previous results and the concavity of $f(x)$,

$$\begin{aligned}
\mathbb{E}_{\mathcal{S}}\Big[f(y_l[t])u_l[t]\Big] &= \mathbb{E}_{\mathcal{S}}\Big[f(y_l[t])u_l[t]\Big|x_l[t] \leq q_l[t]\Big]P\Big[x_l[t] \leq q_l[t]\Big|\mathcal{S}[t]\Big] \\
&\quad +\mathbb{E}_{\mathcal{S}}\Big[f(y_l[t])u_l[t]\Big|x_l[t] > q_l[t]\Big]P\Big[x_l[t] > q_l[t]\Big|\mathcal{S}[t]\Big] \\
&\leq 0 + \mathbb{E}_{\mathcal{S}}\Big[f(y_l[t])u_l[t]\Big|x_l[t] > q_l[t]\Big] \cdot 1 \\
&\leq \mathbb{E}_{\mathcal{S}}\Big[\Big(f(a_l[t]) + f(\mu_l^c)\Big)\mu_l^c\Big] \\
&\leq \mu_l^c f\big(E\big[a_l[t]\big]\big) + \mu_l^c f(\mu_l^c) \\
&= \mu_l^c\Big(f(\rho_l) + f(\mu_l^c)\Big).
\end{aligned}$$

In the previous inequality, we use the fact that $P\Big[x_l[t] \leq q_l[t]\Big|\mathcal{S}[t]\Big] \leq 1$ and $P\Big[x_l[t] > q_l[t]\Big|\mathcal{S}[t]\Big] \leq 1$. Letting $c_1 = \sum_{l=1}^N \mu_l^c\Big(f(\rho_l) + f(\mu_l^c)\Big)$, we have proved the lemma.

$\square$

Lemma 4.3.4. *Under the assumption that* $\mathbb{E}\big[a_l[t]^2\big] < \infty$, *the following inequality is satisfied by choosing a proper positive constant* c_2,

$$\sum_{l=1}^N \mathbb{E}_{\mathcal{S}}\Big[\big(f(y_l[t]) - f(q_l[t])\big)(a_l[t] - x_l[t])\Big] \leq c_2, \forall t, \mathcal{S}[t].$$

Proof. Because the wasted service cannot be greater than the departure, i.e., $x_l[t] \geq u_l[t]$, and $f(x)$ is a concave increasing function, we have

$$\begin{aligned}
\Big|f(y_l[t]) - f(q_l[t])\Big| &\leq f'(0)\Big|q_l[t+1] - q_l[t]\Big| \\
&= f'(0)\Big|a_l[t] - x_l[t] + u_l[t]\Big| \\
&\leq f'(0)\max\{a_l[t], \mu_l^c\}.
\end{aligned}$$

Because $\Big|a_l[t] - x_l[t]\Big| \le \max\{a_l[t], \mu_l^c\}$, we have

$$\begin{aligned} & \sum_{l=1}^{N} \mathbb{E}_{\mathcal{S}}\Big[\big(f(y_l[t]) - f(q_l[t])\big)(a_l[t] - x_l[t])\Big] \\ \le & \sum_{l=1}^{N} \mathbb{E}_{\mathcal{S}}\Big[\big|f(y_l[t]) - f(q_l[t])\big|\big|a_l[t] - x_l[t]\big|\Big] \\ \le & \sum_{l=1}^{N} \mathbb{E}\Big[f'(0)(\max\{a_l[t]^2, (\mu_l^c)^2\})\Big] \\ \le & f'(0)\sum_{l=1}^{N}\Big[\mathbb{E}\big[a_l[t]^2\big] + (\mu_l^c)^2\Big]. \end{aligned}$$

Letting $c_2 = f'(0)\sum_{l=1}^{N}\Big[c_{l2} + (\mu_l^c)^2\Big]$ where c_{l2} satisfies $\mathbb{E}\big[a_l[t]^2\big] < c_{l2} < \infty$, we have proved the lemma.

□

Substituting Lemma 4.3.3 and Lemma 4.3.4 into (4.5) and letting $c_{12} = c_1 + c_2$, the Lyapunov drift can be written as:

$$\begin{aligned} \mathbb{E}_{\mathcal{S}}[\Delta V(\mathcal{S}[t])] &\le \sum_{l=1}^{N} \mathbb{E}_{\mathcal{S}}\Big[f(q_l[t])(a_l[t] - x_l[t])\Big] + c_{12} \\ &= \sum_{l=1}^{N} f(q_l[t])\rho_l - \mathbb{E}_{\mathcal{S}}\Big[\sum_{l\in\tilde{s}[t]} f(q_l[t])\Big] + c_{12} \\ &= \sum_{l=1}^{N} f(q_l[t])\rho_l - \sum_{l\in\tilde{s}[t]} f(q_l[t]) + c_{12}, \end{aligned}$$

where $x_l[t] = 1$ if $l \in \tilde{s}[t]$, and $x_l[t] = 0$ if $l \notin \tilde{s}[t]$. Notice that $\mathbb{E}_{\mathcal{S}}\Big[\sum_{l\in\tilde{s}[t]} f(q_l[t])\Big] = \sum_{l\in\tilde{s}[t]} f(q_l[t])$ because the schedule $\tilde{s}[t]$ and $q_l[t]$ are known conditioned on the network state $\mathcal{S}[t]$. Let $s^*[t]$ be the optimal scheduling algorithm which maximizes the following objective:

$$s^*[t] \in \arg\max_{s\in\mathcal{M}} \sum_{l\in s} f(q_l[t]). \tag{4.6}$$

Now we bound the difference between $\sum_{l\in s^*[t]} f(q_l[t])$ and $\sum_{l\in\tilde{s}[t]} f(q_l[t])$. According to (4.2), we have

$$\sum_{l\in\tilde{s}[t]} f(q_l^{mac}[t]) - \sum_{l\in s^*[t]} f(q_l^{mac}[t]) \geq 0. \tag{4.7}$$

From Lemma 4.3.2 and expressions (4.21) and (4.7), we have

$$\begin{aligned} 0 \;\leq\; & \sum_{l\in s^*[t]} f(q_l[t]) - \sum_{l\in\tilde{s}[t]} f(q_l[t]) = \Big(\sum_{l\in s^*[t]} f(q_l[t]) - \sum_{l\in s^*[t]} f(q_l^{mac}[t])\Big) \\ & + \Big(\sum_{l\in\tilde{s}[t]} f(q_l^{mac}[t]) - \sum_{l\in\tilde{s}[t]} f(q_l[t])\Big) - \Big(\sum_{l\in\tilde{s}[t]} f(q_l^{mac}[t]) - \sum_{l\in s^*[t]} f(q_l^{mac}[t])\Big) \\ \leq\; & Nc_0 = N\log\sigma_{max}/g(0). \end{aligned} \tag{4.8}$$

The rest of the proof is standard. Assume that the traffic load is strictly within the capacity region, i.e., there exists $\varepsilon > 0$ such that $\boldsymbol{\rho} \in \Lambda/(1+\varepsilon)$. Therefore, $(1+\varepsilon)\boldsymbol{\rho} \in \Lambda$. The capacity region is an N-dimensional polyhedron which is a convex hull of all possible schedules. In linear programming, the optimal corner point of a linear optimization under the linear constraints is an optimal point, too; i.e.,

$$\max_{s\in\mathcal{M}} \sum_{l\in s} f\big(q_l[t]\big) = \max_{\boldsymbol{\mu}\in\Lambda} \sum_{l=1}^{N} f\big(q_l[t]\big)\mu_l.$$

Let $\boldsymbol{\mu}^*[t] = \arg\max_{\boldsymbol{\mu}\in\Lambda} \sum_{l=1}^{N} f\big(q_l[t]\big)\mu_l$, and we have the following inequality by definition:

$$\sum_{l\in s^*[t]} f\big(q_l[t]\big) = \sum_{l=1}^{N} f\big(q_l[t]\big)\mu_l^*[t] \geq \sum_{l=1}^{N} f\big(q_l[t]\big)\rho_l(1+\varepsilon).$$

We consider an N-dimensional sphere whose center is at the origin of the capacity region. Initially, the radius of the sphere is zero. We increase the radius gradually until the surface of the sphere touches the boundary of the capacity region Λ in the first quadrant. Let r^*

denote the radius of the sphere in the last step. The radius r^* is determined by the capacity region uniquely, and $r^* > 0$. Let $||f(\boldsymbol{q}[t])||_2 = \big(\sum_{l=1}^{N} f(q_l[t])^2\big)^{\frac{1}{2}}$, and let

$$\boldsymbol{\mu}^r[t] = \Big\{\frac{f(q_1[t])}{||f(\boldsymbol{q}[t])||_2}r^*, \frac{f(q_2[t])}{||f(\boldsymbol{q}[t])||_2}r^*, \cdots, \frac{f(q_N[t])}{||f(\boldsymbol{q}[t])||_2}r^*\Big\}.$$

From the way of creating the sphere, we know $\boldsymbol{\mu}^r[t] \in \Lambda$. From (4.22), the Lyapunov drift can be bounded by the following:

$$\begin{aligned}
\mathbb{E}_{\mathcal{S}}[\Delta V(\mathcal{S}[t])] &\leq \sum_{l=1}^{N} f(q_l[t])\rho_l - \sum_{l\in\tilde{s}[t]} f(q_l[t]) + c_{12} \leq \sum_{l=1}^{N} f(q_l[t])\rho_l - \sum_{l\in s^*[t]} f(q_l[t]) + c_3 \\
&\leq \sum_{l=1}^{N} f(q_l[t])\rho_l - \sum_{l=1}^{N} f(q_l[t])\frac{\mu_l^*[t]}{1+\varepsilon} - \frac{\varepsilon}{1+\varepsilon}\sum_{l=1}^{N} f(q_l[t])\mu_l^*[t] + c_3 \qquad (4.9) \\
&\leq -\frac{\varepsilon}{1+\varepsilon}\sum_{l=1}^{N} f(q_l[t])\mu_l^r[t] + c_3 \\
&= -\frac{\varepsilon}{1+\varepsilon}r^*||f(\boldsymbol{q}[t])||_2 + c_3,
\end{aligned}$$

where $c_3 = c_{12} + Nc_0$. The Lyapunov drift is negative as long as the queue length is sufficiently large. Let

$$\mathcal{B} = \Big\{\mathcal{S} : ||f(\boldsymbol{q})||_2 \leq \frac{1+\varepsilon}{\varepsilon r^*}(c_3 + \delta) \text{ for some } \delta > 0\Big\},$$

where $\boldsymbol{q}$ is a vector of number of packets at each link. From the definition of $\mathcal{B}$, any state in $\mathcal{B}$ must have a finite number of packets and a finite number of MAC-layer packets. $\mathcal{I}[t]$ can also be chosen from a finite set. Therefore, $\mathcal{B}$ contains only a finite set of states. Let $\mathcal{B}^c$ denote the complement of $\mathcal{B}$. For any state $\mathcal{S}[t] \in \mathcal{B}^c$, we have

$$E_{\mathcal{S}}[\Delta V(\mathcal{S}[t])] \leq -\frac{\varepsilon}{1+\varepsilon}r^*||f(\boldsymbol{q}[t])||_2 + c_3 \leq -\delta.$$

Therefore, we have proved throughput-optimality of our scheduling protocol by the Lyapunov stability theorem.

4.4 Proof of Throughput-Optimality under Case 2

Throughout this section, we make the assumption that the file size distribution is a mixture of geometric distributions. Since we use a discrete-time model, we have to specify the order in which files/packets arrive and depart, which we do below:

1. At the beginning of each time slot, a scheduling decision is made by the scheduling algorithm. Packets depart from the MAC layer of scheduled links.
2. File arrivals occur next. Once a file arrives, a new TCP connection is set up for that file with an initial pre-determined congestion window size.
3. For each TCP connection, if the congestion window is not full, packets are injected into the MAC layer from the Transport layer until the window size is fully used or there are no more packets at the Transport layer.

We re-index the files at the beginning of each time slot because some files might have departed during the last time slot.

Define $\bar{q}_l[t] := \mathbb{E}[q_l[t]|\mathcal{S}_l[t]]$ to be the expected queue length at link l given the state $\mathcal{S}_l[t]$. Then,

$$\bar{q}_l[t] = \sum_{i=1}^{n_l[t]} \Big[\sigma_{li}[t]\xi_{li}[t] + q_{li}^{mac}[t]\Big], \tag{4.10}$$

where $n_l[t]$ is the number of files at link l at the beginning of time slot t, which is known if the network state $\mathcal{S}[t]$ is given. Define $\Delta n_l[t]$ as the number of new files arriving at link l at time slot t. The dynamics of $\bar{q}_l[t]$ involve the dynamics of $\boldsymbol{q}_l^{mac}[t]$, $\boldsymbol{\xi}_l[t]$ and $n_l[t]$, and, thus, it consists of: (i) departure of MAC-layer packets, (ii) new file arrivals, (iii) injection of packets into the MAC layer, and (iv) departure of files from the Transport layer:

$$\bar{q}_l[t+1] = \bar{q}_l[t] - d_l^{mac}[t] + a_l[t] + a_l^{mac}[t] - d_l^{tcp}[t], \tag{4.11}$$

where $d_l^{mac}[t]$ is the number of packets that depart from the MAC layer, $a_l[t] = \sum_{i=n_l[t]+1}^{n_l[t]+\Delta n_l[t]} \sigma_{li}[t]$ is the expected number of packet arrivals of new files, $a_l^{mac}[t]$ is the total number of packets injected into the MAC layer to fill up the congestion window after scheduling and new file arrival, and $d_l^{tcp}[t] = \sum_{i=1}^{n_l[t]+\Delta n_l[t]} \sigma_{li}[t] I_{li}[t]$ is the Transport-layer "expected packet departure" because of the MAC-layer injection. Here, $I_{li}[t] = 1$ indicates the last packet of file i leaves the Transport layer during time slot t; otherwise, $I_{li}[t] = 0$. Recall that $\mathbb{E}\big[a_l[t]\big] = \rho_l$ is the mean packet arrival rate at link l.

Let $b_l[t] := a_l^{mac}[t] - d_l^{tcp}[t]$; then we rewrite (4.11) as

$$\begin{aligned} \bar{q}_l[t+1] &= \bar{q}_l[t] - d_l^{mac}[t] + b_l[t] + a_l[t], \\ &= \bar{q}_l[t] - x_l[t] + b_l[t] + a_l[t] + u_l[t], \end{aligned} \tag{4.12}$$

where $u_l[t] = \max\{x_l[t] - q_l^{mac}[t], 0\}$ is the wasted service, i.e., when l is included in the schedule but it does not have packets to transmit. Define $\mathbb{E}_{\mathcal{S}}[\cdot] = \mathbb{E}[\cdot|\mathcal{S}[t]]$. Lemma 4.4.1 characterizes the first and the second moments of $b_l[t]$.

Lemma 4.4.1. *For the process* $\{b_l[t]\}$,

(i) $\mathbb{E}_{\mathcal{S}}\Big[b_l[t]\Big] = 0$.

(ii) $\mathbb{E}_{\mathcal{S}}\big[b_l[t]^2\big] \le \big(\kappa_l + 1\big)/\eta_{min}^2$.

where $\eta_{min} = \min_{1 \le i \le K} \eta_i$.

Proof. Part(i):

Let $a_{li}^{mac}[t]$ denote the actual number of packets of file i injected into the MAC layer, and $d_{li}^{tcp}[t] = \sigma_{li}[t] I_{li}[t]$ denote the expected "packet departure" of file i from the Transport layer. We show a similar result for each individual file, i.e.,

$$\mathbb{E}_{\mathcal{S}}\Big[a_{li}^{mac}[t] - d_{li}^{tcp}[t]\Big] = 0. \tag{4.13}$$

The lemma is proved by summing up equations for all individual files. We only focus on existing files i where $\{i : \xi_{li}[t] = 1\}$ for or new files where $i \in \big(n_l[t]+1, n_l[t] + \Delta n_l[t]\big)$, because the equation (4.13) is automatically satisfied if no packet is in the Transport layer. Let $W^r_{li}[t]$ be the remaining window size of file i at link l after MAC-layer departure but before the MAC-layer injection. We want to show that

$$\mathbb{E}_{\mathcal{S}}\left[a^{mac}_{li}[t] - d^{tcp}_{li}[t]\middle|W^r_{li}[t] = w\right] = 0, \text{ for any } w \geq 0. \tag{4.14}$$

Notice that (4.14) implies (4.13). Because the number of remaining packets at the Transport layer is geometrically distributed with mean size $\sigma_{li}[t]$, the Transport layer will continuously inject packets into the MAC layer with probability $\gamma_{li} = 1 - 1/\sigma_{li}[t]$ as long as all previous packets are successfully injected and the window size is not full.

Clearly, if $w = 0$, no packet can be injected into the MAC layer. Therefore, $a^{mac}_{li}[t] = 0$ and $d^{tcp}_{li}[t] = 0$, and (4.14) is satisfied. Next, we consider the case when $w > 0$. Let $p_w(k, j)$ denote the probability that $a^{mac}_{li}[t] = k$ and $I_{li}[t] = j \in \{0, 1\}$ given that $W^r_{li}[t] = w$. Because Transport-layer packets are injected into the MAC layer as long as the window is not full, we have $p_w(k, 0) = 0$ for $k < w$. Obviously, $p_w(k, 1) = 0$ for $k > w$. The probability that $a^{mac}_{li}[t] = k$ where $k < w$ directly follows the geometric distribution of the remaining packets of file i, i.e.,

$$\begin{aligned} p_w(k,1) &= P(a^{mac}_{li}[t] = k, I_{li}[t] = 1|W^r_{li}[t] = w) = P(a^{mac}_{li}[t] = k|W^r_{li}[t] = w) \\ &= \gamma^{k-1}_{li}(1 - \gamma_{li}) \text{ for } 1 \leq k < w. \end{aligned}$$

The probability that the window size is full (i.e., $a^{mac}_{li}[t] = w$) and the last packet is injected into the MAC-layer after injection (i.e., $I_{li}[t] = 1$) is

$$p_w(w, 1) = P(a^{mac}_{li}[t] = w, I_{li}[t] = 1|W^r_{li}[t] = w) = \gamma^{w-1}_{li}(1 - \gamma_{li}).$$

From the definition of $I_{li}[t]$, we have

$$P(I_{li}[t] = 0|W^r_{li}[t] = w) = 1 - \sum_{k=1}^{w} p_w(k, 1) = \gamma^w_{li}.$$

Now we calculate the left-hand side of (4.14).

$$\begin{aligned}
& \mathbb{E}_{\mathcal{S}}\Big[a_{li}^{mac}[t]-d_{li}^{tcp}[t]\Big|W_{li}^{r}[t]=w\Big] \\
= & \sum_{k=1}^{w} p_w(k,1)\Big(k-\sigma_{li}[t]\Big)+P(I_{li}[t]=0|W_{li}^{r}[t]=w)w \\
= & \sum_{k=1}^{w} k\gamma_{li}^{k-1}(1-\gamma_{li})+(1-\gamma_{li}^{w})\sigma_{li}[t]+w\gamma_{li}^{w} \\
= & (1-\gamma_{li})\frac{d}{d\gamma_{li}}\Big[\frac{\gamma_{li}-\gamma_{li}^{w+1}}{1-\gamma_{li}}\Big]+\frac{1-\gamma_{li}^{w}}{1-\gamma_{li}}+w\gamma_{li}^{w} \\
= & 0.
\end{aligned}$$

Part(ii):

Let $b_{li}[t]=a_{li}^{mac}[t]-d_{li}^{tcp}[t]$ for file i. From the definition of $b_l[t]$, we have

$$b_l[t]=\sum_{i=1}^{n_l[t]} b_{li}[t]+\sum_{i=n_l[t]+1}^{n_l[t]+\Delta n_l[t]} b_{li}[t].$$

Using the fact that new arriving files are mutually independent, and are also independent of current network state, we have

$$\mathbb{E}_{\mathcal{S}}\Big[b_l[t]^2\Big]=\mathbb{E}_{\mathcal{S}}\left[\Big(\sum_{i=1}^{n_l[t]} b_{li}[t]\Big)^2\right]+\mathbb{E}_{\mathcal{S}}\left[\sum_{i=n_l[t]+1}^{n_l[t]+\Delta n_l[t]} b_{li}[t]^2\right]. \tag{4.15}$$

Notice that we also use the fact that $\mathbb{E}_{\mathcal{S}}\big[b_{li}[t]\big]=0$. Because $b_{li}[t]=a_{li}^{mac}[t]-d_{li}^{tcp}[t]$, $a_{li}^{mac}[t]\geq 0$ and $d_{li}^{tcp}[t]\geq 0$, we have $\big|b_{li}[t]\big|\leq\max\{a_{li}^{mac}[t],d_{li}^{tcp}[t]\}$ and therefore $b_{li}[t]^2\leq\max\{a_{li}^{mac}[t]^2,d_{li}^{tcp}[t]^2\}$. Because $b_{li}[t]^2\leq\max\{a_{li}^{mac}[t]^2,d_{li}^{tcp}[t]^2\}$, we have

$$\mathbb{E}_{\mathcal{S}}\Big[b_{li}[t]^2\Big]\leq\max\Big\{\mathbb{E}_{\mathcal{S}}\Big[a_{li}^{mac}[t]^2\Big],\mathbb{E}_{\mathcal{S}}\Big[d_{li}^{tcp}[t]^2\Big]\Big\}.$$

Because the number of packets of file i at the Transport layer is geometrically distributed with parameter $\sigma_{li}[t]$ and the number of packets injected into the MAC-layer $a_{li}^{mac}[t]$ cannot exceed the total number of packets at the Transport layer, therefore, $\mathbb{E}_{\mathcal{S}}\Big[a_{li}^{mac}[t]^2\Big]\leq\sigma_{li}[t]^2\leq 1/\eta_{min}^2$.

And because $\mathbb{E}_{\mathcal{S}}\Big[d_{li}^{tcp}[t]^2\Big] \leq \sigma_{li}[t]^2 \leq 1/\eta_{min}^2$, we have $\mathbb{E}_{\mathcal{S}}\Big[b_{li}[t]^2\Big] \leq 1/\eta_{min}^2$. The second term of (4.15) is bounded by

$$\mathbb{E}_{\mathcal{S}}\left[\sum_{i=n_l[t]+1}^{n_l[t]+\Delta n_l[t]} b_{li}[t]^2\right] \;\leq\; 1/\eta_{min}^2 \cdot \mathbb{E}_{\mathcal{S}}\Big[\Delta n_l[t]\Big] = \kappa_l/\eta_{min}^2.$$

Using Cauchy-Schwarz Inequality first term in (4.15)

$$\mathbb{E}_{\mathcal{S}}\left[\Big(\sum_{i=1}^{n_l[t]} b_{li}[t]\Big)^2\right] = \mathbb{E}_{\mathcal{S}}\left[\Big(\sum_{i\in\mathcal{D}_l[t]} b_{li}[t]\Big)^2\right] \leq |\mathcal{D}_l[t]| \cdot \mathbb{E}_{\mathcal{S}}\left[\sum_{i\in\mathcal{D}_l[t]} b_{li}[t]^2\right] \leq |\mathcal{D}_l[t]|^2/\eta_{min}^2 \leq 1/\eta_{min}^2.$$

Therefore, $\mathbb{E}_{\mathcal{S}}\Big[b_l[t]^2\Big] \leq (\kappa_l+1)/\eta_{min}^2$.

□

The weight of a link based on its MAC queue or the total expected queue length differs by a constant when weight is chosen carefully as stated by the following Lemma.

Lemma 4.4.2. *Let $f(x) = \log(1+x)/g(x)$, then*

$$0 \leq f(\bar{q}_l[t]) - f(q_l^{mac}[t]) \leq c_1', \tag{4.16}$$

where $c_1' = \log(1+1/\eta_{min})/g(0)$.

Proof. Because $q_l^{mac}[t] \leq \bar{q}_l[t]$ and $f(x)$ is an increasing function, the first inequality is straight-forward. From the definition of $\bar{q}_l[t]$ in (4.10),

$$\bar{q}_l[t] = \sum_{i=1}^{n_l[t]}\Big[\sigma_{li}[t]\xi_{li}[t] + q_{li}^{mac}[t]\Big] \leq 1/\eta_{min} n_l[t] + q_l^{mac}[t] \leq (1+1/\eta_{min})q_l^{mac}[t].$$

Therefore,

$$\begin{aligned} f(\bar{q}_l[t]) &\leq f\Big((1+1/\eta_{min})q_l^{mac}[t]\Big) = \frac{\log\Big(1+(1+1/\eta_{min})q_l^{mac}[t]\Big)}{g\Big((1+1/\eta_{min})q_l^{mac}[t]\Big)} \\ &\leq \frac{\log\Big((1+1/\eta_{min})(1+q_l^{mac}[t])\Big)}{g\Big(q_l^{mac}[t]\Big)} \leq f(q_l^{mac}[t]) + \frac{\log(1+1/\eta_{min})}{g(0)}. \end{aligned}$$

Letting $c_1' = \log(1+1/\eta_{min})/g(0)$ concludes the proof. □

Let $F(q) = \int_0^q f(x)dx$. We define the Lyapunov function $V(\mathcal{S})$ as follows:

$$V(\mathcal{S}) = \sum_{l=1}^{L} F(\bar{q}_l). \tag{4.17}$$

Next, we calculate the Lyapunov drift

$$\begin{aligned} \mathbb{E}_{\mathcal{S}}\big[V(\mathcal{S}[t+1]) - V(\mathcal{S}[t])\big] &= \sum_{l=1}^{N}\Big(\mathbb{E}_{\mathcal{S}}\big[F(\bar{q}_l[t+1])\big] - F(\bar{q}_l[t])\Big) \\ &= \sum_{l=1}^{N}\mathbb{E}_{\mathcal{S}}\big[f(y_l[t])(\tilde{a}_l[t] - x_l[t] + u_l[t])\big] \\ &= \sum_{l=1}^{N}\mathbb{E}_{\mathcal{S}}\Big[\big(f(y_l[t]) - f(\bar{q}_l[t])\big)(\tilde{a}_l[t] - x_l[t])\Big] \\ &\quad + \sum_{l=1}^{N}\mathbb{E}_{\mathcal{S}}\Big[f(\bar{q}_l[t])(\tilde{a}_l[t] - x_l[t])\Big] \\ &\quad + \sum_{l=1}^{N}\mathbb{E}_{\mathcal{S}}\big[f(y_l[t])u_l[t]\big], \end{aligned} \tag{4.18}$$

where, by the mean-value theorem, $y_l[t]$ is some value between $\bar{q}_l[t]$ and $\bar{q}_l[t+1]$ and $\tilde{a}_l[t] := a_l[t] + b_l[t]$.

The first term and the third term of (4.18) are bounded as stated by the following lemmas.

Lemma 4.4.3. *There exists a positive constant c_2 such that, for all $\mathcal{S}[t]$,*

$$\sum_{l=1}^{N} \mathbb{E}_{\mathcal{S}}\big[f(y_l[t])u_l[t]\big] \leq c_2'.$$

Proof. Recall that the wasted service is $u_l[t] = \mathbb{1}\{x_l[t] = 1, q_l^{mac}[t] = 0\}$, i.e., no service is wasted as long as $q_l^{mac}[t] \geq 1$. Therefore, since the congestion window size for every file is at least one, we know that $u_l[t] = 0$ if $n_l[t] \geq 1$. Hence, based on the definition of $\bar{q}_l[t]$, $u_l[t] = 0$ as long as $\bar{q}_l[t] \geq q_l^0 := 1 + 1/\eta_{min}$. Also recall that $y_l[t]$ is between $\bar{q}_l[t]$ and $\bar{q}_l[t+1]$ where $\bar{q}_l[t+1] = \bar{q}_l[t] + \tilde{a}_l[t] - d_l[t]$. Therefore, $y_l[t] \leq \max\{\bar{q}_l[t], \bar{q}_l[t] + \tilde{a}_l[t]\}$, and $f(y_l[t]) \leq f(\bar{q}_l[t]) + f(\bar{q}_l[t] + \tilde{a}_l[t])$.

Because $f(x)$ is a nonnegative concave increasing function,

$$\begin{aligned}
\mathbb{E}_{\mathcal{S}}\Big[f(y_l[t])u_l[t]\Big] &\leq \mathbb{E}_{\mathcal{S}}\Big[f(y_l[t])\mathbb{1}\{q_l[t] < q_l^0\}\Big] \\
&\leq \mathbb{E}_{\mathcal{S}}\Big[f(q_l^0) + f(q_l^0 + \tilde{a}_l[t])\Big] \\
&\leq \mathbb{E}_{\mathcal{S}}\Big[2f(q_l^0) + f'(q_l^0)\tilde{a}_l[t]\Big] \\
&\leq 2f(q_l^0) + f'(q_l^0)\rho_l = c_2'.
\end{aligned}$$

□

Lemma 4.4.4. *Assume $\mathbb{E}\big[a_l[t]^2\big] < \infty$; then there exists a positive constant c_3' such that*

$$\sum_{l=1}^{N} \mathbb{E}_{\mathcal{S}}\Big[\big(f(y_l[t]) - f(\bar{q}_l[t])\big)(\tilde{a}_l[t] - x_l[t])\Big] \leq c_3', \forall t, \mathcal{S}[t].$$

Proof. We first bound $\big|f(y_l[t]) - f(\bar{q}_l[t])\big|$. Because $f(x)$ is a concave increasing function and $u_l[t] \leq x_l[t]$,

$$\begin{aligned}
\Big|f(y_l[t]) - f(\bar{q}_l[t])\Big| &\leq f'(0)\Big|\bar{q}_l[t+1] - \bar{q}_l[t]\Big| \\
&= f'(0)\Big|-x_l[t] + u_l[t] + a_l[t] + b_l[t]\Big| \\
&\leq f'(0)\max\Big\{a_l[t] + b_l[t], x_l[t] - b_l[t]\Big\}.
\end{aligned}$$

Now we bound $\big|\tilde{a}_l[t] - x_l[t]\big|$:

$$\big|\tilde{a}_l[t] - x_l[t]\big| = \big|a_l[t] + b_l[t] - x_l[t]\big| \leq \max\Big\{a_l[t] + b_l[t], x_l[t] - b_l[t]\Big\}.$$

Therefore,

$$\begin{aligned}
& \sum_{l=1}^{N} \mathbb{E}_{\mathcal{S}}\Big[\big(f(y_l[t]) - f(\bar{q}_l[t])\big)(\tilde{a}_l[t] - x_l[t])\Big] \\
\leq\ & \sum_{l=1}^{N} \mathbb{E}_{\mathcal{S}}\Big[\big|f(y_l[t]) - f(\bar{q}_l[t])\big| \cdot \big|\tilde{a}_l[t] - x_l[t]\big|\Big] \qquad (4.19)\\
\leq\ & f'(0)\sum_{l=1}^{N} \mathbb{E}_{\mathcal{S}}\Big[\max\Big\{(a_l[t] + b_l[t])^2, (x_l[t] - b_l[t])^2\Big\}\Big] \\
\leq\ & f'(0)\sum_{l=1}^{N} \Big\{\mathbb{E}_{\mathcal{S}}\Big[\big(a_l[t] + b_l[t]\big)^2\Big] + \mathbb{E}_{\mathcal{S}}\Big[\big(x_l[t] - b_l[t]\big)^2\Big]\Big\} \\
\leq\ & 2f'(0)\sum_{l=1}^{N} \Big\{\mathbb{E}_{\mathcal{S}}\Big[a_l[t]^2\Big] + \mathbb{E}_{\mathcal{S}}\Big[x_l[t]^2\Big] + 2\mathbb{E}_{\mathcal{S}}\Big[b_l[t]^2\Big]\Big\},
\end{aligned}$$

where we have used the the Cauchy-Schwarz inequality in the last step. Note that $\mathbb{E}_{\mathcal{S}}\big[x_l[t]^2\big] \leq 1$, $\mathbb{E}_{\mathcal{S}}\big[b_l[t]^2\big] \leq \big(\kappa_l + 1\big)/\eta_{min}^2$ from Lemma 4.4.1, and $\mathbb{E}_{\mathcal{S}}\big[a_l[t]^2\big] = \mathbb{E}\big[a_l[t]^2\big] < c_{l5} < \infty$ for some positive c_{l3} by assumption. Therefore, letting

$$c_3' = 2f'(0)\sum_{l=1}^{N}\Big\{(1 + \big(\kappa_l + 1\big)/\eta_{min}^2 + c_{l3}\Big\}$$

concludes the proof. □

Hence, letting $c_{23} = c_2' + c_3'$, the Lyapunov drift can be bounded by

$$\begin{aligned}
\mathbb{E}_{\mathcal{S}}\big[\Delta V(\mathcal{S}[t])\big] &\leq \sum_{l=1}^{N} \mathbb{E}_{\mathcal{S}}\Big[f(\bar{q}_l[t])(\tilde{a}_l[t] - x_l[t])\Big] + c_{23} \\
&= \sum_{l=1}^{N} f(\bar{q}_l[t])\rho_l - \sum_{l\in\tilde{s}[t]} f(\bar{q}_l[t]) + c_{23}. \qquad (4.20)
\end{aligned}$$

Let $s^*[t]$ be the optimal scheduling algorithm which maximizes the following objective:

$$s^*[t] \in \arg\max_{s\in\mathcal{M}} \sum_{l\in s} f(\bar{q}_l[t]). \tag{4.21}$$

Then, based on definitions (4.2), (4.21), and Lemma 4.4.2, it can be shown that

$$0 \le \sum_{l\in s^*[t]} f(\bar{q}_l[t]) - \sum_{l\in \tilde{s}[t]} f(\bar{q}_l[t]) \le Nc_1'. \tag{4.22}$$

To complete the proof of Theorem 4.2.1, assume that the traffic load is strictly within the capacity region, i.e., there exists $\varepsilon > 0$ such that $\boldsymbol{\rho} \in \mathcal{C}/(1+\varepsilon)$. The capacity region is an N-dimensional polyhedron which is a convex hull of all possible schedules. In linear programming, one of the corner points of a linear optimization over the linear constraints is one of the optimal solutions. Hence, letting

$$\boldsymbol{\mu}^*[t] = \arg\max_{\boldsymbol{\mu}\in\mathcal{C}} \sum_{l=1}^{N} f\big(\bar{q}_l[t]\big)\mu_l, \tag{4.23}$$

the following holds:

$$\sum_{l\in s^*[t]} f\big(\bar{q}_l[t]\big) = \sum_{l=1}^{N} f\big(\bar{q}_l[t]\big)\mu_l^*[t] \ge \sum_{l=1}^{N} f\big(\bar{q}_l[t]\big)\rho_l(1+\varepsilon).$$

Consider the largest sphere centered at the origin and tangent to the boundary of the capacity region $\mathcal{C}$. Let r^* be the radius of such a sphere. The radius r^* is determined by the capacity region uniquely and $r^* > 0$. Define

$$\boldsymbol{\mu}^r[t] = \left(\frac{f(\bar{q}_1[t])}{||f(\bar{\boldsymbol{q}}[t])||_2} r^*, \cdots, \frac{f(\bar{q}_N[t])}{||f(\bar{\boldsymbol{q}}[t])||_2} r^*\right), \tag{4.24}$$

where $||.||_2$ is the Euclidian norm in $\mathbb{R}^N$. Note that $\boldsymbol{\mu}^r[t] \in \mathcal{C}$ by construction. From (4.22), the Lyapunov drift can be bounded as follows:

$$
\begin{aligned}
\mathbb{E}_{\mathcal{S}}[\Delta V(\mathcal{S}[t])] &\le \sum_{l=1}^{N} f(\bar{q}_l[t])\rho_l - \sum_{l\in\bar{s}[t]} f(\bar{q}_l[t]) + c_{23} \\
&\le \sum_{l=1}^{N} f(\bar{q}_l[t])\rho_l - \sum_{l\in s^*[t]} f(\bar{q}_l[t]) + c_4 \\
&\le \sum_{l=1}^{N} f(\bar{q}_l[t])\rho_l - \sum_{l=1}^{N} f(\bar{q}_l[t])\frac{\mu_l^*[t]}{1+\varepsilon} - \frac{\varepsilon}{1+\varepsilon}\sum_{l=1}^{N} f(\bar{q}_l[t])\mu_l^*[t] + c_4 \\
&\le -\frac{\varepsilon}{1+\varepsilon}\sum_{l=1}^{N} f(\bar{q}_l[t])\mu_l^r[t] + c_4 \\
&= -\frac{\varepsilon}{1+\varepsilon} r^* ||f(\bar{\boldsymbol{q}}[t])||_2 + c_4,
\end{aligned}
$$

where $c_4 = c_{23} + Nc_1'$. Therefore, the Lyapunov drift will be negative when $\|\bar{\boldsymbol{q}}\|$[1] is sufficiently large, or when the number of files is sufficiently large. Specifically, consider any constant $\delta > 0$ and let

$$
\mathcal{B}^c = \left\{ \mathcal{S} : \|\mathbf{n}\| \ge f^{-1}\left(\frac{1+\varepsilon}{\varepsilon r^*}(c_4 + \delta) \right) \right\}.
$$

Then, for any $\mathcal{S} \in \mathcal{B}^c$, the Lyapunov drift is less than $-\delta$. Also it is easy to check that $\mathcal{B}$ contains only a finite set of states with finite drift. Therefore, the system is stable by the Foster-Lyapunov stability theorem [67].

4.5 Simulation Results

In this section, we only show simulations for the case when the file size distribution is a mixture of geometric distributions. The result for the bounded file size case is very similar. Consider the simple wireless network of Figure 4.3 under the 1-hop interference model, which implies links sharing a node cannot be active simultaneously. There are 22 distinct maximal schedules for this network. Each link has a unit link capacity. The distribution of files is a

[1] When there is no subscript, $\|z\| = \|z\|_\infty = \max_i z_i = z_{max}$.

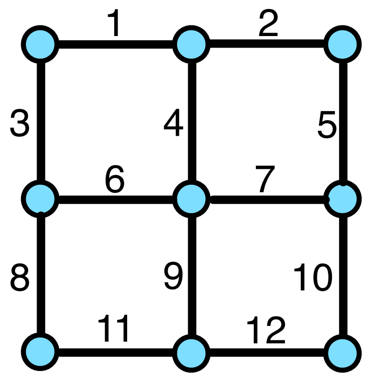

Figure 4.3: A wireless network containing 12 communication links.

mixture of geometric distributions as described next. Once a file is generated, the file size is geometrically distributed with mean 2 with probability 5/6 and is geometrically distributed with mean 100 with probability 1/6. All file arrivals are generated by Bernoulli processes. We say a file is short if its file size is less than half of the mean file size; otherwise, it is a long file. While this definition is somewhat arbitrary, our conclusions continue to hold if this assumption is removed. For simplicity, we divide the links into the following two sets: $\{4, 6, 7, 9\}$ and $\{1, 2, 3, 5, 8, 10, 11, 12\}$, where the links in each set have an equal arrival rate but the arrival rate of links in the second set is half of the arrival rate of links in the first set. We do simulations for different traffic intensities where the traffic intensity is a number such that the load vector divided by the traffic intensity lies on the boundary of the capacity region.

We consider a very naive window flow control algorithm under which the window size is always 1 for links $\{1, 4, 7, 10\}$, 2 for links $\{2, 5, 8, 11\}$ and 3 for links $\{3, 6, 9, 12\}$. We assume that MAC-layer packets are removed in a FIFO order. Once a packet is removed, a packet from the same file is injected from the Transport layer to the tail of the MAC-layer FIFO queue as long as there are packets at the Transport layer. In addition to FIFO, we also do simulations for the case in which the MAC layer receives one-bit information notifying whether the file is short or long. In this case, we give higher priority to short files. Each class (short or long) of files is served in a FIFO order. In the simulation, our scheduling

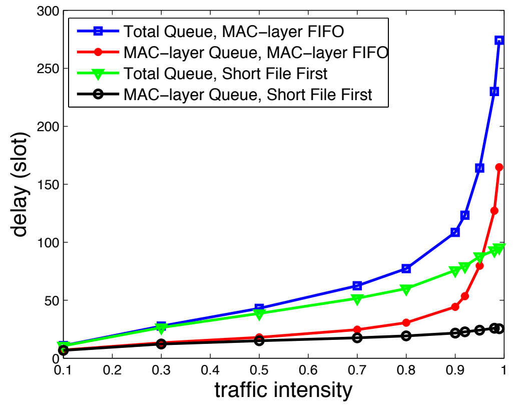

Figure 4.4: Average delay of short files.

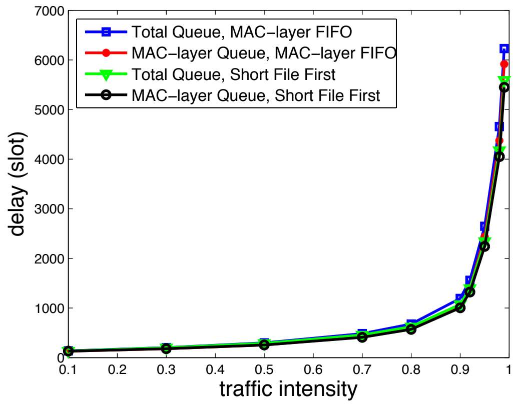

Figure 4.5: Average delay of long files.

algorithm chooses the weight of link l to be $f(q_l^{mac}[t])$ where $f(x) = \log(1 + x)$ by letting $g(x) = 1$. We compare our algorithm with the scheduling algorithm which uses $f(q_l[t])$ as the weight of link l.

Figures 4.4 and 4.5 show the average delays of short and long files for different traffic intensities. For both FIFO and the short-file-first service disciplines for the MAC-layer packet transmission, the short-file delay performance of the scheduling algorithm which only uses the MAC-layer queue information is much better than the performance of the regular maximum weight scheduling that uses the total number of packets as the weight of a link. This is because a large total queue length at a link does not necessarily imply a large number of files at that link. It is possible that the scheduling algorithm chooses a link with large queue length but containing only a few files. Therefore, a link which contains many short files, but has only a small number of packets, will be scheduled infrequently and short files at such links suffer high latency. Also, as we expect, if the MAC layer knows 1-bit additional information to identify short files and uses short-file-first in-link scheduling, the delay performance will be even better.

Figure 4.5 shows that the delay performance of long files is almost the same under both algorithms. However, for the same weight function, using 1-bit extra information can still yield a slightly better delay performance compared to the FIFO performance.

4.6 Summary

Since the scheduling algorithm is part of MAC, it is desirable to design an algorithm that uses only MAC-layer information. We showed that it is possible to design such algorithms that are still throughput optimal. Another advantage of such algorithms is that the long files do not block the transmission of short files and hence, the overall delay performance can be improved. The key element of the algorithm is an appropriate choice of a weight function. Interestingly, by using such weight functions, we can design a distributed version of the algorithm with proven throughput optimality [68].

CHAPTER 5

CONCLUSION

In this chapter, we briefly summarize the contributions of this thesis and suggest possible directions for future work. The main objective of the thesis is to design good resource allocation algorithms, specifically scheduling and routing algorithms, to achieve optimal throughput and good delay performance. For this purpose, we have suggested several new algorithms, and also studied the performance of previously proposed algorithms.

In Chapter 2, we have studied the workload optimality of scheduling algorithms for small generalized switches [69]. By providing scheduling algorithms other than MWS-α algorithm and proving the heavy-traffic sample-path workload optimality of our algorithms, we show that the well-known conjecture that MWS-α is workload optimal when α goes to zero is false. The future direction of this piece of work could be to find workload-optimal scheduling algorithms under more general network settings. However, the biggest challenge is that multiple resources are coupled even in the heavy traffic region in a more general setting.

In Chapter 3, a backpressure-based packet-by-packet adaptive routing and scheduling algorithm is proposed which has near-optimal throughput and good delay performance [70]. We also extend our results to the case with network coding and our algorithm determines the tradeoff between routing and coding automatically. A possible direction for future research is to analytically understand the tradeoff between the shadow queue parameter epsilon and the delay performance of the network.

We present a MaxWeight-type scheduling algorithm with single-hop file arrivals and departures in Chapter 4 [71, 68]. When considering the connection-level scenario, the stability of the network with our algorithm is not obvious. We show that our algorithm does achieve throughput optimality even though it uses only MAC-layer information for scheduling. The

MAC-layer packet service order can be arbitrary as long as one can define a network state which evolves as a Markov chain. It is also shown from simulations that the delay performance of our algorithm is much better compared with the traditional MaxWeight scheduling algorithm. A possible future direction could be to find a set of good scheduling algorithms when there are multi-hop routes in the network.

REFERENCES

[1] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.

[2] X. Wu and R. Srikant, "Regulated maximal matching: A distributed scheduling algorithm for multi-hop wireless networks with node-exclusive spectrum sharing," in *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference'05*, Dec. 2005, pp. 5342–5347.

[3] A. Dimakis and J. Walrand, "Sufficient conditions for stability of longest-queue-first scheduling: Second-order properties using fluid limits," *Advances in Applied Probability*, vol. 38, no. 2, pp. 505–521, June 2006.

[4] X. Wu, R. Srikant, and J. R. Perkins, "Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 6, pp. 595–605, June 2007.

[5] G. Zussman, A. Brzezinski, and E. Modiano, "Multihop local pooling for distributed throughput maximization in wireless networks," in *Proceedings of the 27th IEEE INFOCOM*, 2008, pp. 1139–1147.

[6] C. Joo, X. Lin, and N. B. Shroff, "Understanding the capacity region of the greedy maximal scheduling algorithm in multihop wireless networks," *IEEE/ACM Transaction on Networking*, vol. 17, no. 4, pp. 1132–1145, Aug. 2009.

[7] L. Jiang and J. Walrand, "A distributed csma algorithm for throughput and utility maximization in wireless networks," *IEEE/ACM Transaction on Networking*, vol. 18, no. 3, pp. 960–972, June 2010.

[8] J. Ni, B. Tan, and R. Srikant, "Q-CSMA: Queue-length based CSMA/CA algorithms for achieving maximum throughput and low delay in wireless networks," in *Proceedings of the 29th IEEE INFOCOM*, March 2010, pp. 1–5.

[9] S. Rajagopalan, D. Shah, and J. Shin, "Network adiabatic theorem: An efficient randomized protocol for contention resolution," in *Proceedings of ACM SIGMETRICS/Performance*, 2009, pp. 133–144.

[10] J. He and J. Rexford, “Toward internet-wide multipath routing,” *IEEE Network Magazine*, vol. 22, no. 2, pp. 16–21, March-April 2008.

[11] P. Narvaez, K. Y. Siu, and H. Y. Tzeng, “Efficient algorithms for multi-path link state routing,” in *Proceedings of ISCOM*, 1999.

[12] S. Vutukury and J. J. Garcia-Luna-Aceves, “A simple approximation to minimum-delay routing,” in *Proceedings of ACM SIGCOMM*, 1999, pp. 227–238.

[13] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala, “Path splicing,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 27–38, 2008.

[14] X. Yang and D. Wetherall, “Source selectable path diversity via routing deflections,” *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 159–170, 2006.

[15] D. Xu, M. Chiang, and J. Rexford, “DEFT: Distributed exponentially-weighted flow splitting,” in *Proceedings of the 26th IEEE INFOCOM 2007*, May 2007, pp. 71–79.

[16] L. Bui, R. Srikant, and A. L. Stolyar, “A novel architecture for reduction of delay and queueing structure complexity in the back-pressure algorithm,” *IEEE/ACM Transaction on Networking*, vol. 19, no. 6, pp. 1597–1609, Dec. 2011.

[17] M. J. Neely, E. Modiano, and C. E. Rohrs, “Dynamic power allocation and routing for time varying wireless networks,” *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 1, pp. 89–103, January 2005.

[18] A. L. Stolyar, “Maxweight scheduling in a generalized switch: State space collapse and workload minimization in heavy traffic,” *The Annals of Applied Probability*, vol. 14, no. 1, pp. 1–53, 2004.

[19] D. Shah and D. J. Wischik, “Switched networks with maximum weight policies: Fluid approximation and multiplicative state space collapse,” *The Annals of Applied Probability*, to be published.

[20] I. Keslassy and N. McKeown, “Analysis of scheduling algorithms that provide 100% throughput in input-queued switches,” in *Proceedings of the 39th Allerton Conference on Communications, Control and Computing*, Monticello, IL, 2001.

[21] T. Bonald, S. Borst, and A. Proutiere, “How mobility impacts the flow-level performance of wireless data systems,” in *Proceedings of the 23rd IEEE INFOCOM*, 2004, pp. 1872–1881.

[22] J. Liu, A. Proutiere, Y. Yi, M. Chiang, and V. Poor, “Flow-level stability of data networks with non-convex and time-varying rate regions,” in *Proceedings of ACM SIGMETRICS*, 2007, pp. 239–250.

[23] P. van de Ven, S. Borst, and S. Shneer, "Instability of maxweight scheduling algorithms," in *Proceedings of the 28th IEEE INFOCOM*, 2009, pp. 1701–1709.

[24] S. Liu, L. Ying, and R. Srikant, "Throughput-optimal opportunistic scheduling in the presence of flow-level dynamics," *IEEE/ACM Transaction on Networking*, vol. 19, no. 4, pp. 1057–1070, Aug. 2010.

[25] A. Eryilmaz, R. Srikant, and J. Perkins, "Stable scheduling policies for fading wireless channels," *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 411–424, 2005.

[26] V. J. Venkataramanan and X. Lin, "Structural properties of LDP for queue-length based wireless scheduling algorithms," in *Proceedings of the 45th Annual Allerton Conference on Communication, Control, and Computing*, 2007.

[27] L. Tassiulas and A. Ephremides, "Dynamic scheduling for minimum delay in tandem and parallel constrained queueing models," *Annals of Operation Research*, vol. 18, pp. 333–355, 1994.

[28] T. Ji, E. Athanasopoulou, and R. Srikant, "Optimal scheduling policies in small generalized switches," in *Proceedings of the 28th IEEE INFOCOM*, April 2009, pp. 2921–2925.

[29] J. M. Harrison, "Brownian models of queueing networks with heterogeneous customer populations," in *Stochastic Differential Systems, Stochastic Control Theory and Applications,* vol. 10*, The IMA Volumes in Mathematics and its Applications*, W. Fleming and P.-L. Lions, Eds. New York, NY: Springer-Verlag, 1988, pp. 147–186.

[30] M. I. Reiman, "Some diffusion approximations with state space collapse," in *Modelling and Performance Evaluation Methodology,* vol. 60*, Lecture Notes in Control and Information Sciences*, F. Baccelli and G. Fayolle, Eds. Berlin / Heidelberg: Springer, 1984, pp. 207–240.

[31] C. N. Laws, "Resource pooling in queueing networks with dynamic routing," *Advances in Applied Probability*, vol. 24, no. 3, pp. 699–726, 1992.

[32] F. P. Kelly and C. N. Laws, "Dynamic routing in open queueing networks: Brownian models, cut constraints and resource pooling," *Queueing Systems Theory and Applications*, vol. 13, no. 1-3, pp. 47–86, 1993.

[33] S. Shakkottai, R. Srikant, and A. L. Stolyar, "Pathwise optimality of the exponential scheduling rule for wireless channels," *Advances in Applied Probability*, vol. 36, no. 4, pp. 1021–1045, 2004.

[34] S. L. Bell and R. J. Williams, "Dynamic scheduling of a parallel server system in heavy traffic with complete resource pooling: asymptotic optimality of a threshold policy," *Electronic Journal of Probability*, vol. 10, no. 33, pp. 1044–1115, 2005.

[35] S. P. Meyn, *Control Techniques for Complex Networks.* Cambridge, England: Cambridge University Press, 2007.

[36] I. M. Verloop, S. C. Borst, and R. Nunez-Queija, "Delay optimization in bandwidth-sharing networks," in *Proceedings of the Conference on Information Sciences and Systems (CISS)*, Princeton University, 2006, pp. 1260–1265.

[37] T. Bonald and L. Massoulie, "Impact of fairness on Internet performance," in *Proceedings of ACM Sigmetrics*, 2001, pp. 82–91.

[38] D. Bertsekas, *Dynamic Programming: Deterministic and Stochastic Models.* Englewood Cliffs, NJ: Prentice Hall, 1987.

[39] A. L. Stolyar, private communication, May 2008.

[40] S. G. Mohanty and W. Panny, "A discrete-time analogue of the M/M/1 queue and the transient solution: A geometric approach," *Sankhya: The Indian Journal of Statistics*, vol. 52, pp. 364–370, 1990.

[41] L. Bui, R. Srikant, and A. L. Stolyar, "Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing," in *Proceedings of the 28th IEEE INFOCOM*, April 2009, pp. 2936–2940.

[42] L. Bui, R. Srikant, and A. L. Stolyar, "Optimal resource allocation for multicast sessions in multi-hop wireless networks," *Philosophical Transactions of the Royal Society, Ser. A*, vol. 366, no. 1872, pp. 2059–2074, 2008.

[43] L. Ying, S. Shakkottai, and A. Reddy, "On combining shortest-path and back-pressure routing over multihop wireless networks," in *Proceedings of the 28th IEEE INFOCOM 2009*, April 2009, pp. 1674–1682.

[44] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the air: Practical wireless network coding," in *ACM SIGCOMM Computer Communication Review*, vol. 36, 2006, pp. 243–254.

[45] M. Effros, T. Ho, and S. Kim, "A tiling approach to network code design for wireless networks," in *Information Theory Workshop*, 2006, pp. 62–66.

[46] H. Seferoglu, A. Markopoulou, and U. Kozat, "Network coding-aware rate control and scheduling in wireless networks," in *Special Session on "Network Coding for Multimedia Streaming," ICME*, Cancun, Mexico, June 2009, pp. 1496–1499.

[47] S. Sengupta, S. Rayanchu, and S. Banerjee, "An analysis of wireless network coding for unicast sessions: The case for coding-aware routing," in *Proceedings of the 26th IEEE INFOCOM*, Anchorage, Alaska, May 2007, pp. 1028–1036.

[48] T. Ho and H. Viswanathan, "Dynamic algorithms for multicast with intra-session network coding," *IEEE Transactions on Information Theory*, vol. 55, no. 2, pp. 797–815, Feb. 2009.

[49] A. Eryilmaz and D. S. Lun, "Control for inter-session network coding," in *Proceedings of the Workshop on Network Coding, Theory and Applications (NetCod)*, Jan. 2007.

[50] L. Chen, T. Ho, S. H. Low, M. Chiang, and J. C. Doyle, "Optimization based rate control for multicast with network coding," in *Proceedings of the 26th IEEE INFOCOM*, Anchorage, Alaska, May 2007, pp. 1163–1171.

[51] B. Awerbuch and T. Leighton, "A simple local-control approximation algorithm for multicommodity flow," in *Proceedings of the 34th Annual Symposium on the Foundations of Computer Science*, 1993, pp. 459–468.

[52] C. Joo, X. Lin, and N. B. Shroff, "Understanding the capacity region of the greedy maximal scheduling algorithm in multi-hop wireless networks," in *Proceeding of the 27th IEEE INFOCOM*, 2008, pp. 1103–1111.

[53] A. Brzezinski, G. Zussman, and E. Modiano, "Enabling distributed throughput maximization in wireless mesh networks - a partitioning approach," in *Proceedings of ACM MobiCom*, Sep. 2006, pp. 26–37.

[54] M. Leconte, J. Ni, and R. Srikant, "Improved bounds on the throughput efficient of greedy maximal scheduling in wireless networks," in *Proceedings of ACM MobiHoc*, 2009, pp. 165–174.

[55] B. Li, C. Boyaci, and Y. Xia, "A refined performance characterization of longest-queue-first policy in wireless networks," in *Proceedings of ACM MobiHoc*, 2009, pp. 65–74.

[56] X. Lin and N. Shroff, "On the stability region of congestion control," in *Proceedings of the 42nd Allerton Conference on Communications, Control and Computing*, Monticello, IL, 2004.

[57] M. J. Neely, E. Modiano, and C. Li, "Fairness and optimal stochastic control for heterogeneous networks," in *Proceedings of the 24th IEEE INFOCOM*, 2005, pp. 1723–1734.

[58] A. L. Stolyar, "Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm," *Queueing Systems: Theory and Applications*, vol. 50, no. 4, pp. 401–457, Aug. 2005.

[59] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," *IEEE/ACM Transactions on Networking*, vol. 15, no. 6, pp. 1333–1344, 2007.

[60] A. Eryilmaz and R. Srikant, "Joint congestion control, routing and MAC for stability and fairness in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1514–1524, 2006.

[61] X. Lin, N. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1452–1463, 2006.

[62] M. Bramson, "Convergence to equilbria for fluid models of FIFO queueing networks," *Queueing Systems: Theory and Applications*, vol. 22, pp. 5–45, 1996.

[63] Sprint-Nextel Corp., (2009, Dec.). Sprint IP network performance. [Online]. Available: https://www.sprint.net/performance/.

[64] X. Lin, N. Shroff, and R. Srikant, "On the connection-level stability of congestion-controlled communication networks," *IEEE Transactions on Information Theory*, vol. 54, no. 5, pp. 2317–2338, 2008.

[65] T. Bonald and M. Feuillet, "On the stability of flow-aware CSMA," *Performance Evaluation*, vol. 67, no. 11, pp. 1219–1229, 2010.

[66] M. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: Evidence and possible causes," *IEEE/ACM Transaction on Networking*, vol. 5, no. 6, pp. 835–846, 1997.

[67] S. Asmussen, *Applied Probability and Queues.* New York, NY: Springer-Verlag, 2003.

[68] J. Ghaderi, T. Ji, and R. Srikant, "Connection-level scheduling in wireless networks using only MAC-layer information," in *Proceedings of IEEE INFOCOM*, 2012, to be published.

[69] T. Ji, E. Athanasopoulou, and R. Srikant, "On optimal scheduling algorithms for small generalized switches," *IEEE/ACM Transaction on Networking*, vol. 18, no. 5, pp. 1585–1598, Oct. 2010.

[70] E. Athanasopoulou, L. Bui, T. Ji, R. Srikant, and A. Stoylar, "Backpressure-based packet-by-packet adaptive routing in communication networks," *IEEE/ACM Transaction on Networking*, submitted for publication.

[71] T. Ji and R. Srikant, "Scheduling in wireless networks with connection arrivals and departures," in *Proceedings of UCSD ITA Workshop*, Feb. 2011.