# Resource Management and Pricing in Networks

by

Sharad Birmiwal

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2012

## Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners. Parts of this thesis will appear in [1, 2, 3].

I understand that my thesis may be made electronically available to the public.

**Abstract**

Resource management is important for network design and deployment. Resource management and allocation have been studied under a wide variety of scenarios — routing in wired networks, scheduling in cellular networks, multiplexing, switching, and channel access in opportunistic networks are but a few examples. In this dissertation, we revisit resource management in the context of routing and scheduling in multihop wireless networks and pricing in single resource systems.

The first issue addressed is of delays in multihop wireless networks. The resource under contention is capacity which is allocated by a joint routing and scheduling algorithm. Delay in wireless networks is a key issue gaining interest with the growth of interactive applications and proliferation of wireless networks. We start with an investigation of the back-pressure algorithm (BPA), an algorithm that activates the schedule with the largest sum of link weights in a timeslot. Though the BPA is throughput-optimal, it has poor end-to-end delays. Our investigation identifies poor routing decisions at low loads as one cause for it. We improve the delay performance of max-weight algorithms by proposing a general framework for routing and scheduling algorithms that allow directing packets towards the sink node dynamically. For a stationary environment, we explicitly formulate delay minimization as a static problem while maintaining stability. We see similar improved delay performance with the advantage of reduced per time-slot complexity.

Next, the issue of pricing for flow based models is studied. The increasing popularity of cloud computing and the ease of commerce over the Internet is making pricing a key issue requiring greater attention. Although pricing has been extensively studied in the context of maximizing revenue and fairness, we take a different perspective and investigate pricing with predictability. Prior work has studied resource allocations that link insensitivity and predictability. In this dissertation, we present a detailed analysis of pricing

under insensitive allocations. We study three common pricing models — fixed rate pricing, Vickrey-Clarke-Groves (VCG) auctions, and congestion-based pricing, and provide the expected operator revenue and user payments under them. A pre-payment scheme is also proposed where users pay on arrival a fee for their estimated service costs. Such a mechanism is shown to have lower variability in payments under fixed rate pricing and VCG auctions while generating the same long-term revenue as in a post-payment scheme, where users pay the exact charge accrued during their sojourn. Our formulation and techniques further the understanding of pricing mechanisms and decision-making for the operator.

**Acknowledgements**

I am deeply indebted to Prof. Ravi Mazumdar for his invaluable guidance. His positive attitude and encouragement throughout the four years at Waterloo helped me immensely to achieve my goal. I will always remember the experiences he shared with me to help me develop academically and professionally. I am equally grateful to Dr. Shreyas Sundaram for his meticulous attention to detail which gave me greater confidence in my conclusions. This dissertation would not be complete without the support of my two supervisors.

I would like to thank Prof. D. Manjunath (Indian Institute of Technology Bombay) and Dr. Sayee Kompalli (Centre of Excellence in Wireless Technology) for their invaluable inputs during discussions on Weighted Back-pressure Algorithms. My special thanks to Prof. Manjunath and Dr. Jayakrishnan Nair (California Institute of Technology) for their contributions to the static formulation for delay reduction. I benefited greatly from their passion for research.

I am grateful to Prof. Peter Marbach (University of Toronto), Prof. Raouf Boutaba, Dr. Patrick Mitran, and Prof. Weihua Zhuang for serving on my PhD committee and being accommodating. Their suggestions helped provide a broader context to my research and cover overlooked issues.

It is my pleasure to thank all my colleagues and friends who shared their wisdom and experiences. Their company has made my time at Waterloo unforgettable.

I owe all my achievements to my family. Their unswerving support has helped me overcome difficult times. I have drawn strength and inspiration from them and am fortunate to find my role models so close to me in life.

Waterloo, 2012
Sharad Birmiwal

*Lovingly dedicated to my mother, my symbol of strength.*

# Table of Contents

# List of Figures

# Chapter 1

# Introduction

A network is an interconnection of resources and resource seekers. To maximize the utilization of a network, it is essential to manage resources efficiently. For a resource operator, it is indispensable to predict earnings from the allocation of resources. This dissertation addresses these two issues.

Capacity is the constraining resource in wireless networks. We focus our attention on managing capacity efficiently, viz. delay in multihop wireless networks. To get some perspective, analyzing delay in a network is typically more difficult than analyzing throughput. Throughput entails characterizing the rate at which packets leave a network; delay requires calculating the sojourn times of packets leaving the network. To accurately model end-to-end delay in a multihop network, the state space of the underlying Markov chain must be expanded to include arrival times or the 'age' process. Another challenge in modelling is the correlations between different queues in the network.

We also focus on pricing of a resource sold by an operator or an agent. The resource can be bandwidth in communication systems or CPU time in a cloud computing data centre. The analysis exploits the benefit of predictability ensued by insensitive allocations. Examples of insensitive allocations for a

single server system are the processor sharing discipline and the Last-In-First-Out (LIFO) discipline. We analyze the processor sharing discipline.

## 1.1 Motivation

The demand for video streaming over the Internet has seen an explosive growth. Other delay sensitive applications, e.g., IP telephony or exchange of safety messages in Vehicular Ad hoc Networks (VANETs), are also gaining popularity. However, research on networks has largely focused on optimization of throughput. Foreseeing the future demands for time-sensitive applications, the widespread adoption of wireless networks, and the growth of heterogeneous devices necessitate addressing delay in the design of network algorithms more formally.

The inspiration for studying pricing originates from allowing a resource operator to project earnings and allow making strategic decisions such as installing or reducing capacity. A motivating example for studying pricing and resource allocation is the gaining popularity of cloud computing applications, e.g., the Amazon Elastic Cloud Computer (EC2) which offers computing power as a resource or the Amazon Simple Storage Service (S3) offering storage. Another example is of an Internet Service Provider (ISP) that allocates bandwidth to users. We note that in each case, the user is charged for its resource usage. Considering state-dependent pricing also accurately reflects congestion costs. Thus, a study of usage-based pricing mechanisms with insensitive allocations which offer predictability is required.

## 1.2 Contributions

We present two approaches to improve delay performance of joint routing and scheduling algorithms in multihop wireless networks. Our main contributions

are

- the identification of causes for the poor delay performance of the back-pressure algorithm,
- the design of a dynamic throughput-optimal framework for max-weight scheduling like algorithms where flows and links can be prioritized by assimilating various inputs,
- the analysis of two algorithms that solve a static delay minimization problem and propose two implementations for optimal routes and schedules.

Both the dynamic and the static approaches demonstrate significantly improved delay performance over the back-pressure algorithm.

The second part of the thesis is related to the issue of capacity and pricing. We investigate the relation between pricing as a metric and user performance. This gives rise to a new model and our main contributions are

- studying three pricing models, viz., fixed rate pricing, Vickrey-Clarke-Groves auctions, and congestion based pricing under processor sharing,
- proposing a unique Quality of Service constraint to encourage fairness,
- characterizing mean user payments and mean operator revenue and obtaining insights into the structure of the pricing models,
- designing a pre-payment scheme and evaluating the confidence in means by deriving the second moment of user payments.

## 1.3 Outline

This dissertation is divided into two parts. The first part focuses on studying delay in multihop wireless networks. We start in Chapter 2 with an

investigation of the back-pressure algorithm of [4] — a throughput optimal algorithm known to have poor delay performance. Chapter 2 also develops a class of generalizations of the back-pressure algorithm, we call weighted back-pressure algorithms, that allow prioritization of certain flows and links in the network. This algorithm is analyzed. An example of the remaining hop weighted back-pressure algorithm is constructed and evaluated. Chapter 2 also proposes an adaptive behaviour to provably retain throughput optimality. We refer to this algorithm as a hybrid weighted back-pressure algorithm. A hybrid variant of the remaining hop weighted back-pressure algorithm is presented as an example and evaluated.

In Chapter 3, mean delay minimization is addressed directly under stationary settings. An optimization problem is formulated for obtaining long-term stochastic averages for routing and scheduling. The chapter is dedicated to analyzing this formulation, implementing the optimal solutions obtained, and evaluating its performance.

The second part of this dissertation addresses pricing of resources. Chapter 4 focuses on the processor sharing discipline and explicitly characterizes the mean of user payments and the mean operator revenue under three pricing models. Two design mechanisms are considered: first, where a user pays after completing its service and second, where the user pays the operator's estimated price on arrival. The chapter studies such pricing structures in some detail.

Chapter 5 summarizes the dissertation and presents some future directions and extensions to our work.

# Part I

# Improving Mean Delays in Multihop Wireless Networks

# Chapter 2

# The Weighted Back-pressure Algorithms

## 2.1 Introduction

Designing efficient algorithms for wireless networks has been studied for over three decades. In most deployments (e.g., WLAN, cellular networks, bluetooth), the channel is a shared resource. A sender transmits a message by broadcasting it on a particular choice of frequency, modulation scheme, and time instant (the channel). The receiver, also tuned to the same channel, decodes the message correctly if the received signal is sufficiently noise-free. In the air medium and a broadcast transmission, several sources of noise exist. The signal attenuates rapidly compared to, say, copper used in wired networks. The signal amplifying unit in the receiver to mitigate the loss (gain control) can introduce noise. Another source of noise is the superimposition of multiple copies of the transmitted signal by reflections from surrounding objects. Yet another source of noise is other transmitters, transmitting on the same frequency, in the vicinity of the receiver. The degradation of the signal sent by the intended transmitter, caused by other transmitters is called

interference.

In particular, since the signal attenuation over space is significant, the received signal is dominated by transmitters closer to the receiver. An apt analogy is when two people, call them Alice and Bob, who are equally loud, converse simultaneously with their respective partners. For Charlie situated closer to Alice, Alice's voice will drown out Bob's voice. Furthermore, Bob's voice becomes increasingly inaudible as Bob moves away. This physical layer effect where the nearest (or the strongest heard) transmitter dominates the received signal is called the *capture effect*.

A simplifying assumption often made is that when two nearby transmitters transmit, both transmitted messages are lost due to interference. It is important to note here that collision of messages occur at the receiving station. Such an interference model is called the *protocol interference model*. In this model, concentric circles with radius $r_{tx}$ and $r_i$ respectively define the transmission and the interference region. A receiver can decode a message successfully only if the transmitter is within $r_{tx}$ distance and if no other sender transmits within the interference region.

The *physical interference model*, also known as the *Signal-to-Interference Noise Ratio (SINR) model*, models the wireless channel more accurately. Here, the ratio of the strength of the received signal to the strength of noise and interferences is evaluated. A successful decoding of the message occurs if this ratio lies above a certain threshold. This interference model, though more accurate, suffers from increased complexity.

Several standard mechanisms exist to avoid interference. Frequency Division Multiple Access (FDMA) is an access mechanism where the available channel spectrum is shared, simply by splitting it and allocating chunks to users. FDMA is popular in satellite communication. Time Division Multiple Access (TDMA) is an access mechanism where users are fully allocated the entire resource for dedicated time periods. TDMA is widely used in the GSM

standard. The sharing is in the temporal domain. Another access mechanism is Code Division Multiple Access (CDMA) where multiple senders can transmit simultaneously on the same channel but using different, non-interfering code sequences between transmitter-receiver pairs. CDMA is used as an access mechanism in the 4G mobile telecommunication standard. The actual transmissions are encoded messages using these code sequences.

The scheduling component in network design is to identify which transmitters are allowed to transmit simultaneously. Such a list of transmitters is called a schedule. A self-evident solution to avoid interference is to allow a single transmission at a time in the network. Although such a scheme will work, it underutilizes the available capacity. Consider two pairs of communicating nodes (transmitters or receivers). If they are situated sufficiently far apart, they can transmit simultaneously without interfering with each other. Such a reuse of channel is called *spatial reuse*. Spatial reuse is important to consider for maximizing the utilization of capacity.

Most present day implementations of wireless networks rely on a central entity to facilitate coordination and communication. Commonly seen examples of such controllers include access points in IEEE 802.11 (WiFi) and IEEE 802.16 (WirelessMAN or WiMax) networks and base stations in cellular networks. The advantage of an architecture with controllers is simplified implementation. Such coordinators are feasible when there is a single owner of the network. A network of networks with multiple ownerships or one lacking central infrastructure will require the nodes to coordinate themselves. It is envisioned that such distributed implementations will allow performance to scale with the network size easily. Note that the IEEE 802.11 standard already defines a Distributed Coordination Function (DCF) mode of operation in the absence of access points.

In a system with slotted time, another consideration is the amount of information required to identify an appropriate schedule. For example, an

opportunistic scheduling algorithm would consider current backlogs (queue lengths) at every node to identify the optimal schedule. We call such an algorithm a *dynamic scheduling algorithm*. In contrast, algorithms where the activated schedule depends on long-term time averages are called *static scheduling algorithms*.

Another challenge in store-and-forward networks is to identify how to relay messages between two far-apart communicating nodes. The sequence of nodes a message traverses is called the *route* or the *path* taken by the message. Of course, if the network is small with every node within the transmission range of every other node, a transmitting node can directly transmit the message to the receiving node. In larger networks, this problem is non-trivial. In wireline networks, Dijkstra's algorithm and the Bellman-Ford algorithm form the crux of the standard routing protocols (such as link-state routing protocols and distance vector routing protocols) that generate routing decisions from connectivity information. Extending these algorithms to wireless networks is non-trivial. The challenges arise due to mobility of nodes, frequent change in *link costs* (reliability of links) which are utilized in determining optimal paths, the unreliability of communication between peers, and overheads involved in communication. In truly distributed networks, lack of hierarchical infrastructure (e.g., addressing) exacerbates the problem even further. In a cellular network which has a hierarchical design, each mobile node is managed by a nearby base station (the node is *associated* with the base station). When a node wishes to communicate a message to another node, it passes the message to its base station, which using a back-haul network, relays the message to the base station of the receiver, which passes the message to the intended receiver. The routing decision here is reduced to simply passing the message to the associated base station which uses pre-computed routing decisions to deliver the message to the destination. In systems lacking such central coordinators, intermediate nodes relay the message to the destina-

tion. An example of a routing protocol for such environments is the Ad hoc On-demand Distance Vector (AODV) routing protocol where the route is distributedly generated at the time of initiation of communication between two nodes.

When two nodes communicate, we assume that they require exchanging a sequence of messages (instead of a single message) that are generated (arrive into the network) by some governing law. A stream of messages from one sender node to one destination node pair thus constitutes a *flow*. A flow itself may follow a fixed, single path (single path routing) or may split over several paths (multipath routing) to deliver packets to the destination node.

While designing protocols for networks, it is insufficient to simply schedule nodes or links and route messages (or packets); the aim is to do so efficiently. There are numerous performance metrics that may be optimized. A joint scheduling and routing algorithm is called *throughput optimal* if it can stabilize the queue backlogs in a network for any traffic profile that is *stabilizable*. Stabilizability implies that there exists a joint scheduling and routing algorithm that keeps the backlogs at each queue finite. A throughput optimal algorithm has the largest capacity region (supported traffic profiles).

Delay is another performance metric of interest. The end-to-end delay or the sojourn time of a packet is the total time spent by a packet in the network, i.e., the departure time less the arrival time. Jitter is yet another metric which measures the variance in the delivery times of packets. Recently, especially with the increasing demand of live video streaming traffic on the Internet, delay and jitter have garnered increasingly greater attention.

Here, we first study the back-pressure algorithm (BPA) also known as the max-weight scheduling algorithm, a dynamic joint scheduling and routing algorithm for multihop wireless networks. This algorithm, though known to be throughput optimal, has poor end-to-end delay characteristics. We will identify several reasons for such poor delay performance. We will demonstrate

that by introducing multiplicative weights in the back-pressure algorithm, the mean end-to-end delay characteristics can be improved significantly. We call a max-weight algorithm with multiplicative weights as the weighted back-pressure algorithm (WBPA). We propose an adaptive behaviour to retain throughput optimality and present results obtained via simulations.

### 2.1.1 Background and Related Works

The seminal work of [4] presented the back-pressure algorithm, a throughput optimal joint scheduling and routing algorithm for time-slotted, multihop wireless networks. The back-pressure algorithm proposed in [4] assigned weights to links based on the difference in queue lengths across the link (and thus the name back-pressure) and chose the schedule maximizing the sum of link weights. Throughput optimality is derived via Foster's criteria by a suitable choice of a Lyapunov function to prove the stability of the underlying Markov chain. Significant advances have since been made in the understanding of multihop wireless networks with generalizations to include ergodic channels in [6, 7], to flow control for utility maximization in [8], and to input-queued switch models in [9]. Another strand of research has investigated algorithms that had lower implementation complexity than the back-pressure algorithm. Many variants of the maximum-weight matching schedule have been developed, e.g., randomized scheduling in [10], maximal scheduling in [11], and other suboptimal (but easily implementable in a distributed manner) schedules in [12]. Most suboptimal schemes reduce the schedulable region, sometimes by significant (usually a constant) fraction; see [13] for a formal treatment.

A body of work exists on the Greedy Maximal Scheduling algorithm (GMS) also known as the Longest Queue First (LQF) scheduling scheme in the input-buffered packet switch scheduling literature. To determine the

schedule to activate, this iterative algorithm is run by first selecting the link with the longest backlog and eliminating (disabling) all links interfering with this link. In the next step, the link with the longest backlog in the remaining links is identified and selected and its interfering links are disabled. This step is repeated until all links have been either selected or disabled for the current timeslot. The final set of selected links forms the schedule and is activated in the timeslot. The algorithm is especially popular because of ease of finding distributed approximations and implementations. The LQF scheduling algorithm does not solve the routing problem and is suitable only for one-hop traffic (destination is within the transmission range of the source node) in the current form. It is believed to have a reduced capacity region for general topologies. For topologies where the *local pooling* condition is satisfied, the LQF algorithm is known to be throughput optimal (see [12]). Let $\mathcal{L}$ be the set of links in a wireless network and for $L \subseteq \mathcal{L}$, let $M[L]$ be the set of maximal schedules on $L$. A schedule $s$ is a $0-1$ vector with $s_l = 1$ if link $l$ is activated. A schedule $s$ is maximal if no additional links can be activated in $s$ without causing collisions. A wireless network topology satisfies the local pooling property if there exists a constant $c > 0$ such that for every for all $L \subseteq \mathcal{L}$ and $\phi \in \text{co}(M[L])$, there exists a vector $\alpha$ such that $\alpha^\intercal \phi = c$ holds. Here, $\text{co}(M[L])$ denotes the convex hull of $M[L]$. Vector $\phi$ indicates the capacity or the long term service rate available at each link.

In [14, 15], the idea of local pooling is further generalized to include networks that do not possess the local pooling property. A generalized *local pooling factor*, $\sigma \in [0, 1]$, is defined for any wireless network topology. It is shown in [14] that the minimum capacity region achieved by a scheduling algorithm satisfying the $\sigma$-local pooling property is a $\sigma$ fraction of the total capacity region.

The primary emphasis in much of the early work on multihop wireless networks has been on analyzing the schedulable region; analysis of the delay

performance was not seriously attempted but for some exceptions, e.g., [16]. There is some recent literature in analyzing and understanding the delay performance in the back-pressure based scheduling algorithms, e.g., [17, 18]. Modifications of the max-weight algorithm have also been suggested to improve mean link delays, e.g., [19, 20]. In [21], a delay bound is obtained by identifying bottlenecks in a network topology with fixed routing and max-weight scheduling. In [22], the cause for large delays in the back-pressure algorithm are identified to be the routing of packets over long routes and use of separate packet queues for each destination at every node. Simplifications by considering only one hop flows have also been attempted, e.g., [11, 23]. There have also been some scheduling algorithms proposed to reduce the end-to-end delays. The work in [24] describes a randomized algorithm to reduce the per-hop delay while also reducing the schedulable region. In [25], a network with primary interference constraints is considered to develop an 'emulation based' scheduling algorithm. This scheme reduces the schedulable region by a constant factor. In [26], additional constraints to improve delay are proposed.

The *modified largest weighted delay first* scheduling scheme of [27] merges the idea of delay and max-weight scheduling. The work shows that such an algorithm is throughput-optimal. In [28], the delay properties of the *exponential scheduling* rule is studied and shown to be throughput optimal and to asymptotically minimize a weighted sum of delay of each queue. Both models consider one hop traffic only.

Reduced capacity algorithms may perform better at low loads but queueing delays start dominating at lower loads than for the back-pressure algorithm because of the proximity to the reduced capacity boundary. This effect is illustrated by a simplified view of the delay-throughput curve shown in the 'concept graph' in Fig. 2.1. Delay has two significant components in a store-and-forward multihop network: hop delay is associated with the length of

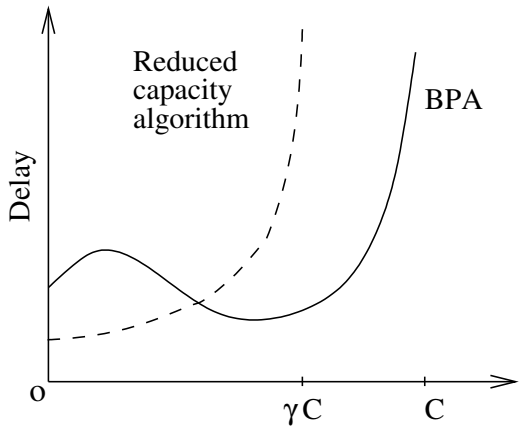



Figure 2.1: Illustration of the end-to-end delay performance of the back-pressure algorithm (solid line) and a reduced capacity routing and scheduling algorithm (dashed line).

the path and queueing delay with congestion at each hop. At low loads the back-pressure algorithm tends to send packets over long routes via a mechanism akin to a random walk (as we will see later). This increases the hop delay significantly. However, as the load increases, sufficient back-pressure develops and the packets experience a positive drift towards the destination. The average hop lengths decrease and reduce the end-to-end delay. Further increase in the load increases the queueing delay and the end-to-end delay starts to increase again. For suboptimal algorithms, although the delays at low loads can be low, the delays at moderate loads (relative to the capacity) can be significant due to an increase in the queueing delays. Thus our work avoids reducing the stability region.

Schemes that reduce the end-to-end delay without decreasing the capacity region have also been proposed. In [5], the average path length across all flows is minimized to improve hop delays. We will demonstrate that in topologies with shorter routes passing through congested nodes, average path length minimization pushes more traffic on congested links thereby increasing

queueing delays. Another proposed scheme is for the flow-control setting in [22]. A routing optimization is performed to minimize the total resource allocation. This is used to develop a scheduling scheme that is expected to reduce delay in the congestion control setting. However, the trade-offs between load balancing, path lengths, and queueing delays is not clear in [22].

In [29], a throughput optimal distributed CSMA algorithm is presented for scheduling one hop traffic. The work does not consider delay performance. The work of [30] shows that unless NP $\subset$ BPP, there exists no algorithm that has high throughput, low delay, and low computational complexity for general network topologies. In [31], a CSMA based algorithm is presented that is throughput-optimal and has order-optimal delay performance for a toroidal interference graph topology.

### 2.1.2 Outline of this Chapter

In Section 2.2, we investigate the causes for high end-to-end delay exhibited by the back-pressure algorithm. In Section 2.3, we present weighted back-pressure algorithms (WBPAs) — generalizations of the the back-pressure algorithm which are dynamic joint scheduling and routing algorithms. A lower bound on the capacity region of a WBPA is shown. Section 2.4 presents adaptive variants of weighted back-pressure algorithms, that we call hybrid weighted back-pressure algorithms, which are throughput optimal. The results are presented in Section 2.5 and the concluding remarks are presented in Section 2.6.

## 2.2 Delays in the Back-pressure Algorithm

The poor delay performance of the back-pressure algorithm is almost a part of folklore. However, as discussed in the previous section, there are few studies to analyze this. This section explores some of the reasons via simulation models. In [22], it has been argued that there are two reasons for this increased delay. First, packets are typically routed over longer routes even when shorter routes are not congested. Second, each node keeps separate queues for packets for each destination. The former phenomenon is explored in more detail in this section. Specifically, a qualitative feel is obtained for this phenomena through simulation experiments.

Consider the node exclusive interference model or the primary interference constraints where a node cannot transmit and receive simultaneously and a node can receive a message from only one transmitter at a time. All edges are half-duplex bidirectional links. The scale for end-to-end delay and hop delay in the following plots is the same where one time slot is the time required for one hop.

### 2.2.1 Overworking and Random Walks at Low Loads

The back-pressure algorithm activates a maximum matching in every slot. The packet at the head of the queue of the sending node of an activated link is transmitted on the link, irrespective of the packet's intended destination. To motivate the consequences, consider the network in Fig. 2.2 at low loads. In an exaggerated scenario, suppose a single packet at node $S$ destined for node $D$ and no other packet in the network. Each egress link from node $S$ has the same back-pressure of 1 packet. Link $SA$ and link $SB$ are equally likely to be activated and thus the packet can get *misrouted*, i.e., sent on a link that is not the shortest path. Packets may also potentially loop in the network before reaching the destination. Note that a misrouted packet has

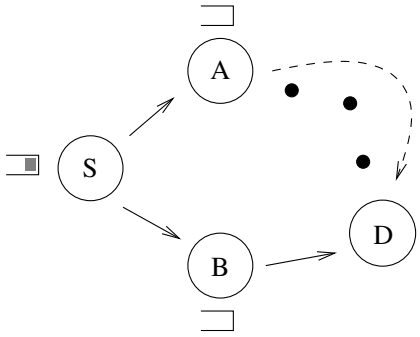
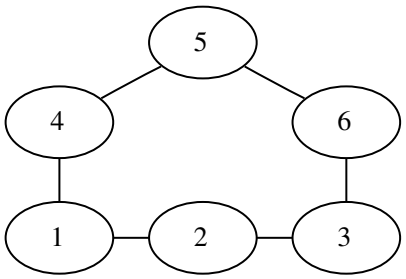


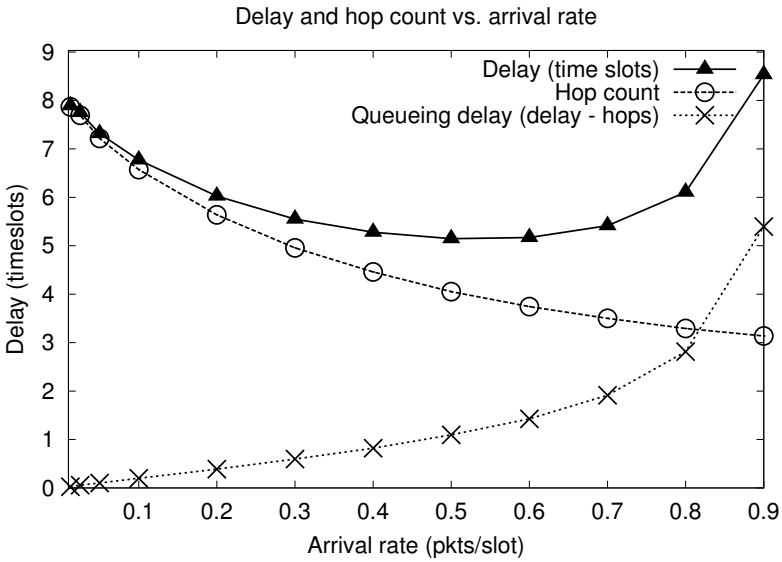

Figure 2.2: Both link $SA$ and link $SB$ have unit back-pressures and are equally likely for activation irrespective of path lengths to destination $D$.

to wait to be scheduled a greater number of times. Thus at low loads, hop delay contributes significantly to end-to-end delay.

The above reasoning is verified by the network topology considered in Fig. 2.3a. In Fig. 2.3b, the end-to-end delay, average path length (in the number of hops traversed by a packet), and queueing delay are plotted in the presence of a single flow from Node 1 to Node 3 in the topology of Fig. 2.3a. Not surprisingly, at low arrival rates, the mean hops to sink (the hop count) is greater than the length of the longest path of 4 hops. The packets traverse the network akin to a random walk. In a hypothetical network where a packet originating at Node 1 is performing a random-walk, the mean absorption time at Node 3 is 8 time slots which matches closely with our experimental result. Fig. 2.3b also shows the monotonically increasing queueing delays in the network, as expected. As the arrival rate increases, the queue backlogs start rising. In [22], an argument is presented for the mean queue length to increase linearly per-hop as the distance to the sink increases. Using the same argument, the mean backlog at Node 4 is greater than the mean backlog at Node 2 and thus new arrivals at Node 1 are directed towards Node 2 to reduce the greater back-pressure. Thus the packets are 'guided' by the back-pressure along shorter routes towards the sink. An alternate view is to interpret the




(a) Network with a single flow from Node 1 to Node 3




(b) Delay components for the flow in Fig. 2.3a

Figure 2.3: Scenario demonstrating overworking and random walks at low loads.

system as a random walk with a greater drift towards the destination on shorter paths. This setting up of backlogs and back-pressures at high arrival rates is indicated by the increase in the end-to-end delay and increase in the queueing delay but a decrease in the average hop count, as seen in the experiments.

When the experiments are repeated with the same flow and low traffic flows between every pair of nodes, similar trends are observed for average hop count and delay.

### 2.2.2 Richer Topologies Lead to Longer Delays at Low Loads

We observe that greater number of egress links at a node leads to greater opportunities for the packet to get misrouted. This is illustrated with the $2\times3$ grid network in Fig. 2.4a. A single flow is considered though the direction of the flow is reversed to obtain two scenarios, i.e., in the first scenario, a single flow exists from Node 2 to Node 1 and in the second scenario, a single flow from Node 1 to Node 2 exists. Since there are fewer opportunities to deviate in the paths for the second flow, it exhibits lower end-to-end delay as compared to the same flow with reversed source and destination nodes. The average path length or the hop count is shown in Fig. 2.4b. The results at low load agree with an analytical model of a single packet performing a random walk (as in Section 2.2.1). For the first scenario, the mean time to hit Node 1 is 6.05 hops and the mean time to hit Node 2 in the second scenario is 3.86 hops.

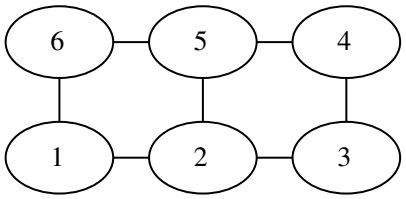
Once again, similar behaviour was observed when the experiments are repeated with low background traffic. The asymmetry in this case arises from the network topology perceived by the packets of the flows. Thus a 'rich' topology is not necessarily good for delay. This can be related to

the Braess's paradox which states that introducing additional links (more capacity) to the network may reduce the performance of the network (see [32]). We note that additional links introduce greater interference in wireless networks and do not necessarily increase capacity.

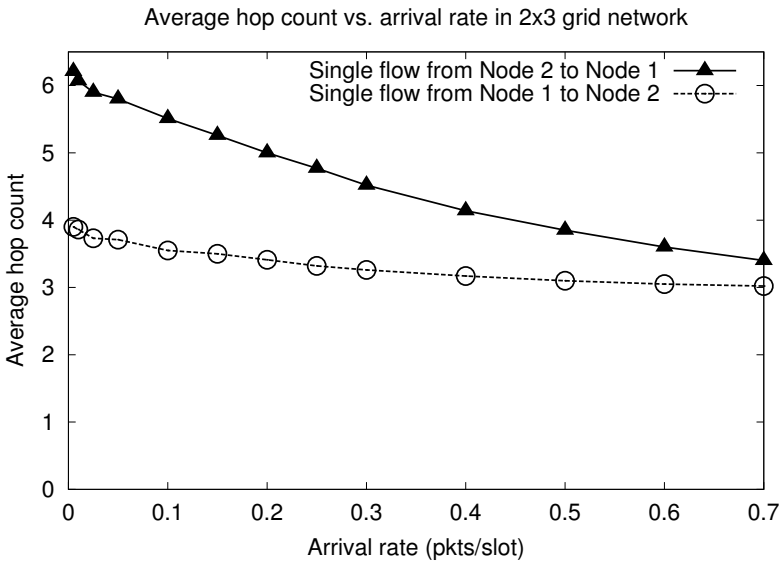
### 2.2.3 Asymmetric Loads Penalize Low Load Flows

The back-pressure algorithm degrades the delay performance of low load flows when the arrival rates of flows are asymmetric. Flows with high arrival rates have greater packets in the network and greater back-pressures, thus dictating maximal schedules. In comparison, the contribution of low arrival rate flows in the link weight sum becomes negligible. As a result, high arrival rate flows get scheduled more often. Further, because of the algorithm's overworking tendency, packets of smaller load flows may get misrouted when instead they should not be scheduled. Thus the propensity of the back-pressure algorithm to 'work harder' also increases the end-to-end delay for some flows. To illustrate this with an experiment, consider the network in Fig. 2.5a with a high load of 0.65 packets per time slot from Node 2 to Node 1 and a flow with low varying arrival rate from Node 6 to Node 1.

Following the argument above, the two schedules, $\{(2,1),(3,4)\}$ and $\{(2,3),(4,1)\}$, in the loop $1-2-3-4-1$ typically contributes significant weight in the sum of link weights. Either matching in the loop is not maximal and link $(5,6)$ is added to make the schedule maximal. A packet at Node 5 destined for Node 1 thus is sent back to Node 6. In this case, frequent activation of link $(5,6)$ leads to the packet vacillating between Node 5 and Node 6. In Fig. 2.5b, the average hop count for the flow from Node 6 to Node 1 at varying arrival rates is shown when a flow with high load in the loop is present and absent. End-to-end delay is significantly high for the flow from Node 6 to Node 1 in the case of traffic in loop.

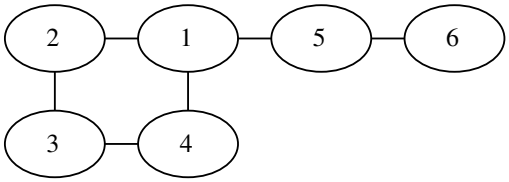



(a) Topology considered

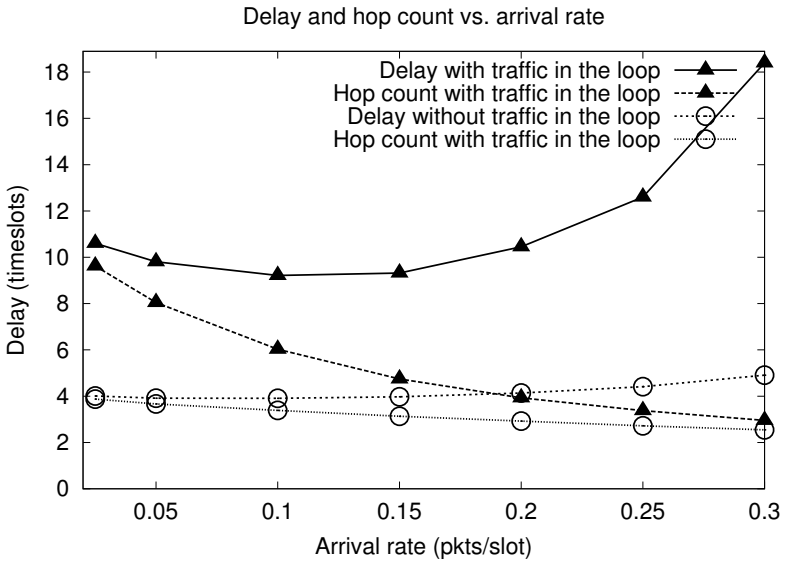



(b) Average hop count for the flows in Fig. 2.4a

Figure 2.4: Scenario demonstrating richer topologies lead to longer delays at low loads.




(a) Network to demonstrate effect of topology and asymmetric load




(b) Average hop count for flow from Node 6 to Node 1 in Fig. 2.5a

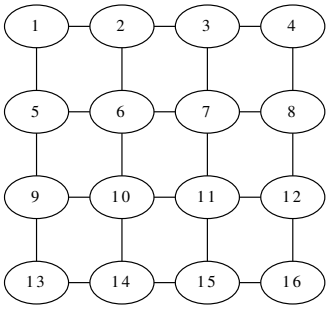
Figure 2.5: Scenario demonstrating the effect of asymmetric loads.

### 2.2.4 Large Networks

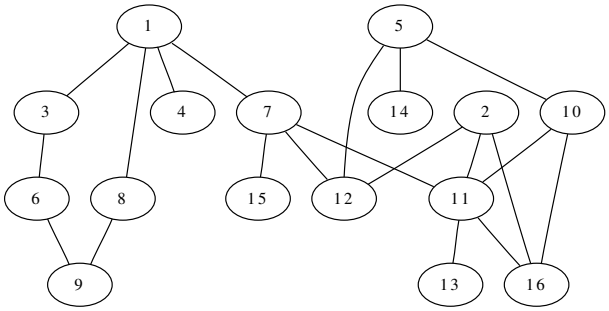
The above 'truisms' of delay behaviour are not confined to simplistic networks but are prevalent more widely. To illustrate this, experiments are performed on several topologies. We present the results obtained on the topologies in Fig. 2.6. The end-to-end delay for these networks are shown in Fig. 2.9a, Fig. 2.9b, and Fig. 2.10a. Note the high hop count at low arrival rates and the early build up of queues for the back-pressure algorithm in Fig. 2.10b.

We can reduce end-to-end delays at low and moderate loads by reducing hop count. In other words, by encouraging packets to take shorter routes based on current congestion levels in the network, the delay performance of a back-pressure like algorithm can be improved. To this effect, a simple generalization of the back-pressure algorithm is presented in the next section.

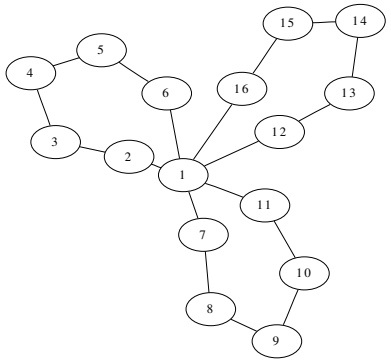
Before presenting the generalization, the proposal of [5] to mitigate delay in back-pressure like algorithms is briefly discussed. Hop delay is identified as a significant contributor to end-to-end delay in [5]. To reduce the average hop delay, [5] first presents a mechanism wherein packet delivery to the sink in a given feasible number of hops is guaranteed by the addition of hop-length dependent queues for each flow at each node. Next, an optimization problem is formulated to minimize the average number of hops taken by all flows in the network. Reducing the mean path length ensures that the hop delay at low loads is minimized. However, the trade-off with queueing delays is not considered. Beyond the assumption of stable arrival rates, [5] requires the arrival rates for each flow to be known a priori. The optimization problem is solved once when the network operation begins and needs to be solved every time a new flow is introduced, a flow departs, or the arrival rates change. In addition, the assignment of newly arriving packets in the system to hop length dependent queues is static and independent of queue backlogs on the paths. The dynamic scheme presented in the next section does not require knowledge of packet arrival rates.




(a) $4 \times 4$ grid network




(b) A sample, randomly generated Erdös-Rényi graph




(c) Network topology with flows on shorter routes passing through Node 1

Figure 2.6: Various topologies used for experiments.

## 2.3 Weighted Back-pressure Algorithms

We first introduce the required notation before describing a generalization of the back-pressure algorithm where multiplicative weights for each flow and each link (call it a flow-link pair) is introduced. This granularity allows for selectively increasing the priority of certain routes for certain flows. When designing algorithms for a network, the multiplicative weights can be based on a multitude of information sources. An example using the number of hops to the sink for the multiplicative weight is constructed. We will see that such a dynamic scheme demonstrates reduced end-to-end delays at low and moderate loads.

It is interesting to compare the above scheme to $\alpha$-weighted algorithms where the difference in the $\alpha$-th power of queue lengths is used as the link weights. The work of [33] shows that $\alpha$-weighted algorithms are stable for any $\alpha > 0$ and for single hop traffic. The end-to-end delay characteristics have not been discussed in [33]. Also, $\alpha$-weighted scheduling uses only queue length information to schedule packets and is thus myopic to other network effects.

### 2.3.1 System Model

The widely used model from [4] is repeated here in short for completeness. Let $\mathcal{N}$ and $\mathcal{L}$ be the set of nodes and unidirectional wireless links respectively. Let $N := |\mathcal{N}|$ and $L := |\mathcal{L}|$. Let $\mathrm{q}(l)$ and $\mathrm{h}(l)$ denote the transmitting and receiving nodes of link $l$. Let $\mathcal{L}^e(n)$ and $\mathcal{L}^i(n)$ denote the set of all egress and ingress links at node $n$. Let $\mathcal{S}$ be the set of collision free schedules. A schedule $c \in \mathcal{S}$ is an $L$-dimensional column vector with elements $c_l = 1$ if link $l$ is active and $c_l = 0$ if link $l$ is inactive in the schedule $c$. The set $\mathcal{S}$ reflects the constraints imposed by interference in the network. Let $S$ be the cardinality of $\mathcal{S}$. The triplet $(\mathcal{N}, \mathcal{L}, \mathcal{S})$ defines the topology of a single

channel multihop wireless network.

Let $\mathcal{F}$ be the set of flows characterized by the destination nodes denoted by $\mathrm{d}_f$ for $f \in \mathcal{F}$. Let $F$ be the cardinality of $\mathcal{F}$. Assume that time is slotted with all packets fitting into a time slot. The queue at node $n$ for flow $f$ is represented by $Q_{n,f}$ and its backlog (length) at time $t$ is denoted by $X_{n,f}(t)$. Let $X_f(t)$ be an $N$-sized column vector with elements $X_{n,f}(t)$ and $X(t)$ be an $N \times F$-sized matrix with elements $X_{n,f}(t)$. The back-pressure of flow $f$ on link $l$ is given by $P_{l,f}(t) = X_{\mathrm{q}(l),f}(t) - X_{\mathrm{h}(l),f}(t)$.

Assume that the packet arrival processes for a flow are independent and identically distributed and that all arrivals occur at the end of a time slot. Let $A_{n,f}(t)$ be the number of new packets arriving at node $n$ for flow $f$ at the end of time slot $t$. Let $\lambda_{n,f} = \mathbb{E}[A_{n,f}(t)]$ and $\lambda = [\lambda_{n,f}]$. Note that $\lambda$ is not assumed to be known.

The multiplicative weights are assigned for each flow-link pair. Let $\beta_{l,f}(t) \in \mathcal{B}$ be the weight assigned to flow $f$ on link $l$ for time slot $t$. We assume that $\mathcal{B}$ has finite cardinality and that for all $\beta \in \mathcal{B}$, $1 \leq m \leq \beta \leq M < \infty$. Let $\beta(t)$ be an $L \times F$ sized matrix with elements $\beta_{l,f}(t)$. A choice of $\beta$ is presented in Section 2.3.4.

The state of the system at time $t$ is denoted by $\xi_t \triangleq (X(t), \beta(t)) \in \Xi$ where $\Xi = \mathbb{Z}_+^{N \times F} \times \mathcal{B}^{L \times F}$. Let $[x]^+ = \max\{0, x\}$.

### 2.3.2 The Scheduling Algorithm

The back-pressure algorithm in [4] assigns weights to each link at a time slot $t$ as

$$v_l^*(t) = \max_{f \in \mathcal{F}} P_{l,f}(t) \tag{2.1}$$

and activates the schedule $\hat{c} \in \mathcal{S}$ with the maximum sum of back-pressures, i.e.,

$$\hat{c} = \arg\max_{c \in \mathcal{S}} \sum_{l \in \mathcal{L}} v_l^*(t)^\intercal c_l.$$

We generalize the above by introducing multiplicative weights in (2.1) as follows. Define new flow-link weights as

$$w_{l,f}(t) = [\beta_{l,f}(t) \times P_{l,f}(t)]^+. \tag{2.2}$$

Define the link weights as $v_l(t) \triangleq \max_{f \in \mathcal{F}} w_{l,f}(t)$. The optimal schedule $\hat{c}$, given by

$$\hat{c} = \arg\max_{c \in \mathcal{S}} v(t)^\intercal c,$$

maximizes the sum of weighted back-pressures. Schedule $\hat{c}$ is executed for a time slot $t$ by activating a flow on each activated link with the maximum link weight. Ties are broken arbitrarily.

### 2.3.3 Analysis

The throughput performance of any given weighted back-pressure algorithm, with respect to that of the back-pressure algorithm is presented here. A lower bound on the capacity region of the former is shown.

**Proposition 1.** *If the multiplicative weights $\beta_{l,f}$ are bounded by $m$ and $M$, i.e., $1 \leq m \leq \beta_{l,f} \leq M$, then for all arrival rates $\lambda \in \frac{m}{M}\mathcal{C}$, the weighted back-pressure algorithm is stable. Here $\mathcal{C}$ is the capacity region of the original back-pressure algorithm.*

*Proof of Proposition 1.* Take the Lyapunov function as

$$V(\xi_t) = \sum_{n \in \mathcal{N}} \sum_{f \in \mathcal{F}} X_{n,f}^2(t).$$

The one-step drift is written as $\mathcal{D}_t = V(\xi_{t+1}) - V(\xi_t) = \sum_{f \in \mathcal{F}} (X_f(t+1) - X_f(t))^\intercal (X_f(t+1) + X_f(t))$.

Since $X_f(t+1) = X_f(t) + R^f E_f(t) + A_f(t)$, where $R^f$ is the routing matrix (as in [4]) for flow $f$ and $E_f(t)$ is an $L \times 1$ indicator vector with ones

corresponding to active links of flow $f$,

$$\begin{aligned}\mathcal{D}_t = &\sum_{f\in\mathcal{F}}[R^f E_f(t) + A_f(t)]^\intercal[R^f E_f(t) + A_f(t)] \\ &+ \sum_{f\in\mathcal{F}} 2X_f(t)^\intercal[R^f E_f(t) + A_f(t)] \end{aligned} \tag{2.3}$$

The expectation of the first term in (2.3), conditioned on $\xi_t$, is bounded as

$$\begin{aligned}&\sum_{f\in\mathcal{F}}\mathbb{E}\left[[R^f E_f(t) + A_f(t)]^\intercal[R^f E_f(t) + A_f(t)]|\xi_t\right] \\ &\quad\le \sum_{f\in\mathcal{F}}\sum_{n\in\mathcal{N}}\mathbb{E}[A_{n,f}^2(t)] + 2L\sum_{f\in\mathcal{F}}\sum_{n\in\mathcal{N}}\lambda_{n,f} + NFL^2 \\ &\quad= b_1. \end{aligned} \tag{2.4}$$

Manipulating the expectation of the second term of (2.3) gives

$$\begin{aligned}&\sum_{f\in\mathcal{F}}\mathbb{E}\left[2X_f(t)^\intercal[R^f E_f(t) + A_f(t)]|\xi_t\right] \\ &\quad= 2\sum_{f\in\mathcal{F}} X_f(t)^\intercal R^f g_f(\xi_t) + 2\sum_{f\in\mathcal{F}} X_f(t)^\intercal\lambda_f \\ &\quad\le -\frac{2}{M}v(t)^\intercal\sum_{f\in\mathcal{F}} g_f(\xi_t) + 2\sum_{f\in\mathcal{F}} X_f(t)^\intercal\lambda_f, \end{aligned} \tag{2.5}$$

where $g_f(\xi_t) = \mathbb{E}[E_f(t)|\xi_t]$.

If $\lambda \in \frac{m}{M}\mathcal{C}$, the second term above can be bounded as

$$2\sum_{f\in\mathcal{F}} X_f(t)^\intercal\lambda_f \le \frac{2}{m}v(t)^\intercal\hat{r} = \frac{2}{m}v(t)^\intercal\sum_{c\in\mathcal{S}}\alpha_c c, \tag{2.6}$$

where $\hat{r}$ is an $L\times 1$ vector with $\hat{r}_l$ the total flow on link $l$, $\alpha_c \ge 0$, and $\sum_{c\in\mathcal{S}}\alpha_c \le 1$.

When more packets in a queue are scheduled for transmission than the backlog, the queue only transmits the backlog length. Thus it appears as the

following bound.

$$\max_{c\in\mathcal{S}}\{v(t)^\intercal c\} \geq v(t)^\intercal \sum_{f\in\mathcal{F}} g_f(\xi_t) \geq \max_{c\in\mathcal{S}}\{v(t)^\intercal c\} - L^2 \tag{2.7}$$

The expectation of the second term of the drift in (2.3) is bounded as

$$\begin{aligned}&\sum_{f\in\mathcal{F}} \mathbb{E}[2X_f(t)^\intercal[R^f E_f(t) + A_f(t)]|\xi_t] \\ &\qquad = 2\frac{L^2}{M} - \frac{2}{M}\max_{c\in\mathcal{S}}\{v(t)^\intercal c\}\left[1 - \frac{M}{m}\sum_{c\in\mathcal{S}}\alpha_c\right]\end{aligned} \tag{2.8}$$

Now, by bounding $\max_{n,f}\{X_{n,f}(t)\} \geq \sqrt{\frac{b}{NF}}$ and using $v_l(t) \geq P_{l,f}(t)$, we show that if $V(\xi_t) \geq b$,

$$\begin{aligned}&\mathbb{E}[V(\xi_{t+1}) - V(\xi_t)|\xi_t] \\ &\quad \leq -\frac{2}{M}\left(1 - \frac{M}{m}\sum_{c\in\mathcal{S}}\alpha_c\right)\frac{1}{N}\sqrt{\frac{b}{NF}} + 2\frac{L^2}{M} + b_1\end{aligned} \tag{2.9}$$

To complete the proof, take

$$b = NF\left(\frac{MN(\epsilon + b_1 + 2L^2/M)}{2(1 - \frac{M}{m}\sum_{c\in\mathcal{S}}\alpha_c)}\right)^2 \tag{2.10}$$

which gives the required $\mathbb{E}[V(\xi_{t+1}) - V(\xi_t)|\xi_t] \leq -\epsilon$ for $V(\xi_t) \geq b$. □

A similar result was shown in [13]. Therein, an algorithm choosing sub-optimal schedules with an approximation ratio $\gamma$, i.e., schedules $c'$ such that $v(t)^\intercal c' \geq \gamma \max_{c\in\mathcal{S}}\{v(t)^\intercal c\}$ where $0 \leq \gamma \leq 1$, were shown to attain at least $\gamma$ fraction of the capacity region. The approach used a utility maximization formulation and considered approximations occurring from possibly distributed implementations. In our approach, the sub-optimality (in their sense) appears by finding the exact solution to our posed weighted maximization problem to allow for flow-link prioritization.

### 2.3.4 Remaining-hops Weighted Back-pressure Algorithm

We present a complete joint scheduling and routing algorithm inspired by the shortest remaining processing time (SRPT) scheduling scheme, which has the smallest mean sojourn time. The multiplicative weights are chosen to increase inversely with the shortest path length to sink. By assigning higher multiplicative weights to links guiding packets closer to the sink, the packets are guided towards shorter routes. The hops to sink can be obtained by a distributed implementation of the Bellman-Ford algorithm or Dijkstra's algorithm.

Define the multiplicative weights as

$$\beta_{l,f}(t) = (N - H(\mathrm{h}(l), f))^{\alpha} \tag{2.11}$$

where $H(n, f)$ is the minimum number of hops from node $n$ to $\mathrm{d}_f$ and $\alpha > 0$. It is important to note that this choice of function assigns positive flow-link weights ($w_{l,f}$) to all egress links with positive back-pressure — making all such links probable for scheduling. If the multiplicative weights ($\beta$) assigned to longer links were negative, the algorithm would disallow packets on these links, making longer routes infeasible thereby reducing the capacity of the network.

For larger values of $\alpha$, the algorithm provides higher preference for flows with fewer hops. Simulation results of the remaining-hops weighted back-pressure algorithm are presented in Section 2.5. A discussion on improving the capacity region by a simple adaptive scheme is presented next.

## 2.4 Retaining Throughput Optimality

Proposition 1 presents a lower bound on the capacity region of a weighted back-pressure algorithm. To counter a decrease in the throughput perfor-

mance, an adaptive algorithm we call the hybrid weighted back-pressure algorithm is proposed next. It is motivated by the following argument. From the modelling assumptions, $\xi_t$ is a discrete time Markov chain over a countably infinite state space. The stability of this Markov chain is determined by its transitions in the 'higher' states (when the network is congested with a large number of packets) of the state space and is not affected by transition probabilities in the 'lower' states. Thus, if the scheduling algorithm makes the Markov chain stable in higher states, the transition probabilities can be changed in the lower states without affecting stability. Specifically, the transition probabilities can be changed to reduce the delay. Note that the delay behaviour is significantly affected by the behaviour in the 'lower' states because the Markov Chain spends most of its time in these states at low loads.

The above informal discussion is made rigorous. Consider a discrete time aperiodic and irreducible Markov chain $\{X_n, n \geq 1\}$ over a countable state space $\mathcal{X}$ with transition probability law $p = (p_{x,y} : x, y \in \mathcal{X})$. Suppose a Lyapunov function $V : \mathcal{X} \to \mathbb{R}_+$ exists such that the drift condition

$$\Delta V_p(x) \triangleq \sum_{y \in \mathcal{X}} p_{x,y} V(y) - V(x) \leq \begin{cases} -\epsilon & \text{if } x \in A^c \\ \eta & \text{if } x \in A \end{cases}$$

is satisfied where $A$ is a finite subset of $\mathcal{X}$, and $\epsilon, \eta > 0$. The reason for using $p$ in the subscript will become apparent shortly. The above bound shows stability of a Markov chain by proving positive recurrence.

Let us define a new transition probability law $p' = (p'_{x,y} : x, y \in \mathcal{X})$ such that

$$p'_{x,y} = \begin{cases} p_{x,y} & \text{if } x \in \Gamma^c \\ r_{x,y} & \text{if } x \in \Gamma, \end{cases}$$

where $r_{x,y}$ are the new transition probabilities and $\Gamma$ is a finite subset of $\mathcal{X}$. The new Markov chain $\{X'_n, n \geq 1\}$ with transition probability law $p'$ is

required to be aperiodic and irreducible. When the same Lyapunov function $V$ for $\{X_n, n \geq 1\}$ is used for $\{X'_n, n \geq 1\}$, it is seen that

$$\Delta V_{p'}(x) \leq \begin{cases} -\epsilon & \text{if } x \in (A \cup \Gamma)^c \\ \eta' & \text{if } x \in (A \cup \Gamma) \end{cases}$$

holds for some new $\eta' > 0$. This satisfies the Foster-Lyapunov criterion for stability. The argument shows that a Markov chain keeps its stability property when the transition probability law is changed over a finite set $\Gamma \subset \mathcal{X}$ while maintaining aperiodicity and irreducibility.

The above argument for stability holds for any throughput optimal algorithm (with an aperiodic and irreducible underlying Markov chain) with transition probabilities redefined over a finite subset (call them "petite" states). Call the remaining state space as "large" states. In the hybrid weighted back-pressure framework, the back-pressure algorithm is the throughput optimal algorithm. The transition probabilities are changed over the petite set by employing a weighted back-pressure algorithm on petite sets. Hybrid weighted back-pressure algorithms are discussed next.

### 2.4.1 Hybrid Weighted Back-pressure Algorithms

Define $U(\xi_t)$ as

$$U(\xi_t) = \sum_{f \in \mathcal{F}} \sum_{n \in \mathcal{N}} X^2_{n,f}(t). \tag{2.12}$$

The optimal schedule in a hybrid weighted back-pressure algorithm is chosen by

$$\hat{c} = \begin{cases} \arg\max_{c \in \mathcal{S}} \sum_{l \in \mathcal{L}} v_l(t)^{\intercal} c_l & \text{if } \xi_t \in \chi_z \\ \arg\max_{c \in \mathcal{S}} \sum_{l \in \mathcal{L}} v_l(t)^{*\intercal} c_l & \text{otherwise,} \end{cases} \tag{2.13}$$

where the petite set $\chi_z := \{\xi \in \Xi : U(\xi) < z\}$. For any real $z \geq 0$, $\chi_z$ is finite. Although our argument holds for any choice of petite set, the

choice of $U(\xi_t)$ makes the proof of stability for the hybrid weighted back-pressure algorithm immediate. The following proposition proves throughput optimality of hybrid weighted back-pressure algorithms.

**Proposition 2.** *Any hybrid weighted back-pressure algorithm defined as above is throughput optimal.*

*Proof of Proposition 2.* Let $b$ be the parameter used in Eq (A.26) of [4]. The proof is a straight forward extension of the proof of Lemma 3.2 in [4] by choosing $V(\xi_t) = U(\xi_t)$ and characterizing the finite set as $\chi_d$ where $d = \max\{d, b\}$. □

The complete algorithm is listed in Algorithm 1. The parameter $z$ is left as a design choice in the algorithm. The effect of varying $z$ is shown in Section 2.5.

### 2.4.2 Hybrid Remaining-hops Weighted Back-pressure Algorithm

A hybrid algorithm is presented here. The remaining-hops weighted back-pressure algorithm of Section 2.3.4 is used to define weights over petite states. The petite set is defined as $\chi_z$. For convenience, this algorithm and the remaining-hops weighted back-pressure algorithm are referred to as HRH and RH respectively in the later sections.

## 2.5 Simulation Results

We compare the back-pressure algorithm, the model in [5] (identified as the "Ying et al" model in the figures), the remaining-hops weighted back-pressure (RH) algorithm, and the hybrid remaining-hops weighted back-pressure (HRH) algorithm via simulations. Poisson arrivals and the primary

**Algorithm 1** Steps run in an iteration of a hybrid weighted back-pressure algorithm

{assign weights}
**for all** $l \in \mathcal{L}$ **and** $f \in \mathcal{F}$ **do**
$w_{l,f}(t) = [P_{l,f}(t)\beta_{l,f}(t)]^+$
**end for**
$v_l(t) \leftarrow \max_{f \in \mathcal{F}} w_{l,f}(t)$
$v_l^*(t) \leftarrow \max_{f \in \mathcal{F}} P_{l,f}(t)$

Solve for $\hat{c}$ as in (2.13) {identify optimal schedule}

{execute $\hat{c}$}
**for all** $l$ such that $\hat{c}_l = 1$ **do**

$$\hat{f}(t) \leftarrow \begin{cases} \arg\max_{f \in \mathcal{F}} \ w_{l,f}(t) & \text{if } \xi_t \in \chi_z \\ \arg\max_{f \in \mathcal{F}} \ P_{l,f}(t) & \text{otherwise} \end{cases}$$

Activate flow $\hat{f}(t)$ on link $l$
**end for**

interference constraints are assumed. The arrival rate ($\lambda$) in subsequent plots is the rate at which traffic is generated for any sink node. The average end-to-end delay versus $\lambda$ is plotted. The effect of $\alpha$ and $z$ is first described before presenting our results on various networks.

### 2.5.1 Effect of $\alpha$ and $z$

The RH and the HRH algorithms assign higher priority to shorter paths by introducing multiplicative weights $\beta_{f,l}$ as in (2.11) that assign higher weights to links on shorter paths. As $\alpha$ increases, the maximal schedule activated in a slot is increasingly determined by the backlogs on shorter flows and

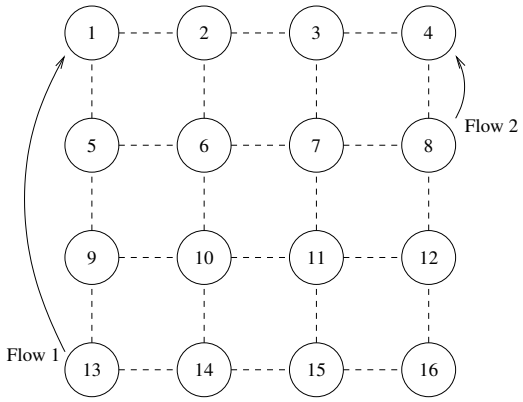


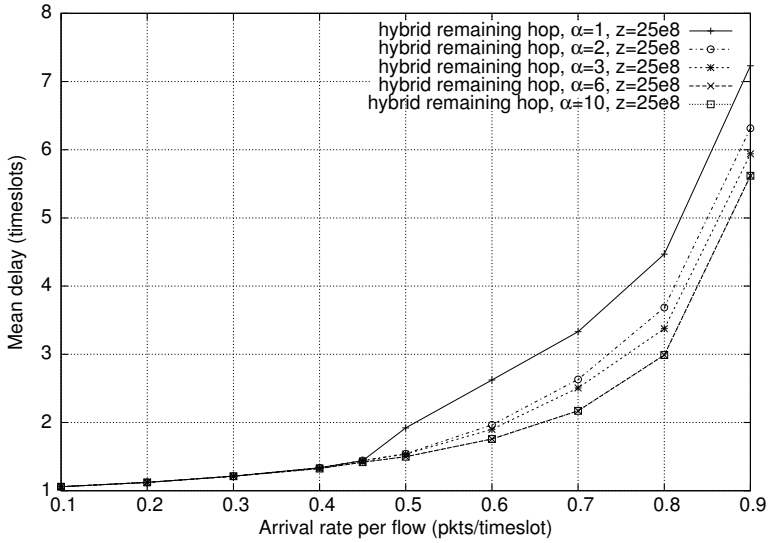

Figure 2.7: The 4×4 grid topology with 2 flows.

shorter paths. This is evident as max $\sqrt[\alpha]{\sum_k y_k^\alpha}$ is dominated by $\max_k\{y_k\}$ as $\alpha$ increases. Consider the network in Fig. 2.7. The effect of varying $\alpha$ is depicted in Fig. 2.8a. Note that the flows are not restricted to the shortest paths. We see that the delay on the shorter flow decreases as $\alpha$ increases.

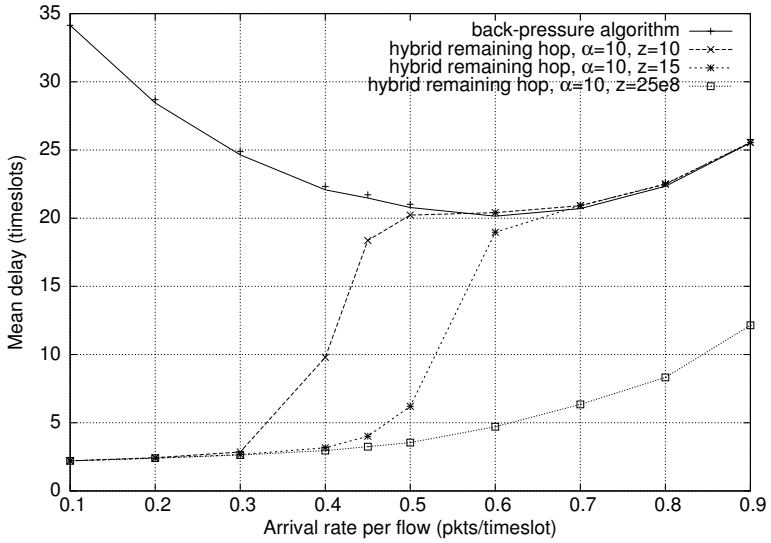
The effect of varying $z$ is accurately reflected in the simulations. For a given $z$, the algorithm uses the link weights from the back-pressure algorithm when the queue backlogs are sufficiently high. Thus, for arrival rates beyond a certain magnitude, the mean delay resembles that of the back-pressure algorithm. As $z$ increases, this transitioning arrival rate increases as longer queue lengths are required before switching to weights from the back-pressure algorithm. This is reflected in Fig. 2.8b where the transitioning arrival rate is seen to increase as $z$ increases.

### 2.5.2 Large Networks

The results presented in this section are representative of results obtained for various topologies. The algorithm of [5], the RH algorithm, and the HRH algorithm have comparable performance (see Fig. 2.9) as demonstrated for the $4 \times 4$ grid topology in Fig. 2.6a and the random topology in Fig. 2.6b.

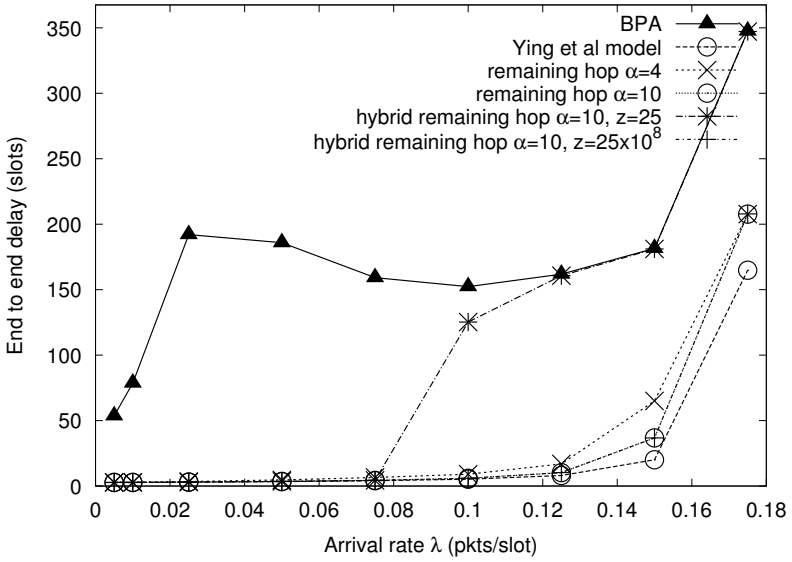



(a) Effect of $\alpha$ on mean delay of Flow 2

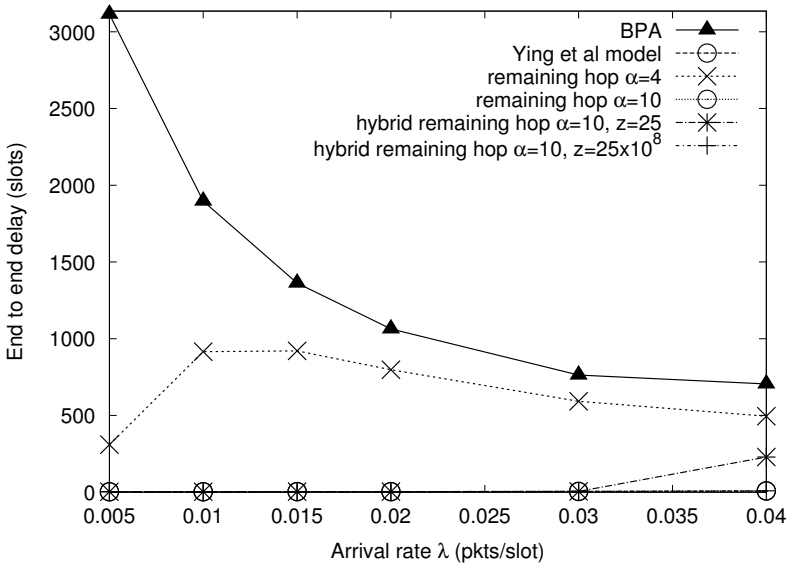



(b) Effect of $z$ on mean delay

Figure 2.8: The effect of $\alpha$ and $z$ on the topology in Fig. 2.7.

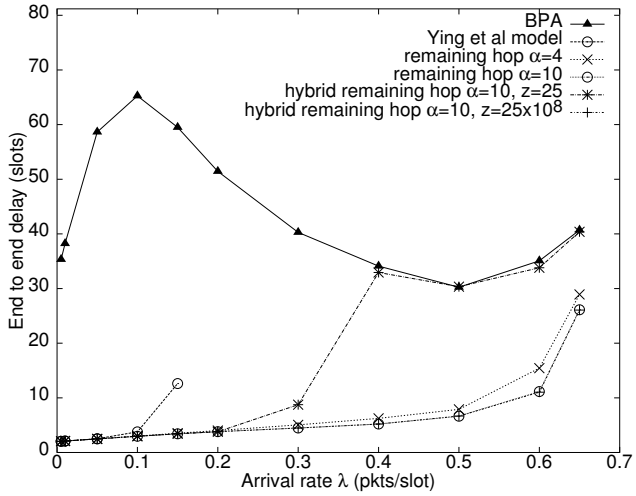



(a) Grid network of Fig. 2.6a

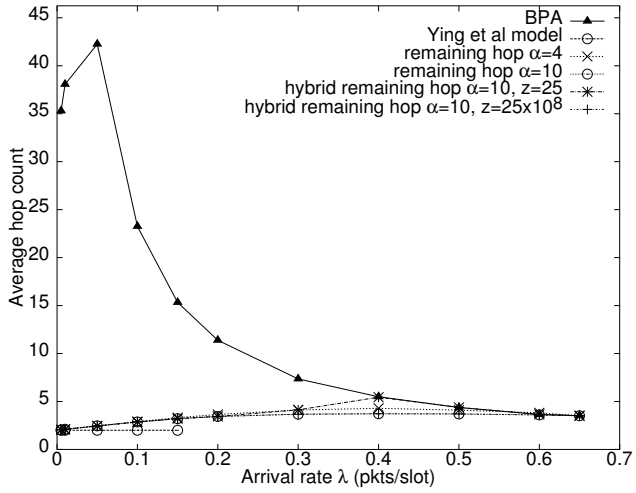



(b) Random network of Fig. 2.6b

Figure 2.9: Mean delay of different algorithms.




(a) End-to-end delay




(b) Average hops count

Figure 2.10: Performance of various algorithms on a network with short routes sharing a congested node (see Fig. 2.6c).

The algorithms perform well by reducing the hop count at low and moderate loads. The topology in Fig. 2.6c is motivated next.

The RH and the HRH algorithms route more packets on shorter paths. One may suspect the two algorithms to perform poorly when shorter routes pass through the congested part of the network. This motivates the topology in Fig. 2.6c where three flows with the shortest path of two hops, exist from Node 6 to Node 2, from Node 11 to Node 7, and from Node 16 to Node 12. Note that all shortest paths pass through Node 1, the node expected to be most congested in the network. All links are half-duplex. Alternate four-hop long routes also exist between the source and destination nodes. Because of dynamic routing, the packets may get misrouted to even longer routes in other loops from Node 1. We find that the RH and the HRH algorithms perform well though the model of [5] performs poorly at moderate loads. The algorithm of [5] saturates Node 1 at approximately 0.16 packets/timeslot and thus the delay rises sharply close to this reduced capacity of the network. This limit is evident as even under no misrouting of packets, Node 1 is activated twice for each flow (to receive in one timeslot and transmit to the sink in the next) and there are 3 flows. This algorithm does not stabilize the network for higher loads.

## 2.6 Summary

We identify three causes for the poor delay performance of the back-pressure algorithm. We observe that the interaction of flows and the topology itself can lead to poor delay performance. This chapter chiefly addresses the third causation of the random walk like behaviour by increasing the proclivity of the packets to take shorter paths to the destination dynamically. This is achieved by introducing a framework where the priority of certain flows and links are increased by assigning higher weights. We demonstrate that

by leveraging path length information, the delay performance of the back-pressure algorithm can be improved significantly.

One approach for distributed implementation is following the work in [34] and the references therein. The cost of distributed implementation is a possible reduction in the capacity as the activated schedules are sub-optimal solutions to the weighted matching problem.

# Chapter 3

# A Static Formulation for Reducing Delay

## 3.1 Introduction

In non-stationary environments, a max-weight algorithm for routing and scheduling is suitable since it strives to maintain stability, e.g., the back-pressure algorithm which is throughput optimal but does not explicitly minimize delay. Such algorithms typically entail high per timeslot complexity. Under stationary settings, a static problem can be formulated for minimizing mean delays in the network while maintaining stability. In this chapter, we take this approach for multihop wireless networks. The approach is inspired by the early works on delay in wired networks. Wired network models invariably assume link capacities to be constants and routing parameters (either single-path or multi-path) to be the variables in the formulation to minimize delay. In an extension to wireless networks, interference is one reason for variable link capacities. This extension, as we will see, becomes non-trivial since the (delay) objective function is nonconvex.

We take a static approach to route and schedule packets while minimizing

delay; the static solution does not depend on the instantaneous network state information and slows the timescale of routing and scheduling computations from per slot to the timescale of changes in the network traffic pattern.

### 3.1.1 Background and Related Works

Multipath routing for mean delay minimization has been extensively studied for wireline networks, e.g., see [35, 36, 37, 38]. Among other techniques, these approaches have included the flow deviation method and the projected gradient method to optimize delay, e.g., [35, 36]. Second derivative based techniques have been used for faster convergence, e.g., [37]. As has been noted already, routing was the only variable considered in these formulations and the delay objective was a convex function. We will see that the objective is nonconvex when scheduling parameters (link capacities) are variable. Some standard nonlinear optimization techniques are discussed next.

The flow-deviation method applied to minimize delay in a wired network (see [35, 39]) requires moving part of the flow from all non-shortest paths (inferred from the first derivative of delay on a path) to the shortest path. The algorithm converges to the global minima for convex delay functions. In the projected gradient method, the direction of steepest descent, $-\nabla f$, is identified first. If even a small step of $\epsilon$ in $-\nabla f$ direction makes the point infeasible, the new search direction is set to the projection of the gradient onto the set of feasible directions ($g = \mathsf{P}(-\nabla f)$). The iteration completes by taking an optimally sized step in the search direction. In a similar algorithm, a step is made in the direction of the steepest descent and the resulting point is projected back onto the set of feasible solutions. The algorithms converge to a local minima for nonconvex problems. See [39, 40, 41] for more details.

The block descent algorithm almost surely converges to the global minima for a convex problem and converges to a local minima for a nonconvex

problem. A single iteration of the algorithm breaks the task into solving several optimization problems over smaller spaces. Specifically, the objective is sequentially optimized in every block (sub space of the feasible space) in an iteration; the algorithm repeats this iteration until the algorithm converges. See [39, 41, 42] for further details.

We mention simulated annealing next, a random algorithm that converges to the global minima but has a very slow convergence rate (see [43, 44, 45] for example). The algorithm generates a sequence of states $\{\vec{x}_n\}$ that converge to the global optima in distribution. The stochastic process $\{\vec{x}_n\}$ is a time inhomogeneous Markov process where the transition probability from $\vec{x}$ to $\vec{x}'$ depends on an external scheduling parameter $c_n$ typically referred to as the temperature. The transition probabilities also depend on the difference in objective at $\vec{x}$ and $\vec{x}'$. The algorithm takes a 'non-greedy' approach as the objective can increase in an iteration though such transitions are less likely to occur than transitions to lower objective values. The role of the temperature schedule $\{c_n\}$ is to characterize the probability of making upward transitions: transitions to higher objective values are more likely when the temperature is high. The convergence guarantees hold when the cooling schedule $\{c_n\}$ is sufficiently slow. The cooling schedule depends on the size of the domain and the size of a neighbourhood (see [43, 46]). These reasons are attributed to the infeasibility of this algorithm to our problem.

### 3.1.2 Outline of this Chapter

In Section 3.2, we formulate a static model for minimizing delay and present our analysis. Section 3.3 characterizes the set of local minima and presents two class of algorithms that solve the optimization problem from Section 3.2. We present two heuristical implementations and evaluate them in Section 3.4. Our results are summarized in Section 3.5.

## 3.2 Static Formulation for Minimizing Delay

In this section, we pose a static optimization problem for minimizing average delay, along the lines of [35, 36, 37]. Here, a flow is defined by the ordered set of the sending node ($s_f$) and the receiving node ($d_f$). Let the packet arrival rate of flow $f$ be $\lambda_f$. We assume that a queue exists at each node for each flow. Changes in notation from Chapter 2 are briefly defined. Consider an ergodic schedule in which schedule $s$ is used for fraction $\phi_s$ of time. We will say that $\phi := (\phi_s, s \in \mathcal{S})$ is the *scheduling* in the network. Clearly, $\phi_s \geq 0$ and $\sum_{s\in\mathcal{S}} \phi_s = 1$; this defines the set $\Phi$ of feasible scheduling vectors. The capacity (in packets/slot) of link $l$ is thus $\mu_l := \sum_{s\in\mathcal{S}} \phi_s s_l$. We will assume ergodic routing and let $x_{f,l}$, $f \in \mathcal{F}$ and $l \in \mathcal{L}$, denote the rate at which packets of flow $f$ are transmitted on link $l$. Let $x_f := (x_{f,l}, l \in \mathcal{L})$, $x := (x_f, f \in \mathcal{F})$. We will say that $x$ is the *routing* in the network. Clearly, for each $f \in \mathcal{F}$, the vector $x_f$ is non-negative and satisfies the following flow conservation constraints: for all $i \in \mathcal{N}$ and $f \in \mathcal{F}$,

$$\sum_{l\in\mathcal{L}^i(i)} x_{f,l} = \begin{cases} \sum_{l\in\mathcal{L}^e(i)} x_{f,l} + \lambda_f & \text{if } i = d_f, \\ \sum_{l\in\mathcal{L}^e(i)} x_{f,l} - \lambda_f & \text{if } i = s_f, \\ \sum_{l\in\mathcal{L}^e(i)} x_{f,l} & \text{otherwise.} \end{cases}$$

These constraints define the feasibility set $\mathsf{X}_f$ for $x_f$; let $\mathsf{X} := \prod_{f\in\mathcal{F}} \mathsf{X}_f$. The rate at which packets arrive for transmission on link $l$ is $\gamma_l := \sum_{f\in\mathcal{F}} x_{f,l}$.

Let $D_l(\gamma_l, \mu_l)$ reflect the mean delay on link $l$ as a function of the packet arrival rate $\gamma_l$ and the mean service rate $\mu_l$. In general, this function can depend on other statistical parameters but our interest is in optimizing delay in these two parameters. Such assumptions have been used in previous work, e.g., see [36, 37, 47].

The mean packet delay in the network is minimized by minimizing the average number of packets in the system obtained by Little's Law, i.e., by

minimizing $G := \sum_l \gamma_l D_l(\gamma_l, \mu_l)$. For numerical evaluation in Section 3.4, we assume the mean delay function to be identical at each link. This assumption is made for ease of solving. The exact arrival distribution for modelling downstream links in a network is difficult to characterize even for simple Poisson external arrivals.

We make the following regularity assumptions on the link delay function $D_l$. Define $Z := \{(\gamma_l, \mu_l) \in \mathbb{R}^2 \ : \ \mu_l \geq \epsilon, \ 0 \leq \gamma_l < \mu_l\}$, where $\epsilon > 0$ is a small positive constant. We impose a lower bound $\epsilon$ on $\mu_l$ to eliminate a possible discontinuity at the origin of the function $\gamma_l D_l(\gamma_l, \mu_l)$. $D_l$ is twice continuously differentiable over $Z$, and is defined to be $\infty$ outside this set. Over its effective domain, $D_l(\gamma_l, \mu_l)$ is strictly increasing and convex with respect to $\gamma_l$ and strictly decreasing and convex with respect to $\mu_l$. Finally, $\lim_{\gamma_l \uparrow \mu_l} D_l(\gamma_l, \mu_l) = \infty$ for all $\mu_l \geq \epsilon$. These assumptions are natural to make and are properties that hold for common queueing systems, e.g., they are applicable for the M/G/1 server with first come first serve discipline.

We are interested in optimally choosing routing $x$ and scheduling $\phi$ to minimize the mean packet delay in the network. Formally, this optimization problem can be stated as follows:

$$\begin{aligned} \min \quad & G(x, \phi) \\ \text{subject to} \quad & (x, \phi) \in \mathsf{X} \times \Phi \end{aligned} \tag{P1}$$

We assume that the packet arrival rates $\lambda_f$, $f \in \mathcal{F}$, are within the capacity region of the network; this ensures that the above optimization problem is feasible. Also, note that we do not explicitly include a stability constraint for each link since the objective function is defined to be $\infty$ if the constraints are violated.

The function $D_l(\gamma_l, \mu_l)$ is meant to model the average delay on link $l$. A standard approach would be to model $D_l$ using the Pollaczek-Khinchin formula for the M/G/1 FCFS queue (as in [36, 37, 47]), with $\gamma_l$ and $\mu_l$

denoting the arrival rate and the service rate respectively. Note that in all these models, $\lim_{\gamma_l \to 0} D_l(\gamma_l, \mu_l) = \frac{1}{\mu_l}$. The following proposition states that for such 'standard' queueing delay models, the average number of packets in the queue (including the one being served), given by $\gamma_l D_l(\gamma_l, \mu_l)$ is not jointly convex with respect to $\gamma_l$ and $\mu_l$. This suggests that (P1) is in general a non-convex optimization problem for standard queueing delay models.

**Proposition 3.** *If $D_l$ satisfies the regularity conditions listed above and $D_l(0, \mu_l) = \frac{1}{\mu_l}$, then $\gamma_l D_l(\gamma_l, \mu_l)$ cannot be convex over the interior of $Z$.*

*Proof of Proposition 3.* Let $F(\gamma, \mu) := \gamma D_l(\gamma, \mu)$. Pick $\mu_0 > \epsilon$. Since $\frac{\partial D_l(\gamma,\mu)}{\partial \mu}$ is assumed to be continuous over $Z$, $\lim_{\gamma \downarrow 0} \frac{\partial D_l(\gamma,\mu_0)}{\partial \mu} < 0$.

A necessary condition for $F$ to be convex over $Z$ is that the determinant of its Hessian (denoted by $\det(\nabla^2 F)$) be non-negative. $\det(\nabla^2 F)$ is easily computed as

$$\det(\nabla^2 F(\gamma, \mu)) = \gamma \frac{\partial^2 D}{\partial \mu^2} \left( 2 \frac{\partial D}{\partial \gamma} + \gamma \frac{\partial^2 D}{\partial \gamma^2} \right) - \left( \frac{\partial D}{\partial \mu} + \gamma \frac{\partial^2 D}{\partial \mu \partial \gamma} \right)^2 .$$

Since $D$ is assumed to be twice continuously differentiable over $Z$, all the derivatives on the right hand side of the above equation must be bounded over the compact set $\{(\gamma, \mu_0) : 0 \leq \gamma \leq \mu_0/2\}$. Therefore,

$$\lim_{\gamma \downarrow 0} \det(\nabla^2 F(\gamma, \mu_0)) < 0.$$

This implies that for small enough $\gamma > 0$, $\det(\nabla^2 F(\gamma, \mu_0)) < 0$. Therefore, $F$ cannot be convex over $Z$. □

If we fix the scheduling vector $\phi$, then it is easy to see that (P1) is convex and reduces to the optimal routing problem of [36, 37]. Similarly, if we fix the routing $x$, then (P1) reduces to a convex optimization and is therefore easy to solve. Proposition 3 above suggests that with both routing and scheduling as variables, (P1) is in general a non-convex optimization. We now provide algorithms to compute a local minimum of (P1).

## 3.3 Optimal Solutions

First, we characterize the set of local minimizers of (P1). Define $J = \{(x, \phi) \in \mathsf{X} \times \Phi \mid G(x, \phi) < \infty\}$, the set of all feasible solutions. Since we have assumed the optimization in (P1) is feasible, the set $J$ is non-empty. Define

$$J^* = \left\{ (\bar{x}, \bar{\phi}) \in J \;\middle|\; \begin{array}{l} \bar{x} \in \arg\min_{x \in \mathsf{X}} G(x, \bar{\phi}) \\ \bar{\phi} \in \arg\min_{\phi \in \Phi} G(\bar{x}, \phi) \end{array} \right\},$$

the set of all optimal solutions.

**Lemma 1.** *$J^*$ is the set of local minimizers of $G$ over $\mathsf{X} \times \Phi$.*

*Proof of Lemma 1.* The proof follows easily from the fact that $\mathsf{X}$ and $\Phi$ are convex, and that a minimization with respect to either variable $x$ or $\phi$ keeping the other fixed is a convex minimization. Since $\mathsf{X}$ and $\Phi$ are convex sets, $(\bar{x}, \bar{\phi})$ is a local minimizer of $G$ over $\mathsf{X} \times \Phi$ iff

$$\begin{array}{rl} & \nabla_x G(\bar{x}, \bar{\phi}) \cdot (x - \bar{x}) + \nabla_\phi G(\bar{x}, \bar{\phi}) \cdot (\phi - \bar{\phi}) \geq 0 \\ & \qquad \forall \quad (x, \phi) \in \mathsf{X} \times \Phi \\ \iff & \begin{array}{l} \nabla_x G(\bar{x}, \bar{\phi}) \cdot (x - \bar{x}) \geq 0 \quad \forall \quad x \in \mathsf{X}, \\ \nabla_\phi G(\bar{x}, \bar{\phi}) \cdot (\phi - \bar{\phi}) \geq 0 \quad \forall \quad \phi \in \Phi \end{array} \\ \iff & \begin{array}{l} \bar{x} \in \arg\min_{x \in \mathsf{X}} G(x, \bar{\phi}), \\ \bar{\phi} \in \arg\min_{\phi \in \Phi} G(\bar{x}, \phi). \end{array} \end{array}$$

□

The preceding lemma states that the set of local minimizers of (P1) are precisely the tuples $(x, \phi)$ satisfying the property that with fixed routing vector $x$, the scheduling vector $\phi$ is optimal, and vice-versa. Clearly, if $G$ is convex, then $J^*$ is the set of global minimizers of $G$ over $\mathsf{X} \times \Phi$.

We now provide two approaches to compute a local minimizer of (P1). The first is a block descent algorithm which cyclically performs (convex)

optimizations with respect to routing and scheduling. Second is a class of algorithms that chooses between a routing update and a scheduling update in each iteration.

### 3.3.1 Block Descent Algorithm

We now describe an algorithm (see Algorithm 2) for computing a local minimum of (P1) based on the block descent algorithm (see Proposition 2.7.1 of [41]). Let $(x^0, \phi^0) \in J$ denote a starting feasible point for (P1). Such a point is easy to compute since $J$ is defined by linear constraints. The algorithm is parametrized by a positive constant $c$.

**Algorithm 2** Block descent

**for** $i \geq 0$ **do**

$\phi^{i+1} \leftarrow \arg\min_{\phi \in \Phi} G(x^i, \phi) + \frac{1}{c} \parallel \phi - \phi^i \parallel^2$

$x^{i+1} \leftarrow \arg\min_{x \in \mathsf{X}} G(x, \phi^{i+1}) + \frac{1}{c} \parallel x - x^i \parallel^2$

**end for**

Note that the optimizations involved in each iteration of Algorithm 2 are convex, and hence can be performed by standard techniques. The following lemma guarantees the convergence of this block descent algorithm.

**Lemma 2.** *The sequence* $\{(x^i, \phi^i)\}$ *generated by Algorithm 2 converges to an element of* $J^*$.

*Proof of Lemma 2.* Invoking Proposition 2.7.1 of [41], the sequence $\{(x^i, \phi^i)\}$ converges to a local minimum of (P1) if, for any feasible point $(x^0, \phi^0)$ of (P1), the optimizations

$$\min_{x \in \mathsf{X}} G(x, \phi^0) + \frac{1}{c} \parallel x - x^0 \parallel^2, \text{ and}$$
$$\min_{\phi \in \Phi} G(x^0, \phi) + \frac{1}{c} \parallel \phi - \phi^0 \parallel^2$$

yield unique minimizers. The quadratic term is added to the objective function to guarantee a unique solution by its strict convexity. □

### 3.3.2 A Class of Iterative Algorithms

Next, we introduce another class of iterative algorithms (see Algorithm 3) that guarantee convergence to $J^*$. Such an algorithm is specified by two algorithmic maps: a 'routing update' mapping $A_{RT} : J \to \mathsf{X}$ which provides descent by updating the routing vector and a 'scheduling update' mapping $A_{SC} : J \to \Phi$ which provides descent by updating the scheduling vector. We require $A_{RT}$ and $A_{SC}$ to be closed maps. Let $X, Y$ be closed sets in $J$. A map $A : X \to Y$ is closed at $x \in X$ if

$$\left.\begin{array}{ll} x_k \in X & x_k \to x \\ y_k \in A(x_k) & y_k \to y \end{array}\right\} \text{ implies that } y \in A(x).$$

The map $A$ is closed on a subset $Z \subset X$ if it is closed at each point $z \in Z$. See [42, Chapter 7] for a discussion on algorithmic maps.

Algorithm 3 describes the algorithm derived from $A_{RT}$ and $A_{SC}$. As before, we assume that a starting feasible point $(x^0, \phi^0) \in J$ is available.

**Algorithm 3** Algorithm for solving (P1)

**for** $i \geq 0$ **do**
  **if** $G(A_{RT}(x^i, \phi^i), \phi^i) \leq G(x^i, A_{SC}(x^i, \phi^i))$ **then**
    $x^{i+1} \leftarrow A_{RT}(x^i, \phi^i)$ ; $\phi^{i+1} \leftarrow \phi^i$
  **else**
    $x^{i+1} \leftarrow x^i$ ; $\phi^{i+1} \leftarrow A_{SC}(x^i, \phi^i)$
  **end if**
**end for**

In each iteration, the algorithm performs either the routing update or the scheduling update, whichever produces the greater descent in the objective

function value. The following theorem states that so long as the routing update $A_{RT}$ and the scheduling update $A_{SC}$ are continuous, and satisfy the following descent criteria, the sequence $\{(x^i, \phi^i)\}$ generated by the algorithm converges to the set $J^*$. Since $J^*$ characterizes the set of local minima, the algorithmic maps $A_{RT}$ and $A_{SC}$ are assumed to generate a feasible point and decreasing objective values. Most importantly, the descent properties required of the routing and scheduling updates are decoupled implying that these update rules can be designed independently.

**Theorem 1.** *For any $(\hat{x}, \hat{\phi}) \in J$ satisfying $G(\hat{x}, \hat{\phi}) \leq G(x^0, \phi^0)$, assume that the routing update $A_{RT}$ satisfies the following descent property.*

*(a) If $\hat{x} \in \arg\min_{x \in \mathsf{X}} G(x, \hat{\phi})$, then $G(A_{RT}(\hat{x}, \hat{\phi}), \hat{\phi}) = G(\hat{x}, \hat{\phi})$;*

*(b) if $\hat{x} \notin \arg\min_{x \in \mathsf{X}} G(x, \hat{\phi})$, then there exists $\delta > 0$ such that for all $(\tilde{x}, \tilde{\phi}) \in \mathsf{X} \times \Phi$ satisfying $\| (\tilde{x}, \tilde{\phi}) - (\hat{x}, \hat{\phi}) \| < \delta$, $G(A_{RT}(\tilde{x}, \tilde{\phi}), \tilde{\phi}) < G(\hat{x}, \hat{\phi})$.*

*Similarly, for any $(\hat{x}, \hat{\phi}) \in J$, satisfying $G(\hat{x}, \hat{\phi}) \leq G(x^0, \phi^0)$, assume that the scheduling update $A_{SC}$ satisfies the following descent property.*

*(a') If $\hat{\phi} \in \arg\min_{\phi \in \Phi} G(\hat{x}, \phi)$, then $G(\hat{x}, A_{SC}(\hat{x}, \hat{\phi})) = G(\hat{x}, \hat{\phi})$;*

*(b') if $\hat{\phi} \notin \arg\min_{\phi \in \Phi} G(\hat{x}, \phi)$, then there exists $\delta > 0$ such that for all $(\tilde{x}, \tilde{\phi}) \in \mathsf{X} \times \Phi$ satisfying $\| (\tilde{x}, \tilde{\phi}) - (\hat{x}, \hat{\phi}) \| < \delta$, $G(\tilde{x}, A_{SC}(\tilde{x}, \tilde{\phi})) < G(\hat{x}, \hat{\phi})$.*

*Then every limit point of the sequence $\{(x^i, \phi^i)\}$ generated by Algorithm 3 lies in $J^*$.*

*Proof of Theorem 1.* The proof is similar to the convergence proof of Theorem 7.2.3 in [42]. The descent assumptions on $A_{RT}$ and $A_{SC}$ imply that

$\{G(x^i, \phi^i)\}$ is a non-increasing sequence (bounded below by the value of $G$ at the solution of (P1)). Therefore, there exists $G^* \in \mathbb{R}$ such that

$$\lim_{i\to\infty} G(x^i, \phi^i) = G^*, \quad G(x^i, \phi^i) \geq G^* \quad \forall\ i. \tag{3.1}$$

Now, since the sequence $\{(x^i, \phi^i)\}$ is contained in the sublevel set $\{(x, \phi) : G(x, \phi) \leq G(x^0, \phi^0)\}$, a compact space in $\mathsf{X} \times \Phi$, it must have a converging subsequence $(x^{i(n)}, \phi^{i(n)}) \overset{n\uparrow\infty}{\to} (x^*, \phi^*)$. Note that since $G$ is continuous, $G(x^*, \phi^*) = G^*$. We need to prove that $(x^*, \phi^*) \in J^*$.

Let us assume (for the sake of obtaining a contradiction) that $(x^*, \phi^*) \notin J^*$. Then from the definition of $J^*$, at least one of the following conditions must hold.

(i) $x^* \notin \arg\min_{x\in\mathsf{X}} G(x, \phi^*)$

(ii) $\phi^* \notin \arg\min_{\phi\in\Phi} G(x^*, \phi)$

Let us say (i) holds. Then from the descent assumption on $A_{RT}$, there exists $\epsilon > 0$ such that for all $(\tilde{x}, \tilde{\phi}) \in \mathsf{X} \times \Phi$ satisfying $\parallel (\tilde{x}, \tilde{\phi}) - (x^*, \phi^*) \parallel < \epsilon$, we must have $G(A_{RT}(\tilde{x}, \tilde{\phi}), \tilde{\phi}) < G(\hat{x}, \hat{\phi})$. Since $(x^{i(n)}, \phi^{i(n)}) \overset{n\uparrow\infty}{\to} (x^*, \phi^*)$, there exists $n_0 \in \mathbb{N}$ such that $\parallel (x^{i(n_0)}, \phi^{i(n_0)}) - (x^*, \phi^*) \parallel < \epsilon$, which implies that

$$G(A_{RT}(x^{i(n_0)}, \phi^{i(n_0)}), \phi^{i(n_0)}) < G(x^*, \phi^*) = G^*. \tag{3.2}$$

Algorithm 3 picks $(x^{i(n_0)+1}, \phi^{i(n_0)+1})$ such that

$$G(x^{i(n_0)+1}, \phi^{i(n_0)+1}) \leq G(A_{RT}(x^{i(n_0)}, \phi^{i(n_0)}), \phi^{i(n_0)}). \tag{3.3}$$

Combining (3.2) and (3.3), we conclude that $G(x^{i(n_0)+1}, \phi^{i(n_0)+1}) < G^*$, which is a contradiction. This means that Condition (i) above cannot hold. Using an identical argument, it can be shown that Condition (ii) also cannot hold. Therefore, $(x^*, \phi^*) \in J_r^*$. □

Since (P1) reduces to a convex optimization when either the routing vector or the scheduling vector is fixed, the update rules $A_{RT}$ and $A_{SC}$ may be designed by standard techniques in convex optimization. In particular, the projected gradient methods, such as in [41], satisfy the descent requirements and qualify in this class of iterative algorithms.

## 3.4 Evaluation

In this section we evaluate the mean delay performance, via simulations, of the static formulation in (P1). We make the assumption that the delay at each link is given by $1/(\mu_l - \gamma_l)$. The performance of the algorithms is compared with that of the back-pressure algorithm, the HRH algorithm, and the model of [5]. The simulations are run on a $4 \times 4$ grid topology with 240 flows (see Fig. 2.6a).

The $x$ and $\phi$ chosen as a solution to (P1) can be implemented via many schemes; the delay performance of the routing and scheduling scheme will depend on the chosen scheme. A simple static routing algorithm suggests itself. At node $i$, a packet of flow $f$ is routed on link $l$ with a probability $p_{f,l}$ independent of all other packets, where

$$p_{f,l} := \begin{cases} \frac{x_{f,l}}{\sum_{l' \in \mathcal{L}^e(i)} x_{f,l'}} & \text{if } \sum_{l' \in \mathcal{L}^e(i)} x_{f,l'} > 0 \\ 0 & \text{otherwise.} \end{cases}$$

A simple scheduler, we call the *independent scheduler*, also suggests itself. In each slot, schedule $s \in \mathcal{S}$ is chosen with probability $\phi_s$ independent of all other slots. In this scheme, the interval of time between the activations of schedule $s$ has high variance. Reducing this variance can reduce the mean delays in the network. This leads us to the *max-delta scheduler* which is a generalization of the round-robin scheduler. Let $\Pi(t)$ be the schedule acti-

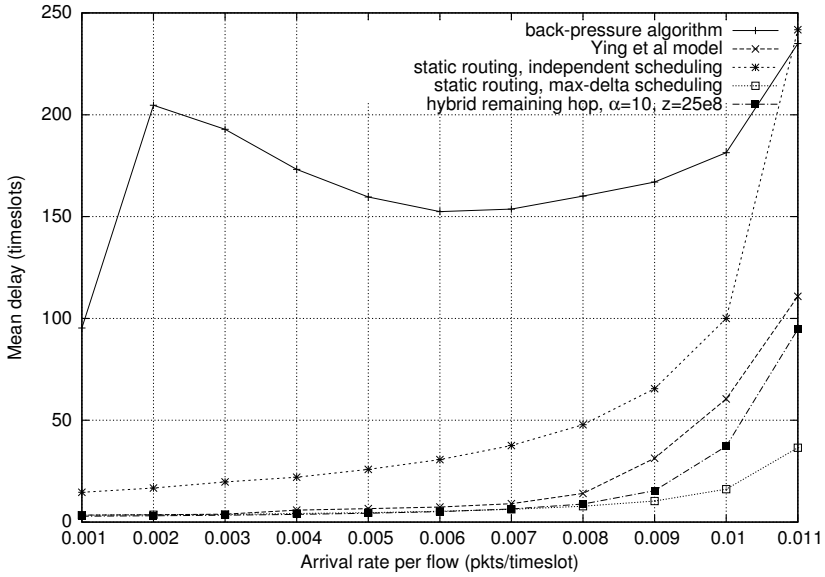



Figure 3.1: Mean end-to-end delay in a $4 \times 4$ grid network (see Fig. 2.6a) when $x$ and $\phi$ are chosen from (P1) and implemented using the static routing and scheduling schemes. Performance curves of the schemes of [4] and [5] are also provided.

vated in slot $t$. Define

$$\hat{\phi}_s(t) := \frac{1}{t} \sum_{u=0}^{t-1} \mathbf{1}(\Pi(u) = s),$$

the fraction of time schedule $s$ has been activated up to time $t$. The schedule activated during time slot $t$ is determined by

$$\Pi(t) = \arg\max_{s \in \mathcal{S}} \{\phi_s - \hat{\phi}_s(t)\},$$

where ties are broken according to some pre-determined rule. Observe that this gives us a deterministic sequence of schedules which can be pre-computed.

Fig. 3.1 shows the poor delay performance of the back-pressure algorithm. We see that the independent scheduling algorithm with static routing

performs significantly better than the back-pressure algorithm at low and moderate loads because the static routing prohibits random walk like behaviour. However, since the schedules are chosen randomly in each slot, the interval between consecutive link activations can have a high variance. This and the fact that the schedule in each slot is independent of the queue occupancies causes the delay to continue to be high (though better than that of the back-pressure algorithm) for low loads and become worse than that of the back-pressure algorithm at high loads. The scheme of [5] offers a better performance here. The max-delta scheduler with static routing performs better than the independent scheduler. We attribute this to the reduced variance of the inter-activation time of the links. Figure 3.1 also shows the results for the HRH algorithm discussed in Section 2.4.1. This algorithm performs significantly better than the back-pressure algorithm. Our results on various topologies indicate the scheme of [5], the HRH algorithm, and the static routing with max-delta scheduling to perform about the same. The scheme of [5] however suffers from a reduced stability region as demonstrated in Section 2.5 under certain topologies.

## 3.5 Summary

The focus of this chapter is on reducing mean delays in multihop wireless networks via a static formulation. We prove an interesting result: mean delay functions are not convex for wireless networks when both capacity and routing parameters are variables.

Two algorithms are proposed to obtain the solutions to the static delay problem. The block descent algorithm in Algorithm 2 requires solving for the unique solution to the two minimization problems in every iteration. In contrast, the iterative algorithm in Algorithm 3 only requires an improvement in the objective in every iteration. Although both routing and scheduling

updates are calculated, only one is used and the other is discarded. Presently, the algorithms require the knowledge of the packet arrival rates a priori and solve optimization problems centrally. Promising leads for distributed implementations involve estimating traffic arrival rate, e.g., see [48, 49], and obtaining distributed solutions to optimization problems, e.g., see [36, 50, 51, 52, 53].

We present two self-evident implementations of the static solutions and show that implementing static schemes can achieve significantly better mean delays, a benefit aside from the reduced complexity. We benchmark our results against the popular back-pressure algorithm in [4] and the model of [5].

# Part II

# Pricing of Resources

# Chapter 4

# Pricing of Resources

## 4.1 Introduction

A wired or wireless network represents a collection of available resources and users requiring access to the resources. There are numerous such examples: an Internet Service Provider (ISP) charges for allocating bandwidth (or capacity) to its customers, cloud computing data-centers where CPU time of servers is shared between customers, an online anonymizing proxy service where the proxy's capacity is shared between its users concurrently, cellular networks where again the capacity is shared between end users associated with the base station, and even examples such as printers shared on a network. Sharing of the access medium itself is another example. In such scenarios, the resource may be available *gratis* (for example the unlicensed band) where a user doesn't have to pay for access, or it may be sold by an operator for a fee. We focus our attention to shared resource environments where an operator charges for allocating its resource to users. The question we seek to answer in this chapter is how to share a resource, hereafter assumed to be bandwidth, fairly and get predictable revenue without the operator requiring too much statistical information on users.

Our formulation will consist of a server where users arrive, remain until their service requirements are met, and then depart from the system. The allocation of the resource to a user is assumed to depend on the number of users in the system. Such queueing models typically assume the knowledge of the arrival process and the service requirements for performance analysis. We wish to obtain results on the revenue of the operator and the user costs when only the first-order statistical information on the arrival process and the service requirements are known. Such an allocation mechanism where the stationary distribution of the underlying Markov chain only depends on the first-order moments is called an *insensitive allocation*. Insensitive allocations offer predictability — predictability of revenue to the operator and predictability of costs to the customers.

The processor sharing discipline, where a server allocates equal resources to each user in the system, results in insensitivity. Insensitive allocations also exhibit product-form distributions that allow explicitly evaluating several performance metrics such as dimensioning of capacity or throughput. In the case of a single server system, processor sharing is the socially optimum policy when the users' utilities are the logarithm of their allocated bandwidth. The log utility function is also of interest since its solutions coincide with the Nash bargaining solution (see [54, 55]), with proportional fairness (see [56, 50]), and with insensitive allocations in the case of a single server system (see [57]). We emphasize that the processor sharing discipline, although simple, is of key interest to us since the insensitivity property offers predictability. It allows us to gain insights into the structure of the pricing mechanisms.

Numerous pricing models or variations for resource allocations exist. We assume a usage based pricing model with an implicit assumption of the user having elastic demand, i.e., the user takes whatever portion of the resource is allocated to it. This assumption is not too far off from reality, e.g., the Amazon Elastic Compute Cloud (EC2) and the Amazon Simple Storage Service

(S3) rely on usage based pricing (see [58]).

We restrict ourselves to three pricing models reflecting three different ideologies. In the fixed rate pricing model, the price of per-unit bandwidth decreases as the number of users increases. Contrariwise, the price of per-unit bandwidth increases with the number of users in the congestion based pricing model. The third pricing model is the Vickrey-Clarke-Groves (VCG) auction which we show approximates indifference to the number of users in the system. The three pricing models are detailed in Section 4.3.1.

We note that processor sharing is not an appropriate model for modelling every shared resource. Clearly, a printer on the network cannot be shared concurrently between multiple users and is exclusively allocated to a single user at a time, typically by the First-Come-First-Server (FCFS) discipline. That said, processor sharing models a computing server's resource (e.g., Amazon's EC2) and bandwidth allocations well. The motivation of our work stems from the domain of cloud computing where the end-user and the data center operator require respectively to ascertain their expected costs and revenue. The results can also be used in modelling large file transfers over TCP connections through a payment charging proxy or for web applications where file sizes can have general statistical distributions. Predictability is attained at the cost of not employing admission control. A simple generalization of the processor sharing discipline, the discriminatory processor sharing discipline, is not insensitive and thus its analysis becomes very difficult (see [59]).

In this chapter, we explicitly characterize the mean revenue earned by the operator and the mean payments made by the user under the processor sharing discipline in the three pricing models. The mean payments derived rely on sample path arguments. We present an alternative scheme where the operator charges users upfront on arrival. The analysis of this scheme is based on the stationary measures instead of the sample path arguments.

We also provide the second moment of payments by users in such a scheme which defines the volatility of payments.

### 4.1.1 Background and Related Works

Utility maximization is a well-known technique used for efficiently sharing a resource. The model assumes that each user $r$ has a utility function $U_r(\Lambda)$ which quantifies the personal worth attributed to $\Lambda$ amount of resource. The goal of an efficient sharing policy is to maximize the aggregate of individual utilities also called as the *social welfare*, i.e.,

$$\begin{aligned} \max \quad & \sum_r U_r(\Lambda_r) \\ \text{subject to} \quad & \sum_r \Lambda_r \leq \text{Total Available Resource.} \end{aligned} \tag{4.1}$$

Utility maximization techniques can achieve certain fairness criteria by their choice of utility functions. The simplest fairness criterion is of max-min fairness. An allocation $\vec{\Lambda} = (\Lambda_r, \forall\, r)$ is max-min fair if the utility of user $r$ cannot be increased without decreasing the utility of user $r'$ where $U_{r'}(\Lambda_{r'}) \leq U_r(\Lambda_r)$.

In [56, 50], proportional fairness is applied to allocations in networks. A feasible allocation $\vec{\Lambda}$ is proportionally fair if for any other feasible allocation $\vec{\Lambda}'$,

$$\sum_r \frac{\Lambda'_r - \Lambda_r}{\Lambda_r} \leq 0.$$

This allocation is realized by assuming log utility functions in (4.1). Another fairness criterion is the $(p, \alpha)$-proportional fairness proposed in [60]. A feasible allocation $\vec{\Lambda}$ is $(p, \alpha)$-proportionally fair if for any other feasible allocation $\vec{\Lambda}'$,

$$\sum_r p_r \frac{\Lambda'_r - \Lambda_r}{\Lambda_r^\alpha} \leq 0.$$

This criterion simplifies to proportionally fair for $\alpha = 1$ and approaches max-min fairness as $\alpha \to \infty$.

Processor sharing is a sharing discipline used to allocate a resource simultaneously between multiple users. It has several advantages when applied to networks. Processor sharing leads to equal allocation of a resource and is thus fair. It also restricts users with large service requirements from hindering other users for a long time (as would be the case with the FCFS policy). The TCP congestion control mechanism results in processor sharing between several competing TCP flows with the same Round-Trip-Time (RTT). In [61], processor sharing discipline is empirically shown to have shorter mean flow completion time for TCP flows with different RTTs. For analytical results on the sojourn time distribution and the distribution of the number of jobs for an $M/G/1/\infty$ processor sharing queue, see [62].

Resource allocation and pricing, along with mechanism design are well studied problems. Mechanism design relates to the design of pricing schemes along with their implementation to induce optimal behaviour in a set of users. In the context of sharing a server in data-centers or sharing bandwidth in networks, a plethora of pricing based models exist, e.g., [63, 64]. See [65] for a survey of pricing in homogeneous and heterogeneous wireless networks and [66] for a survey of pricing mechanisms in cognitive radio networks.

One of the pricing models we analyze is the VCG auction, a generalization of the second-price auction (see [67]). In a second-price auction of a single indivisible item, the highest bidder is awarded the item but pays the second highest bid. This model is efficient in the sense that the bidder with the highest valuation of the item wins the auction. Second-price auctions have been used by governments in the sale of wireless spectrum. A generalization of the second-price auction is also used for the sale of keywords by a search engine to potential advertisers; advertisers bid to place their advertisement higher when certain keywords are searched (see [68] for an analysis). While

the model may not maximize the revenue for the operator, it is of theoretical interest since it is incentive compatible. Incentive compatibility implies that the dominant strategy of bidders is to bid their valuations truthfully (see [69]).

The VCG auction that we use is a generalization from a single indivisible item to an infinitely divisible commodity (see [70]). Consider a single auctioneer with a resource of capacity $C$ and $R$ bidders indexed by the set $\{1, \ldots, R\}$. Bidder $r$ has the utility function $U_r(\cdot)$. The bidders bid by submitting a valuation function $W_r(x)$ which may or may not be the true, privately held valuation function. Since the VCG mechanism is incentive compatible, the dominant strategy for bidders is to report their true valuation function $U_r(\cdot)$.

The first step by the auctioneer is to obtain the optimal allocation, i.e., maximize the social welfare. Given the set of all bids $\vec{W} = (W_1, \ldots, W_R)$, the optimal allocation $\vec{\Lambda}^{VCG}(\vec{W}) := (\Lambda_1^{VCG}(\vec{W}), \ldots, \Lambda_R^{VCG}(\vec{W}))$ is given by

$$\begin{aligned} \vec{\Lambda}^{VCG} &= \arg\max_{\vec{\Lambda}} \sum_{r=1}^{R} W_r(\Lambda_r) \\ \text{subject to} \quad & \Lambda_r \geq 0 \quad \text{for all } r \\ & \sum_{r=1}^{R} \Lambda_r \leq C. \end{aligned}$$

After identifying the optimal allocation, the price charged to bidder $r$ is evaluated as

$$p_r = \max_{\vec{\Lambda}} \left( \sum_{s \neq r | \Lambda_r = 0} W_s(\Lambda_s) \right) - \sum_{s \neq r} W_s(\Lambda_r^{VCG}). \tag{4.2}$$

The first summation here is the maximum social welfare when user $r$ is removed from bidding (denoted by allocating no resource to $r$). The second summation is the social welfare when bidder $r$ is present but its utility is not

included. The price charged to bidder $r$ is thus the decrease in the social welfare by it entering the auction.

Another pricing model considered in this work is congestion based pricing which employs the Lagrange shadow prices. In constrained optimization theory, the dual variables arise naturally and have the interpretation of the costs associated with hitting the constraints (see [71, 72]). The primal-dual algorithm relies on these shadow prices and has an easy, distributed implementation, e.g., [50]. In [63], the shadow prices associated with the congestion in the maximal clique of a multihop wireless network is used to maximize social welfare.

### 4.1.2 Outline of this Chapter

We present the system model in Section 4.2 and analyze the pricing models in Section 4.3. We present some simulation results in Section 4.4 and present our concluding remarks in Section 4.5.

## 4.2 System Model

We model an infinitely divisible resource as a server with $M/G$ inputs, capacity $C$, and using the processor sharing discipline. The system consists of users that represent file transfers or flows. Each user arriving to the server belongs to one of $K$ classes indexed by the set $\{1, \ldots, K\}$. A class is distinguished by its arrival and service requirement characteristics. Class $k$ user arrivals are modelled as a Poisson process with rate $\lambda_k$. Each class $k$ user brings a random amount of work, independent and identically distributed, with some general distribution with mean $\nu_k$. At an instant $t$, let $\vec{x}(t) := (x_1(t), \ldots, x_K(t))$ denote the number of users of each type present in the system with $x_k(t) \geq 0$ denoting the number of class $k$ users. The allocation to a user in the system

is $C/|\vec{x}|$ and the total allocation to class $k$ is given by $\Lambda_k(\vec{x}) = x_k C/|\vec{x}|$. Let $\vec{\Lambda}(\vec{x}) = (\Lambda_1(\vec{x}), \ldots, \Lambda_K(\vec{x}))$. As discussed in the previous section, processor sharing results from maximizing social welfare when each user has a log utility function. We will assume this utility function when we derive the payments under VCG auctions and congestion based pricing.

Define $\alpha_k := \lambda_k \nu_k$ and $\vec{\alpha} := (\alpha_1, \ldots, \alpha_K)$. The traffic intensity of class $k$ is denoted by $\rho_k = \alpha_k/C$. Let the total traffic intensity $\rho$ be given by $\rho = \sum_{k=1}^{K} \rho_k$. We use the notation $\vec{\alpha}^{\vec{x}} = \prod_{k=1}^{K} \alpha_k^{x_k}$ for convenience.

Let $\pi$ be the stationary distribution of the underlying Markov process, $\mathbb{P}_N$ be the Palm probability associated with the stationary point process $N$, and $\mathbb{E}_N$ be the expectation with respect to the Palm probability. Palm probability at arrivals is the same as the stationary measure since we have Poisson arrivals and due to the Poisson Arrivals See Time Averages (PASTA) property. With a slight abuse of notation, let $\mathbb{E}_{\vec{x}}$ be the expectation conditioned on seeing the arrival state as $\vec{x}$. Let $\Phi(\vec{x})$ be the balance function given as

$$\Phi(\vec{x}) = \frac{1}{C^{|\vec{x}|}} \left( \frac{|\vec{x}|!}{x_1! \ldots x_K!} \right) \tag{4.3}$$

for $\vec{x} \in \mathbb{Z}_+^K$ and $\Phi(\vec{x}) = 0$ otherwise. See Section 4.2.2 for a discussion on balance functions. Define $\chi(\vec{x}) := \Phi(\vec{x})\vec{\alpha}^{\vec{x}}$. $\chi(\vec{x})$ is an invariant distribution of the underlying Markov process. The stationary distribution $\pi(\vec{x})$ is then given by

$$\pi(\vec{x}) = \frac{\chi(\vec{x})}{\sum_{\vec{y}} \chi(\vec{y})}. \tag{4.4}$$

In our analysis, the arrival rate is independent of the number of users present in the system and subsequently independent of the per unit-time price. This simplifying assumption makes the notation and analysis easier. However, one can extend the model to state dependent arrival rates provided the new rates satisfy the balance property.

We describe a Quality of Service (QoS) constraint that we introduce next.

We then review key ideas from insensitive allocations followed by the Swiss Army formula.

### 4.2.1 A QoS Requirement

We will see that under VCG auctions and congestion-based pricing, the operator can collect arbitrarily large revenue per unit-time by installing small capacity, leading to longer sojourn times and greater accrued payments. To overcome this, we study a QoS requirement defined as follows. A class $k$ user pays the operator if and only if the rate allocated at time $t$, $C/|\vec{x}(t)|$, is equal to or greater than $r_k$. For ease of exposition, all $r_k$ are assumed to be identical, i.e., $r_{\min} = r_k$. The $C/|\vec{x}| \geq r_{\min}$ condition is then equivalent to a $|\vec{x}| \leq n^*$ condition where $n^* = \lfloor C/r_{\min} \rfloor$. This QoS constraint provides the motivation for the operator to offer some minimum service rates. Other variations of the QoS constraint can be easily incorporated into our framework. For example, to deter users from free riding, one can consider a fixed entry fee or charge a constant fee when the minimum rate requirement is not met.

### 4.2.2 Review of Insensitive Allocations

An allocation is said to be insensitive if $\pi(\vec{x})$ depends only on the first-order moments of the arrival process $(\lambda_k)$ and the service distribution $(\nu_k)$ (see [57]). For all $k, k'$ such that $x_k, x_{k'} > 0$, insensitive allocations satisfy the following balance property:

$$\frac{\Lambda_k(\vec{x} - \vec{e}_{k'})}{\Lambda_k(\vec{x})} = \frac{\Lambda_{k'}(\vec{x} - \vec{e}_k)}{\Lambda_{k'}(\vec{x})}.$$

In words, the fractional change in the allocation to class $k$ when a class $k'$ user is removed is the same as the fractional change in the allocation to class $k'$ when a class $k$ user is removed. The balance property is equivalent to the

existence of a balance function $\Phi$ such that,

$$\Lambda_k(\vec{x}) = \frac{\Phi(\vec{x} - \vec{e}_k)}{\Phi(\vec{x})},$$

for all $k$ such that $x_k > 0$ (see [57]). A balanced fair allocation is a maximal, insensitive allocation. In the case of a single server, a maximal allocation means that the server's capacity is fully utilized. For a network of servers, the balanced fair allocation does not necessarily coincide with either max-min fair allocations or proportionally fair allocations. In [73], it is shown that for a processor sharing network, insensitive allocations asymptotically converge to proportional fairness.

The balance function for a single, processor-sharing server is given in (4.3). For the single server case, the insensitive allocation with this balance function is maximal and coincides with both the max-min allocation and the proportionally fair allocation (see [57]). In our model, a property exhibited by the balance function is

$$\Phi(\vec{x}) = \frac{1}{C} \sum_{k=1}^{K} \Phi(\vec{x} - \vec{e}_k).$$

The system is stable if the traffic intensity is less than unity, i.e.,

$$\rho < 1.$$

Queueing networks with insensitivity were first studied by Kelly and Whittle and thus are called Whittle-Kelly networks (see [74, 75]). See [74, 76, 77, 78, 73] and the references therein for further discussions on insensitivity and processor sharing.

### 4.2.3 The Swiss Army Formula

Consider an ordered, simple point process $\{T_n\}_{n\in\mathbb{Z}}$ with $T_0 \leq 0 < T_1$ and a simple point process $\{\tau_n\}_{n\in\mathbb{Z}}$. Let $A$ and $D$ be the counting measures

associated with $\{T_n\}$ and $\{\tau_n\}$ respectively. In the context of a queueing server, $A$ and $D$ correspond to the arrival and departure processes. For each $n \in \mathbb{Z}$, we require

$$W_n := \tau_n - T_n \geq 0,$$

i.e., the sojourn time of the $n^{th}$ arrival, $W_n$, is non-negative. The number of users in the system at time $t$, $X(t)$, follows the following conservation equation.

$$X(b) - X(a) = A((a, b\,]) - D((a, b\,])$$

Define the intensity of arrivals as $\lambda_A := \mathbb{E}[A(0, 1]]$. Let $\{B(t)\}_{t\in\mathbb{R}}$ be a cádlág process and let $\{Z(t)\}_{t\in\mathbb{R}}$ be a non-negative real-valued stochastic process on the same probability triple. Then, assuming the system is ergodic, the Swiss Army formula is given as

$$\lambda_A \mathbb{E}_A \left[ \int_{(0,W_0]} Z(s) dB(s) \right] = \frac{1}{t} \mathbb{E} \left[ \int_{(0,t]} X(s_-) Z(s) dB(s) \right]$$

See [79, 80] for further discussions.

## 4.3 Analysis

We will see that the zeroth, first, and the second moments of the number of users in the system will play an important role in our analysis of revenue collected by the operator. Thus, we start by characterizing the following

terms that aid in deriving the aforementioned moments.

$$t(n) = \sum_{\vec{x}:|\vec{x}|=n} \chi(\vec{x}) \tag{4.5}$$

$$s_k(n) = \sum_{\vec{x}:|\vec{x}|=n} x_k \chi(\vec{x}) \tag{4.6}$$

$$\bar{s}_k(n) = \sum_{m>n} s_k(m) \tag{4.7}$$

$$s_{i,j}(n) = \sum_{\vec{x}:|\vec{x}|=n} x_i x_j \chi(\vec{x}) \tag{4.8}$$

After normalizing the scale invariant distribution, the expressions in (4.5), (4.6), and (4.8) respectively denote the zeroth moment, the first moment, and the second moment (if $i = j$) of the number of users in the system. We will see that other expressions of interest are manipulated into the terms above, e.g.,

$$\sum_{\vec{x}} x_k \chi(\vec{x}) = \sum_{n=0}^{\infty} s_k(n), \quad \text{and}$$
$$\mathbb{E}[|\vec{x}|^2 x_i] = \frac{1-\rho}{\Phi(\vec{0})} \sum_{n=0}^{\infty} n^2 s_i(n).$$

The following lemmas evaluate $t(n), s_k(n), \bar{s}_k(n)$, and $s_{i,j}(n)$.

**Lemma 3.** *Let $t(n)$ be defined as in (4.5). Then, $t(n) = \Phi(\vec{0})\rho^n$ and $\sum_{n=0}^{\infty} t(n) = \sum_{\vec{y}} \chi(\vec{y}) = \frac{\Phi(\vec{0})}{1-\rho}$.*

*Proof of Lemma 3.* By definition, $t(0) = \chi(\vec{0}) = \Phi(\vec{0})$.

$$\begin{aligned} t(n) &= \sum_{\vec{x}:|\vec{x}|=n} \chi(\vec{x}) \\ &= \sum_{\vec{x}:|\vec{x}|=n} \frac{1}{C} \sum_{m=1}^{K} \Phi(\vec{x} - \vec{e}_m)\vec{\alpha}^{\vec{x}} \\ &= \sum_{m=1}^{K} \rho_m \sum_{\vec{x}:|\vec{x}|=n-1} \Phi(\vec{x})\vec{\alpha}^{\vec{x}} \\ &= \rho \cdot t(n-1). \end{aligned}$$

Also,

$$\sum_{\vec{x}} \chi(\vec{x}) = \sum_{n=0}^{\infty} t(n) = \frac{\Phi(\vec{0})}{1-\rho}.$$

□

**Lemma 4.** *Let $s_k(n)$ and $\bar{s}_k(n)$ be defined as in (4.6) and (4.7). Then*

$$s_k(n) = n\rho^{n-1}\rho_k\Phi(\vec{0}),$$

*and*

$$\bar{s}_k(n) = \frac{\Phi(\vec{0})}{1-\rho}\rho^n\rho_k\left(n + \frac{1}{1-\rho}\right).$$

*Proof of Lemma 4.* We start with

$$\begin{aligned} s_k(n) &= \sum_{\vec{x}:|\vec{x}|=n} \frac{x_k}{C} \sum_{m=1}^{K} \Phi(\vec{x} - \vec{e}_m)\vec{\alpha}^{\vec{x}} \\ &= \sum_{m=1}^{K} \rho_m \left[ \sum_{\vec{x}:|\vec{x}|=n-1} x_k\chi(\vec{x}) + \sum_{\vec{x}:|\vec{x}|=n-1} (\vec{e}_m)_k\chi(\vec{x}) \right] \\ &= \sum_{m=1}^{K} \rho_m s_k(n-1) + \sum_{m=1}^{K} \rho_m \sum_{\vec{x}:|\vec{x}|=n-1} (\vec{e}_m)_k\chi(\vec{x}). \end{aligned}$$

Or,

$$\begin{aligned} s_k(n) &= \rho s_k(n-1) + \rho_k t(n-1) \\ &= \rho s_k(n-1) + \rho_k \rho^{n-1} \Phi(\vec{0}). \end{aligned} \tag{4.9}$$

It is easily shown that $s_k(n) = n\rho^{n-1}\rho_k\Phi(\vec{0})$ is the solution to the recursion in (4.9). The first part of the result follows. Next,

$$\begin{aligned} \bar{s}_k(n) &= \rho_k \Phi(\vec{0}) \sum_{m>n} m\rho^m \\ &= \rho_k \Phi(\vec{0}) \frac{\rho^n}{1-\rho} \left( n + \frac{1}{1-\rho} \right). \end{aligned}$$

□

**Lemma 5.** *Let $s_{i,j}(n)$ be defined as in (4.8). Then,*

$$s_{i,j}(n) = \begin{cases} n(n-1)\rho_i\rho_j\rho^{n-2}\Phi(\vec{0}) & \text{if } i \neq j \\ n\left((n-1)\rho_i^2 + \rho_i\rho\right)\rho^{n-2}\Phi(\vec{0}) & \text{if } i = j. \end{cases}$$

*Proof of Lemma 5.* The proof relies on establishing a recursive expression for $s_{i,j}(n)$. The above is the solution of the recursion.

$$\begin{aligned} s_{i,j}(n) &= \sum_{m=1}^{K} \frac{\alpha_m}{C} \sum_{\vec{x}:|\vec{x}|=n} x_i x_j \Phi(\vec{x} - \vec{e}_m) \vec{\alpha}^{\vec{x}-\vec{e}_m} \\ &= \sum_{m=1}^{K} \rho_m \sum_{\vec{y}:|\vec{y}|=n-1} (\vec{y} + \vec{e}_m)_i (\vec{y} + \vec{e}_m)_j \Phi(\vec{y}) \vec{\alpha}^{\vec{y}} \\ &= \rho s_{i,j}(n-1) + \rho_j s_i(n-1) + \rho_i s_j(n-1) + \mathbf{1}_{(i=j)}\rho_i t(n-1). \end{aligned}$$

Note that $s_{i,j}(0) = 0$ for any $i, j$ and that $s_{i,j}(1) = 0$ if $i \neq j$. □

The above three lemmas compute the zeroth, the first, and the second moment of the number of users in the system under the invariant distribution $\chi(\vec{x})$. Note that Lemma 3 gives the expression for the normalizing term

$(\sum_{\vec{x}} \chi(\vec{x}))^{-1}$ in (4.4) for obtaining the stationary distribution from the invariant distribution. We also define the following terms which are needed for evaluating prices.

$$u(n) := \sum_{\vec{x}:|\vec{x}|>n} (|\vec{x}| - 1/2)\, \pi(\vec{x}) \tag{4.10}$$

$$v(n) := \sum_{\vec{x}:|\vec{x}|=n} |\vec{x}|^2 \chi(\vec{x}) = n^2 t(n) \tag{4.11}$$

$$g_k(n) := \sum_{\vec{x}:|\vec{x}|=n} \frac{x_k}{|\vec{x}|} \chi(\vec{x}) = s_k(n)/n \tag{4.12}$$

**Proposition 4.** *Let $u(n)$ be defined as in* (4.10)*. Then,*

$$u(n) = \rho^{n+1} \left( n + \frac{1}{1-\rho} - \frac{1}{2} \right).$$

*Proof of Proposition 4.*

$$\begin{aligned} u(n) &= \sum_{\vec{x}:|\vec{x}|>n} \left( |\vec{x}| - \frac{1}{2} \right) \pi(\vec{x}) \\ &= \sum_{\vec{x}:|\vec{x}|>n} |\vec{x}| \pi(\vec{x}) - \frac{1}{2} \sum_{\vec{x}:|\vec{x}|>n} \pi(\vec{x}) \\ &= \sum_{\vec{x}:|\vec{x}|>n} (x_1 + \cdots + x_K) \pi(\vec{0}) \chi(\vec{x}) - \frac{1}{2} \sum_{\vec{x}:|\vec{x}|>n} \pi(\vec{0}) \chi(\vec{x}). \end{aligned}$$

Using $\pi(\vec{0}) = (\sum_{\vec{x}} \chi(\vec{x}))^{-1} = (\sum_{n\geq 0} t(n))^{-1}$,

$$u(n) = \frac{\sum_{k=1}^{K} \bar{s}_k(n)}{\sum_{m\geq 0} t(m)} - \frac{\sum_{m>n} t(m)}{2 \sum_{m\geq 0} t(m)}.$$

Using Lemma 3 and Lemma 4, we get

$$u(n) = \rho^{n+1} \left( n + \frac{1}{1-\rho} \right) - \frac{\rho^{n+1}}{2}.$$

□

This proposition is useful for evaluating revenue in VCG auctions.

### 4.3.1 The Pricing Models

We implicitly assume that the QoS requirement is always enforced unless stated otherwise. The first pricing model is the fixed rate pricing where the user pays a fixed per unit-time, per unit-resource price of $\beta$, i.e., if a user is allocated $\Lambda$ resource for time $T$, the user pays $\beta\Lambda T$. Thus, the per unit-time price is

$$c_k^F(\vec{x}) = \begin{cases} \frac{\beta C}{|\vec{x}|} & \text{if } 1 \leq |\vec{x}| \leq n^* \\ 0 & \text{otherwise.} \end{cases} \tag{4.13}$$

The operator's revenue is the aggregate of user payments. The per unit-time revenue is

$$R_F(\vec{x}) = \begin{cases} |\vec{x}|\frac{\beta C}{|\vec{x}|} = \beta C & \text{if } 1 \leq |\vec{x}| \leq n^* \\ 0 & \text{otherwise.} \end{cases}$$

The log utility assumption is important for the next two pricing models. Under the VCG auction, a user pays the decrease in maximum social welfare caused by it entering the system. Let $r$ index over the set of users and with a slight abuse of notation, let $\Lambda_r$ indicate the allocation to user $r$. If $|\vec{x}| \geq 2$, this price is calculated as

$$\begin{aligned} c_k^V(\vec{x}) &= \max_{\vec{\Lambda}} \sum_{s \neq r | \Lambda_r = 0} \log(\Lambda_s) - \sum_{s \neq r} \log(\Lambda_s^{PS}) \\ &= (|\vec{x}| - 1) \log \frac{C}{|\vec{x}| - 1} - (|\vec{x}| - 1) \log \frac{C}{|\vec{x}|} \\ &= (|\vec{x}| - 1) \log \frac{|\vec{x}|}{|\vec{x}| - 1}, \end{aligned}$$

where, $\Lambda_s^{PS}$ is the allocation to user $s$ under processor sharing, i.e., $\Lambda_s^{PS} = C/|\vec{x}|$. The aggregate per unit-time revenue collected by the operator is given by

$$R_V(\vec{x}) = \begin{cases} |\vec{x}|(|\vec{x}| - 1) \log \frac{|\vec{x}|}{|\vec{x}|-1} & \text{if } 2 \leq |\vec{x}| \leq n^* \\ 0 & \text{otherwise.} \end{cases}$$

To gain further insights in the revenue and payment problem, the following approximation is shown to hold.

**Proposition 5.** *$R_V(\vec{x}) \approx |\vec{x}| - \frac{1}{2}$ and the approximation error is $O(1/|\vec{x}|)$. $|\vec{x}| - \frac{1}{2}$ is an upper bound on $R_V(\vec{x})$.*

*Proof of Proposition 5.*

$$\begin{aligned} R_V(\vec{x}) &= |\vec{x}|(|\vec{x}|-1)\log\left(\frac{|\vec{x}|}{|\vec{x}|-1}\right) \\ &= |\vec{x}|(|\vec{x}|-1)\sum_{n=1}^{\infty}(-1)^{n+1}\frac{1}{n}\left(\frac{1}{|\vec{x}|-1}\right)^n \end{aligned}$$

Subtract $(|\vec{x}| - 1/2)$ from both sides.

$$\begin{aligned} R_V(\vec{x}) - \left(|\vec{x}| - \frac{1}{2}\right) &= \frac{1}{2} - \frac{|\vec{x}|}{2(|\vec{x}|-1)} + \frac{|\vec{x}|}{3(|\vec{x}|-1)^2} - \dots \\ &= -\frac{1}{2(|\vec{x}|-1)} + \sum_{m=3}^{\infty}\frac{(-1)^{m-1}|\vec{x}|}{m(|\vec{x}|-1)^{m-1}} \end{aligned}$$

This proves the $O(1/|\vec{x}|)$ approximation. To show $(|\vec{x}| - 1/2)$ is an upper bound, rewrite the $m$-th term above as

$$\frac{|\vec{x}|}{m(|\vec{x}|-1)^{m-1}} = \frac{1}{m(|\vec{x}|-1)^{m-2}} + \frac{1}{m(|\vec{x}|-1)^{m-1}}.$$

Substitution and simplifying gives

$$\begin{aligned} R_V - \left(|\vec{x}| - \frac{1}{2}\right) &= \sum_{m=1}^{\infty}\frac{(-1)^m}{(m+1)(m+2)(|\vec{x}|-1)^m} \\ &= \sum_{m=1,3,\dots}\frac{(-1)^m}{(m+2)(|\vec{x}|-1)^m}\left[\frac{1}{m+1} - \frac{1}{(m+3)(|\vec{x}|-1)}\right] \\ &< 0 \end{aligned}$$

as

$$\frac{1}{m+1} - \frac{1}{(m+3)(|\vec{x}|-1)} > 0.$$

Thus, $R_V - (|\vec{x}| - 1/2) < 0$. □

The above approximation for VCG revenue is used throughout this work. The per unit-time price paid by the user under this approximation is given by

$$c_k^V(\vec{x}) = \begin{cases} \left(1 - \frac{1}{2|\vec{x}|}\right) & \text{if } 2 \leq |\vec{x}| \leq n^* \\ 0 & \text{otherwise,} \end{cases} \tag{4.14}$$

and the per unit-time revenue earned by the operator is

$$R_V(\vec{x}) = \begin{cases} |\vec{x}| - 1/2 & \text{if } 2 \leq |\vec{x}| \leq n^* \\ 0 & \text{otherwise.} \end{cases}$$

In congestion-based pricing, the Lagrange shadow price or the dual variable in the social welfare maximization problem is charged. This shadow price has the advantage of leading the system to the social optima in a distributed implementation. The shadow price under processor sharing is $|\vec{x}|/C$. Thus, the payment per unit-time made by a class $k$ user is given by

$$c_k^L(\vec{x}) = \begin{cases} \frac{|\vec{x}|}{C} & \text{if } 1 \leq |\vec{x}| \leq n^* \\ 0 & \text{otherwise,} \end{cases} \tag{4.15}$$

and the aggregate per unit-time revenue collected by the operator is

$$R_L(\vec{x}) = \begin{cases} \frac{|\vec{x}|^2}{C} & \text{if } 1 \leq |\vec{x}| \leq n^* \\ 0 & \text{otherwise.} \end{cases}$$

Fundamentally, the per unit-time price charged to users under fixed rate pricing, VCG auctions, and the congestion-based pricing are proportional to $1/|\vec{x}|$, to $\approx 1$, and to $|\vec{x}|$. The consequence is that a user is charged less (offered a discounted per unit-time price) at high loads under fixed rate pricing, offered a constant price under VCG auctions, and is charged more under congestion-based pricing (high demand implies the resource is precious and thus the price goes up).

### 4.3.2 Post-payments vs. Pre-payments

All three pricing models discussed in our work charge a user based on the number of users in the system. A change in the number of users is reflected in the instantaneous per unit-time price. For a tagged user, the exact charge accrued is evaluated by tracking arrivals and departures during the user's sojourn. We derive the mean of this exact payment incurred by a user using sample path arguments from Palm probability. The mean of the operator's revenue is independently derived. Since the total charge accrued is only known at the end of sojourn, we refer to such an implementation as the *post-payment scheme*.

After deriving the mean revenue and post-payment expressions, we will investigate a *pre-payment scheme* where a user is charged a fee upfront based on the system load (the number of users present in the system) on arrival and the expected sojourn time observed. Prices are adjusted to ensure the same mean payment for each class as in the post-payment scheme. A pre-payment scheme has several benefits. First, the user is aware of the payment upfront unlike the post-payment scheme where a user may be billed a large fee caused by sudden high loads during sojourn. Second, the second moment (and thus the standard deviation) of user payments in the pre-payment scheme can be exactly characterized.

### 4.3.3 Mean Operator Revenue

The mean operator revenue is derived under the three pricing models. The expressions hold under both the pre-payment and the post-payment scheme since the mean payment by each class remains the same under both schemes. The revenues are per unit-time.

**Proposition 6.** *The operator's revenue per unit-time under under the three*

*pricing models is given by*

$$\bar{R}_F = \beta C \rho (1 - \rho^{n^*}) \tag{4.16}$$

$$\bar{R}_V = \frac{\rho^2}{2}\left(1 + \frac{2}{1-\rho}\right) - \rho^{n^*+1}\left(n^* - \frac{1}{2} + \frac{1}{1-\rho}\right) \tag{4.17}$$

$$\bar{R}_L = \frac{1-\rho}{C}\sum_{n=1}^{n^*} n^2 \rho^n. \tag{4.18}$$

*Proof of Proposition 6.* The mean revenue under fixed rate pricing is given by

$$\begin{aligned}
\bar{R}_F &= \sum_{\vec{x}:1\le|\vec{x}|\le n^*} \beta C \pi(\vec{x}) \\
&= \beta C \frac{1}{\sum_{\vec{x}} \chi(\vec{x})} \sum_{\vec{x}:1\le|\vec{x}|\le n^*} \chi(\vec{x}) \\
&= \beta C \frac{\sum_{n=1}^{n^*} t(n)}{\sum_{n=0}^{\infty} t(n)}.
\end{aligned}$$

Using Lemma 3,

$$\bar{R}_F = \beta C \frac{1-\rho}{\Phi(\vec{0})} \frac{\Phi(\vec{0})(\rho - \rho^{n^*+1})}{1-\rho} = \beta C \rho (1 - \rho^{n^*}).$$

The mean revenue under VCG auctions is

$$\begin{aligned}
\bar{R}_V &= \sum_{\vec{x}:2\le|\vec{x}|\le n^*} \left(|\vec{x}| - \frac{1}{2}\right) \pi(\vec{x}) \\
&= u(1) - u(n^*).
\end{aligned}$$

The result for $\bar{R}_V$ follows by simplification. The mean revenue under congestion-

based pricing is given by

$$\begin{aligned}\bar{R}_L &= \sum_{\vec{x}:1\leq|\vec{x}|\leq n^*} \frac{|\vec{x}|^2}{C}\pi(\vec{x}) \\ &= \frac{\pi(\vec{0})}{C} \sum_{\vec{x}:1\leq|\vec{x}|\leq n^*} |\vec{x}|^2\chi(\vec{x}) \\ &= \frac{1}{C}\frac{\sum_{1\leq n\leq n^*} v(n)}{\sum_{n\geq 0} t(n)}.\end{aligned}$$

Using $v(n) = n^2 t(n)$ and Lemma 3,

$$\bar{R}_L = \frac{1-\rho}{C}\sum_{n=1}^{n^*} n^2\rho^n.$$

The following identity (for $\rho < 1$) simplifies the summation.

$$\sum_{n=1}^{m} n^2\rho^n = \frac{\rho}{(1-\rho)^3}[1+\rho-\rho^m(m^2\rho^2-(2m^2+2m-1)\rho+(m+1)^2)]$$

□

The proof highlights that the mean revenue earned by the operator for fixed rate pricing, VCG auctions, and congestion-based pricing is respectively related to the zeroth, first, and the second moment of the total number of users in the system, i.e.,

$$\begin{aligned}\bar{R}_F &\propto \mathbb{E}[|\vec{x}|^0\mathbf{1}_{(1\leq|\vec{x}|\leq n^*)}] \\ \bar{R}_V &\propto \mathbb{E}[(|\vec{x}|-1/2)\mathbf{1}_{(2\leq|\vec{x}|\leq n^*)}] \\ \bar{R}_L &\propto \mathbb{E}[|\vec{x}|^2\mathbf{1}_{(1\leq|\vec{x}|\leq n^*)}].\end{aligned}$$

This key insight is attributed to the inherent structure of the pricing models identified in Section 4.3.1. The mean per unit-time revenue of the operator without the QoS constraint is obtained next.

**Corollary 1.** *Let the operator's revenue without the QoS be denoted by* $\hat{R}_F, \hat{R}_V$, *and* $\hat{R}_L$ *under the three pricing models. Then,*

$$\hat{R}_F = \beta C \rho$$
$$\hat{R}_V = \frac{\rho^2}{2}\left(\frac{3-\rho}{1-\rho}\right)$$
$$\hat{R}_L = \frac{\rho(1+\rho)}{C(1-\rho^2)}.$$

The above is an immediate consequence by taking the limit $n^* \to \infty$. We observe that the operator's revenue becomes arbitrarily large as $\rho \to 1$ for VCG auctions and for the Lagrange pricing model.

### 4.3.4 Post-payments: Exact Charge Accrued by Users

We present the mean of payments made by class $k$ users next.

**Proposition 7.** *The mean payment by class k users under the three pricing models are given by*

$$\bar{c}_k^F = \nu_k \beta (1-\rho^{n^*})$$
$$\bar{c}_k^V = \frac{\nu_k}{C}\left(\rho\left(\frac{1-\rho^{n^*}}{1-\rho}\right) + \frac{\rho}{2} - \left(n^* - \frac{1}{2}\right)\rho^{n^*}\right)$$
$$\bar{c}_k^L = \frac{\nu_k(1-\rho)}{C^2}\sum_{n=1}^{n^*} n^2 \rho^{n-1}.$$

*Proof of Proposition 7.* Let $A_k$ be the simple point process marking the arrivals of class $k$ users and $W_0^k$ be the random variable denoting the sojourn time of the arrival at time 0. The mean of payments by class $k$ users under fixed rate pricing is given by

$$\bar{c}_k^F = \mathbb{E}_{A_k}\left[\int_0^{W_0^k} \frac{\beta C}{|\vec{x}(t)|}\mathbf{1}_{(1\le|\vec{x}(t)|\le n^*)}dt\right].$$

Applying the Swiss Army formula,

$$\begin{aligned}
\bar{c}_k^F &= \frac{1}{\lambda_k}\frac{1}{t}\mathbb{E}\left[\int_0^t \frac{x_k(s_-)\beta C}{|\vec{x}(s)|}\mathbf{1}_{(1\le|\vec{x}(s)|\le n^*)}ds\right] \\
&= \frac{\beta C}{\lambda_k}\mathbb{E}\left[\frac{x_k}{|\vec{x}|}\mathbf{1}_{(1\le|\vec{x}|\le n^*)}\right] \\
&= \frac{\beta C}{\lambda_k}\sum_{n=1}^{n^*}\sum_{\vec{x}:|\vec{x}|=n}\frac{x_k}{n}\pi(\vec{x}) \\
&= \frac{\beta C}{\lambda_k}\frac{1}{\sum_{n=0}^{\infty}t(n)}\sum_{n=1}^{n^*}g_k(n) \\
&= \nu_k\beta(1-\rho^{n^*}).
\end{aligned}$$

Similarly, the mean of payments under the VCG auction by class $k$ users is given by

$$\bar{c}_k^V = \mathbb{E}_{A_k}\left[\int_0^{W_0^k}\left(1-\frac{1}{2|\vec{x}(t)|}\right)\mathbf{1}_{(2\le|\vec{x}(t)|\le n^*)}dt\right].$$

Again, using the Swiss Army formula,

$$\begin{aligned}
\bar{c}_k^V &= \frac{1}{\lambda_k}\mathbb{E}\left[\left(1-\frac{1}{2|\vec{x}|}\right)\mathbf{1}_{(2\le|\vec{x}|\le n^*)}\right] \\
&= \frac{1}{\lambda_k}\mathbb{E}[x_k\mathbf{1}_{(2\le|\vec{x}|\le n^*)}] - \frac{1}{\lambda_k}\mathbb{E}\left[\frac{x_k}{2|\vec{x}|}\mathbf{1}_{(2\le|\vec{x}|\le n^*)}\right]. \qquad (4.19)
\end{aligned}$$

Let $J_1$ and $J_2$ be the first and the second term respectively in (4.19). Then,

$$\begin{aligned}
J_1 &= \frac{1}{\lambda_k}\sum_{n=2}^{n^*}\sum_{\vec{x}:|\vec{x}|=n}x_k\pi(\vec{x}) \\
&= \frac{1-\rho}{\lambda_k\Phi(\vec{0})}\sum_{n=2}^{n^*}s_k(n) \\
&= \frac{\nu_k(1-\rho)}{C}\sum_{n=2}^{n^*}n\rho^{n^*-1},
\end{aligned}$$

and,

$$\begin{aligned} J_2 &= \frac{1}{2\lambda_k} \sum_{n=2}^{n^*} \sum_{\vec{x}:|\vec{x}|=n} \frac{x_k}{|\vec{x}|} \pi(\vec{x}) \\ &= \frac{1-\rho}{2\Phi(\vec{0})\lambda_k} \sum_{n=2}^{n^*} g_k(n) \\ &= \frac{\nu_k \rho}{2C}(1-\rho^{n^*-1}). \end{aligned}$$

Using the identity

$$\sum_{n=1}^{m} n\rho^{n-1} = \frac{1-\rho^{m+1}-(m+1)(1-\rho)\rho^m}{(1-\rho)^2},$$

and simplifying provides the required result. For congestion-based pricing, the mean of payments by class $k$ users is given by

$$\bar{c}_k^L = \mathbf{E}_{A_k}\left[\int_0^{W_0^k} \frac{|\vec{x}(t)|}{C} \mathbf{1}_{(1\le|\vec{x}(t)|\le n^*)} dt\right].$$

Applying the Swiss Army formula gives,

$$\begin{aligned} \bar{c}_k^L &= \frac{1}{\lambda_k C} \mathbb{E}[x_k |\vec{x}| \mathbf{1}_{(1\le|\vec{x}|\le n^*)}] \\ &= \frac{1}{\lambda_k C} \sum_{n=1}^{n^*} \sum_{\vec{x}:|\vec{x}|=n} x_k n \pi(\vec{x}) \\ &= \frac{1-\rho}{\lambda_k C \Phi(\vec{0})} \sum_{n=1}^{n^*} n s_k(n) \\ &= \frac{\nu_k(1-\rho)}{C^2} \sum_{n=1}^{n^*} n^2 \rho^{n-1}, \end{aligned}$$

which shows the required result. □

Note that the metering required by the operator for each user in the system is at the time-scale at which users enter and leave the system. We provide the mean of payments by class $k$ users under the three pricing models without the QoS constraint next.

**Corollary 2.** *Let $\hat{c}_k^F, \hat{c}_k^V$, and $\hat{c}_k^L$ be the mean of payments by class $k$ users under the three pricing models when the QoS constraint is not enforced. Then,*

$$\hat{c}_k^F = \beta\nu_k$$
$$\hat{c}_k^V = \frac{\nu_k\rho}{2C}\left(\frac{3-\rho}{1-\rho}\right)$$
$$\hat{c}_k^L = \frac{\nu_k(1+\rho)}{C^2(1-\rho)^2}.$$

We observe that the mean of payments by users can become arbitrarily large as $\rho \to 1$ under VCG auctions and congestion based pricing models. This supports the QoS constraint that we impose.

**Remark 1.** *The mean operator revenue per unit-time and mean user payments are consistent and satisfy a conservation-of-money principle in means. The aggregate of mean payment per unit-time by class $k$ users is given by $\mathbb{E}[x_k]\frac{\bar{c}_k^X}{\mathbb{E}[W_k]} = \lambda_k\bar{c}_k^X$ (by Little's Law). It is easily verified that $\bar{R}_X = \sum_{k=1}^K \lambda_k\bar{c}_k^X$.*

### 4.3.5 Pre-payments: Freezing Prices on Arrival

In this section, a pre-payment mechanism is devised where the user is charged a price upfront on arrival. The price depends on the underlying pricing ideology, i.e., per unit-time price behaves similar to fixed rate pricing, VCG auctions, or congestion-based pricing. However, the price charged to a given user now only depends on the expected sojourn time at arrival and on the number of users in the system *when that user arrives* instead of the varying number of users during sojourn. The payments are adjusted such that the mean of payments by class $k$ users remain the same as under post-payments (see Section 4.3.4).

Let $W_k$ be the random variable denoting the sojourn time of a class $k$ arrival. Under the pricing mechanism $X$, where $X$ is a placeholder for $F$, $V$, or $L$, let $\gamma_k^X(\vec{x})$ be the per unit-time price fixed on class $k$ user's arrival when

the arrival observes the system state as $\vec{x}$. Under the pre-payment scheme, the price charged to any class $k$ user is given by

$$p_k^X(\vec{x}) = \gamma_k^X(\vec{x} + \vec{e}_k)\mathbb{E}_{\vec{x}}[W_k]. \tag{4.20}$$

**Proposition 8.** *The mean sojourn time for a class $k$ user conditioned on the starting state $\vec{x}$ is a linear functional of the number of users of each class $x_k$, i.e., for a processor sharing server with multiclass $M/G$ inputs,*

$$\mathbb{E}_{\vec{x}}[W_k] = A_{k,0} + \sum_{m=1}^{K} A_{k,m}x_m.$$

*Proof of Proposition 8.* The proposition is an immediate consequence of [81, Theorem 6]. The argument developed is as follows.

In [81, Theorem 2], a discrete-time round robin approximation of the discriminatory processor sharing (DPS) discipline is considered. It is shown that the mean sojourn time of a new tagged arrival (call it user $T$) can be decomposed into the sum of $|\vec{x}|+1$ terms. One of the terms is the contribution of the $T$ and the future users arriving during $T$'s servicing (in the round robin discipline). This term is $A_{k,0}$. The remaining terms are the contributions of each user (and all future users arriving during their servicing) present at $T$'s arrival.

The rest of [81] shows that the round robin approximation converges almost surely to the continuous-time DPS system. In [81, Theorem 6], it is shown that a similar decomposition holds for the continuous-time DPS system. □

Based on the structure of fixed rate pricing, VCG auctions, and congestion-based pricing, we define $\gamma_k^X(\vec{x})$ as

$$\begin{aligned}\gamma_k^F(\vec{x}) &= \sigma_k^F|\vec{x}|^{-1}\\ \gamma_k^V(\vec{x}) &= \sigma_k^V\\ \gamma_k^L(\vec{x}) &= \sigma_k^L|\vec{x}|.\end{aligned}$$

Note that $\gamma_k^X(\vec{x})$ is not zero for $|\vec{x}| > n^*$. Otherwise, arrivals observing the state in $|\vec{x}| > n^*$ may free ride the system. The constants $\sigma_k^F$, $\sigma_k^V$, and $\sigma_k^L$ are determined by equating the mean payments by class $k$ users to the payments in Section 4.3.4, i.e.,

$$\mathbb{E}[p_k^X(\vec{x})] = \bar{c}_k^X.$$

**Proposition 9.** *The constants $\sigma_k^F, \sigma_k^V$, and $\sigma_k^L$ are*

$$\sigma_k^F = \frac{\nu_k \beta (1-\rho^{n^*})}{(1-\rho)} \left( \frac{A_{k,0}}{\rho} \log \frac{1}{1-\rho} + \frac{1}{\rho^2} \left( \frac{\rho}{1-\rho} - \log \frac{1}{1-\rho} \right) \sum_{m=1}^{K} A_{k,m} \rho_m \right)^{-1}$$

$$\sigma_k^V = \rho(1-\rho^{n^*}) + \frac{\rho(1-\rho)}{2} - (1-\rho)\left(n^* - \frac{1}{2}\right)\rho^{n^*}$$

$$\sigma_k^L = \frac{\nu_k (1-\rho)^2}{C^2} \frac{\sum_{n=1}^{n^*} n^2 \rho^{n-1}}{A_{k,0} + \frac{2}{1-\rho} \sum_{m=1}^{K} A_{k,m} \rho_m}.$$

*Proof of Proposition 9.* Suppose a class $k$ arrival sees the system state as $\vec{x}$ on arrival. The fixed rate, pre-payment price charged is

$$p_k^F(\vec{x}) = \frac{\sigma_k^F}{|\vec{x} + \vec{e}_k|} \left( A_{k,0} + \sum_{m=1}^{K} A_{k,m} x_m \right).$$

It is required that the mean payment by a class $k$ user equal $\bar{c}_k^F$, i.e.,

$$\mathbb{E}[p_k^F(\vec{x})] = \bar{c}_k^F. \tag{4.21}$$

Starting with the left hand side (LHS) of (4.21),

$$\begin{aligned} LHS &= \sum_{\vec{x}} \frac{\sigma_k^F}{|\vec{x} + \vec{e}_k|} \left( A_{k,0} + \sum_{m=1}^{K} A_{k,m} x_m \right) \pi(\vec{x}) \\ &= \sigma_k^F (1-\rho) \left[ \frac{A_{k,0}}{\rho} \log \frac{1}{1-\rho} + \left( \frac{\frac{\rho}{1-\rho} - \log \frac{1}{1-\rho}}{\rho^2} \right) \sum_{m=1}^{K} A_{k,m} \rho_m \right] \end{aligned}$$

Equating this to $\bar{c}_k^F$ gives $\sigma_k^F$. Similarly, under VCG auctions,

$$p_k^V(\vec{x}) = \sigma_k^V \left( A_{k,0} + \sum_{m=1}^{K} A_{k,m} x_m \right),$$

and

$$\begin{aligned}
\mathbb{E}[p_k^V(\vec{x})] &= \sigma_k^V \sum_{\vec{x}} \left( A_{k,0} + \sum_{m=1}^{K} A_{k,m} x_m \right) \pi(\vec{x}) \\
&= \sigma_k^V \mathbb{E}[W_k] \\
&= \sigma_k^V \frac{\nu_k}{C(1-\rho)}.
\end{aligned}$$

Equating this to $\bar{c}_k^V$ gives $\sigma_k^V$. Last, under congestion-based pricing,

$$p_k^L(\vec{x}) = \sigma_k^L |\vec{x} + \vec{e}_k| \left( A_{k,0} + \sum_{m=1}^{K} A_{k,m} x_m \right).$$

Taking expectation gives

$$\begin{aligned}
\mathbb{E}[p_k^L(\vec{x})] &= \sigma_k^L \sum_{\vec{x}} |\vec{x} + \vec{e}_k| \left( A_{k,0} + \sum_{m=1}^{K} A_{k,m} x_m \right) \pi(\vec{x}) \\
&= \frac{\sigma_k^L}{1-\rho} \left[ A_{k,0} + \frac{2}{1-\rho} \sum_{m=1}^{K} A_{k,m} \rho_m \right],
\end{aligned}$$

equating which to $\bar{c}_k^L$ gives $\sigma_k^L$. □

Since the mean payment by class $k$ users remain the same as under Section 4.3.4, the mean revenue collected by the operator is unaffected and is the same as in Section 4.3.3. The evaluation of the moments of number of users earlier allows the explicit characterization of the second moment of class $k$ payments. Define $\mathrm{Li}_2(\rho) := \sum_{n=1}^{\infty} \frac{\rho^n}{n^2}$.

**Proposition 10.** *The second moment of pre-payments by class $k$ users is given by*

$$\begin{aligned}
\mathbb{E}[(p_k^F(\vec{x}))^2] &= (\sigma_k^F)^2(1-\rho)A_{k,0}^2\frac{\mathrm{Li}_2(\rho)}{\rho} \\
&+ (\sigma_k^F)^2(1-\rho)\left(\frac{\rho+3\log(1-\rho)-3\rho\log(1-\rho)}{\rho^3(1-\rho)}+\frac{2\,\mathrm{Li}_2(\rho)}{\rho^3}\right)\sum_{m=1}^K A_{k,m}^2\rho_m^2 \\
&+ (\sigma_k^F)^2(1-\rho)(-\mathrm{Li}_2(\rho)-\log(1-\rho))\sum_{m=1}^K\frac{A_{k,m}^2\rho_m}{\rho^2} \\
&+ \frac{2A_{k,0}(\sigma_k^F)^2(1-\rho)}{\rho^2}(-\mathrm{Li}_2(\rho)-\log(1-\rho))\sum_{m=1}^K A_{k,m}\rho_m \\
&+ \frac{(\sigma_k^F)^2}{\rho^3}\left[\rho+3(1-\rho)\log(1-\rho)+2(1-\rho)\,\mathrm{Li}_2(\rho)\right]\sum_{i=1}^K\sum_{j=1,j\neq i}^K A_{k,i}A_{k,j}\rho_i\rho_j
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}[(p_k^V(\vec{x}))^2] &= (\sigma_k^V)^2A_{k,0}^2+\frac{2(\sigma_k^V)^2}{(1-\rho)^2}\sum_{m=1}^K A_{k,m}^2\rho_m^2+\frac{2(\sigma_k^V)^2}{(1-\rho)^2}\sum_{i=1}^K\sum_{j=1,j\neq i}^K A_{k,i}A_{k,j}\rho_i\rho_j \\
&+\frac{(\sigma_k^V)^2}{(1-\rho)}\sum_{m=1}^K A_{k,m}^2\rho_m+\frac{2(\sigma_k^V)^2}{(1-\rho)}A_{k,0}\sum_{m=1}^K A_{k,m}\rho_m
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}[(p_k^L(\vec{x}))^2] &= \frac{(\sigma_k^L)^2A_{k,0}^2(1+\rho)}{(1-\rho)^2}+(\sigma_k^L)^2\sum_{m=1}^K A_{k,m}^2\frac{2\rho_m(2+9\rho_m+3\rho_m\rho-\rho-\rho^2)}{(1-\rho)^4} \\
&+2A_{k,0}(\sigma_k^L)^2\frac{2\rho+4}{(1-\rho)^3}\sum_{m=1}^K A_{k,m}\rho_m+\frac{(\sigma_k^L)^26(3+\rho)}{(1-\rho)^4}\sum_{i=1}^K\sum_{j=1,j\neq i}^K A_{k,i}A_{k,j}\rho_i\rho_j
\end{aligned}$$

*Proof of Proposition 10.* The steps for deriving the second moment of pre-

payments under fixed rate pricing mechanism are outlined here.

$$\begin{aligned}\mathbb{E}[(p_k^F(x))^2] &= \sum_{\vec{x}} \frac{(\sigma_k^F)^2}{|\vec{x}+\vec{e}_k|^2}\left(A_{k,0}+\sum_{m=1}^{K} A_{k,m}x_m\right)^2 \pi(\vec{x}) \\ &= \frac{(\sigma_k^F)^2(1-\rho)}{\Phi(\vec{0})}\sum_{\vec{x}}\frac{\chi(\vec{x})}{|\vec{x}+\vec{e}_k|^2}\left(A_{k,0}^2+\sum_{m=1}^{K}A_{k,m}^2x_m^2\right. \\ &\quad \left.+2A_{k,0}\sum_{m=1}^{K}A_{k,m}x_m+\sum_{i=1}^{K}\sum_{j\neq i}A_{k,i}A_{k,j}x_ix_j\right) \end{aligned} \tag{4.22}$$

Define $S_1^F, S_2^F, S_3^F$, and $S_4^F$ as the following.

$$\begin{aligned} S_1^F &:= \frac{(\sigma_k^F)^2(1-\rho)}{\Phi(\vec{0})}A_{k,0}^2\sum_{\vec{x}}\frac{\chi(\vec{x})}{(|\vec{x}|+1)^2} \\ &= (\sigma_k^F)^2(1-\rho)\frac{A_{k,0}^2}{\rho}\sum_{n=1}^{\infty}\frac{\rho^n}{n^2} \end{aligned}$$

$$\begin{aligned} S_2^F &:= \frac{(\sigma_k^F)^2(1-\rho)}{\Phi(\vec{0})}\sum_{m=1}^{K}A_{k,m}^2\sum_{\vec{x}}\frac{x_m^2\chi(\vec{x})}{(|\vec{x}|+1)^2} \\ &= (\sigma_k^F)^2(1-\rho)\sum_{m=1}^{K}A_{k,m}^2\sum_{n=0}^{\infty}\frac{n(n-1)\rho_m^2\rho^{n-2}+n\rho_m\rho^{n-1}}{(n+1)^2} \\ &= (\sigma_k^F)^2(1-\rho)\sum_{m=1}^{K}A_{k,m}^2\left[\frac{\rho_m}{\rho^2}(-\operatorname{Li}_2(\rho)-\log(1-\rho))\right. \\ &\quad \left.+\rho_m^2\left(\frac{2\operatorname{Li}_2(\rho)}{\rho^3}+\frac{\rho+3\log(1-\rho)-3\rho\log(1-\rho)}{\rho^3(1-\rho)}\right)\right] \end{aligned}$$

$$\begin{aligned} S_3^F &:= \frac{(2A_{k,0}\sigma_k^F)^2(1-\rho)}{\Phi(\vec{0})}\sum_{m=1}^{K}A_{k,m}\sum_{\vec{x}}\frac{x_m\chi(\vec{x})}{(|\vec{x}|+1)^2} \\ &= \frac{2A_{k,0}(\sigma_k^F)^2(1-\rho)}{\Phi(\vec{0})}\sum_{m=1}^{K}A_{k,m}\sum_{n=0}^{\infty}\frac{s_m(n)}{(n+1)^2} \\ &= \frac{2A_{k,0}(\sigma_k^F)^2(1-\rho)(-\operatorname{Li}_2(\rho)-\log(1-\rho))}{\rho^2}\sum_{m=1}^{K}A_{k,m}\rho_m \end{aligned}$$

$$
\begin{aligned}
S_4^F &:= \frac{(\sigma_k^F)^2(1-\rho)}{\Phi(\vec{0})} \sum_{\vec{x}} \frac{\chi(\vec{x})}{|\vec{x}+\rho_k|^2} \sum_{i=1}^{K} \sum_{j \neq i} A_{k,i} A_{k,j} x_i x_j \\
&= \frac{(\sigma_k^F)^2(1-\rho)}{\Phi(\vec{0})} \sum_{i=1}^{K} \sum_{j \neq i} A_{k,i} A_{k,j} \sum_{n=0}^{\infty} \frac{s_{i,j}(n)}{(n+1)^2} \\
&= (\sigma_k^F)^2(1-\rho) \left( \sum_{i=1}^{K} \sum_{j \neq i} A_{k,j} A_{k,j} \rho_i \rho_j \right) \sum_{n=1}^{\infty} \frac{n(n-1)}{(n+1)^2} \rho^{n-2} \\
&= \frac{(\sigma_k^F)^2}{\rho^3} (\rho + 3(1-\rho)\log(1-\rho) + 2(1-\rho)\,\mathrm{Li}_2(\rho)) \sum_{i=1}^{K} \sum_{j \neq i} A_{k,j} A_{k,j} \rho_i \rho_j
\end{aligned}
$$

Using $S_1^F, S_2^F, S_3^F$, and $S_4^F$ in (4.22) gives $\mathbb{E}[(p_k^F(\vec{x}))^2]$.

The steps for deriving the second moment under VCG auctions are outlined next.

$$
\begin{aligned}
\mathbb{E}[(p_k^V(\vec{x}))^2] = \frac{(\sigma_k^V)^2(1-\rho)}{\Phi(\vec{0})} \sum_{\vec{x}} \Bigg[ & A_{k,0}^2 + \sum_{m=1}^{K} A_{k,m}^2 x_m^2 \\
& + 2A_{k,0} \sum_{m=1}^{K} A_{k,m} x_m + \sum_{i=1}^{K} \sum_{j \neq i} A_{k,i} A_{k,j} x_i x_j \Bigg] \chi(\vec{x}) \qquad (4.23)
\end{aligned}
$$

Define $S_1^V, S_2^V, S_3^V$, and $S_4^V$ as the following.

$$
\begin{aligned}
S_1^V &:= \frac{(\sigma_k^V)^2(1-\rho)}{\Phi(\vec{0})} A_{k,0}^2 \sum_{\vec{x}} \chi(\vec{x}) \\
&= (\sigma_k^V)^2 A_{k,0}^2
\end{aligned}
$$

$$
\begin{aligned}
S_2^V &:= \frac{(\sigma_k^V)^2(1-\rho)}{\Phi(\vec{0})} \sum_{m=1}^{K} A_{k,m}^2 \sum_{\vec{x}} x_m^2 \chi(\vec{x}) \\
&= \frac{(\sigma_k^V)^2(1-\rho)}{\Phi(\vec{0})} \sum_{m=1}^{K} A_{k,m}^2 \sum_{n=0}^{\infty} s_{m,m}(n) \\
&= \frac{2(\sigma_k^V)^2}{(1-\rho)^2} \sum_{m=1}^{K} A_{k,m}^2 \rho_m^2 + \frac{(\sigma_k^V)^2}{(1-\rho)} \sum_{m=1}^{K} A_{k,m}^2 \rho_m
\end{aligned}
$$

$$S_3^V := \frac{2(\sigma_k^V)^2(1-\rho)}{\Phi(\vec{0})} \sum_{\vec{x}} A_{k,0} \sum_{m=1}^{K} A_{k,m} x_m \chi(\vec{x})$$
$$= \frac{2A_{k,0}(\sigma_k^V)^2}{(1-\rho)} \sum_{m=1}^{K} A_{k,m} \rho_m$$

$$S_4^V := \frac{(\sigma_k^V)^2(1-\rho)}{\Phi(\vec{0})} \sum_{\vec{x}} \sum_{i=1}^{K} \sum_{j \neq i} A_{k,i} A_{k,j} x_i x_j \chi(\vec{x})$$
$$= \frac{2(\sigma_k^V)^2}{(1-\rho)^2} \sum_{i=1}^{K} \sum_{j \neq i} A_{k,i} A_{k,j} \rho_i \rho_j$$

Using $S_1^V, S_2^V, S_3^V$, and $S_4^V$ in (4.23) gives $\mathbb{E}[(p_k^V(\vec{x}))^2]$.

Last, the steps for deriving the second moment of pre-payments under congestion based pricing are outlined.

$$\mathbb{E}[(p_k^L(\vec{x}))^2] = \sum_{\vec{x}} (p_k^L(\vec{x}))^2 \pi(\vec{x})$$
$$= (\sigma_k^L)^2 \sum_{\vec{x}} |\vec{x} + \vec{e}_k|^2 \left( A_{k,0}^2 + \sum_{m=1}^{K} A_{k,m} x_m \right)^2 \pi(\vec{x})$$
$$= (\sigma_k^L)^2 \sum_{\vec{x}} |\vec{x} + \vec{e}_k|^2 A_{k,0}^2 \pi(\vec{x}) + (\sigma_k^L)^2 \sum_{\vec{x}} |\vec{x} + \vec{e}_k|^2 \sum_{m=1}^{K} A_{k,m}^2 x_m^2 \pi(\vec{x})$$
$$+ (\sigma_k^L)^2 2 \sum_{\vec{x}} |\vec{x} + \vec{e}_k|^2 A_{k,0} \sum_{m=1}^{K} A_{k,m} x_m \pi(\vec{x})$$
$$+ (\sigma_k^L)^2 \sum_{\vec{x}} |\vec{x} + \vec{e}_k|^2 \sum_{i=1}^{K} \sum_{j=1, j \neq i}^{K} A_{k,i} A_{k,j} x_i x_j \pi(\vec{x})$$
$$:= S_1^L + S_2^L + S_3^L + S_4^L$$

Now, $S_1^L$ simplifies to

$$\begin{aligned} S_1^L &= (\sigma_k^L)^2 \sum_{\vec{x}} |\vec{x} + \vec{e}_k|^2 A_{k,0}^2 \pi(\vec{x}) \\ &= \frac{(\sigma_k^L)^2 A_{k,0}^2 (1-\rho)}{\Phi(\vec{0})} \sum_{\vec{x}} (|\vec{x}| + 1)^2 \chi(\vec{x}) \\ &= \frac{(\sigma_k^L)^2 A_{k,0}^2 (1+\rho)}{(1-\rho)^2} \end{aligned}$$

Simplify $S_2^L$ as,

$$\begin{aligned} S_2^L &= (\sigma_k^L)^2 \sum_{\vec{x}} |\vec{x} + \vec{e}_k|^2 \sum_{m=1}^{K} A_{k,m}^2 x_m^2 \pi(\vec{x}) \\ &= \frac{(\sigma_k^L)^2 (1-\rho)}{\Phi(\vec{x})} \sum_{m=1}^{K} A_{k,m}^2 \sum_{n=0}^{\infty} (n+1)^2 s_{m,m}(n) \\ &= 2(\sigma_k^L)^2 \sum_{m=1}^{K} A_{k,m}^2 \frac{\rho_m (2 + 9\rho_m + 3\rho_m \rho - \rho - \rho^2)}{(1-\rho)^4}. \end{aligned}$$

For $S_3^L$,

$$\begin{aligned} S_3^L &= (\sigma_k^L)^2 \sum_{\vec{x}} |\vec{x} + \vec{e}_k|^2 2A_{k,0} \sum_{m=1}^{K} A_{k,m} x_m \pi(\vec{x}) \\ &= \frac{2A_{k,0} (\sigma_k^L)^2 (1-\rho)}{\Phi(\vec{0})} \sum_{m=1}^{K} A_{k,m} \sum_{n=0}^{\infty} (n^2 + 2n + 1) s_m(n) \\ &= 2A_{k,0} (\sigma_k^L)^2 \frac{(2\rho + 4)}{(1-\rho)^3} \sum_{m=1}^{K} A_{k,m} \rho_m. \end{aligned}$$

Last, $S_4^L$ is simplified as

$$\begin{aligned} S_4^L &= (\sigma_k^L)^2 \sum_{\vec{x}} |\vec{x} + \vec{e}_k|^2 \sum_{i=1}^{K} \sum_{j=1, j\neq i}^{K} A_{k,i} A_{k,j} x_i x_j \pi(\vec{x}) \\ &= \frac{(\sigma_k^L)^2 (1-\rho)}{\Phi(\vec{0})} \sum_{i=1}^{K} \sum_{j=1, j\neq i}^{K} A_{k,i} A_{k,j} \sum_{n=0}^{\infty} (n+1)^2 s_{i,j}(n) \\ &= \frac{6(\sigma_k^L)^2 (3+\rho)}{(1-\rho)^4} \sum_{i=1}^{K} \sum_{j=1, j\neq i}^{K} A_{k,i} A_{k,j} \rho_i \rho_j. \end{aligned}$$

Combining $S_1^L, S_2^L, S_3^L$ and $S_4^L$ gives the result. □

We note that the proofs rely on further metrics such as $\mathbb{E}[x_i|\vec{x}|]$, $\mathbb{E}[x_i x_k |\vec{x}|^2]$ (higher moments), and $\mathbb{E}[x_i x_j / |\vec{x}|^2]$.

## 4.4 Simulation Results

To gain some qualitative insights, a server with a single class of users with $\lambda_0 = 0.3$ packets/second and $\nu_0 = 1$ bits is considered. The service requirements are assumed to be exponentially distributed. The QoS constraint is defined with $r_{\min} = 0.1$ bits/second. From [82, Corollary 1], the mean sojourn time for an $M/M/1$ $PS$ server with a single class of users and conditioned on $x$ number of users present in the system is given by

$$\mathbb{E}[W_0 |\, x \text{ users in the system}] = \frac{x}{\lambda_0(2b-1)} + \frac{1}{\lambda_0(2b-1))},$$

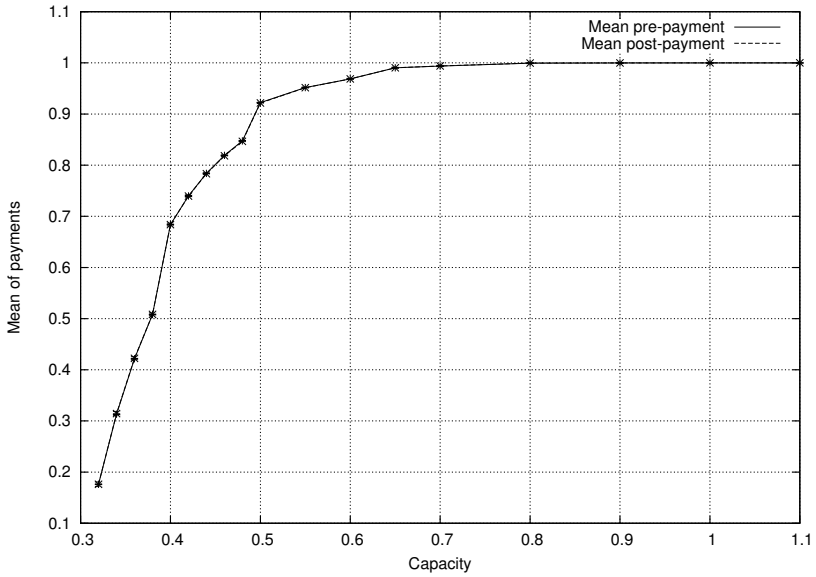
where $b = 1/\rho_0$. For a new arrival observing the system to have $x$ number of users present already, the coefficients $\{A_{0,k}\}$ are determined as follows.

$$\begin{aligned}
\mathbb{E}_x[W_0] &= \mathbb{E}[W_0|x+1 \text{ users in the system}] \\
&= A_{0,0} + A_{0,1}x, \text{ where} \\
A_{0,0} &= \frac{2}{\lambda_0(2b-1)} \\
A_{0,1} &= \frac{1}{\lambda_0(2b-1)}
\end{aligned}$$

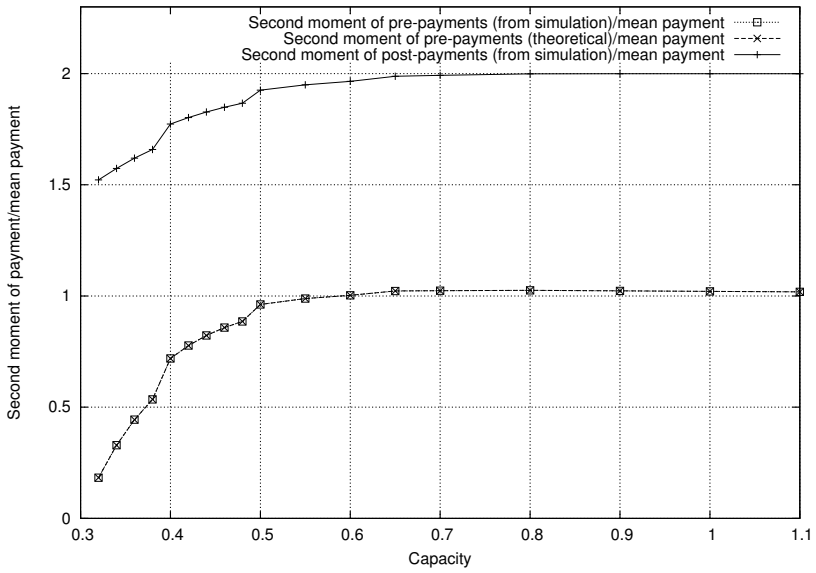
Note that $\mathbb{E}_x[\cdot]$ indicates that the new arrival sees the system with $x$ number of users; the system thus has $x+1$ number of users.

Fig. 4.1a, Fig. 4.2a, and Fig. 4.3a plot the mean payments (from simulations) made by the users under the three pricing models. Note that the traffic intensity $\rho$ decreases as $C$ increases. The discontinuity in the plots is attributed to the discontinuity in $n^* = \lfloor C/r_{\min} \rfloor$. These plots agree with the result of Proposition 7.

Fig. 4.1b, Fig. 4.2b, and Fig. 4.3b plot the ratio of the second moment of payments and the mean payment under the three pricing models and compares the results from simulations with our analysis (Proposition 10). We observe that under fixed-rate pricing and VCG auctions, the pre-payment scheme has a smaller second moment than the post-payment scheme. The operator will prefer the pre-payment scheme under these two pricing models since it generates the same long-term revenue with greater confidence. Under congestion-based pricing, for certain range of the capacity, the pre-pricing scheme exhibits higher variability (see Fig. 4.3b). Thus, the operator's decision of the pricing mechanism will depend on the capacity installed.

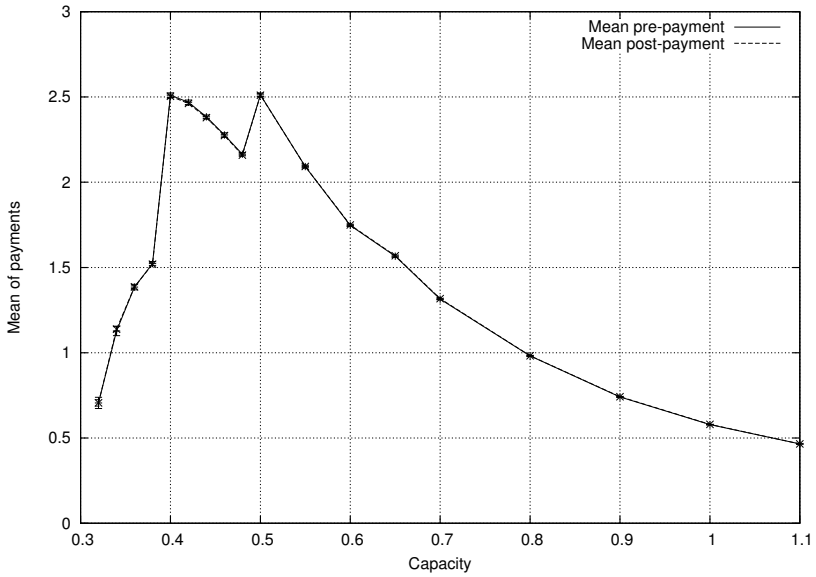



(a) Mean payments under fixed rate pricing from simulations

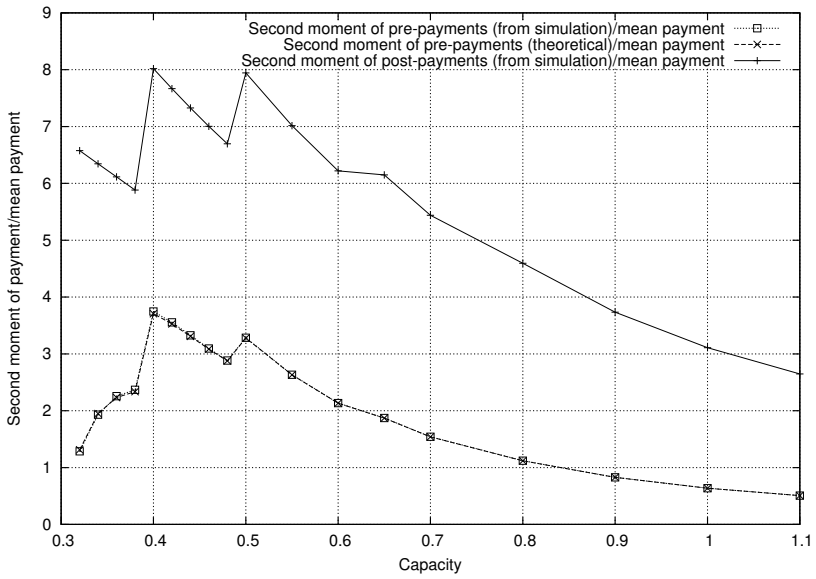



(b) Ratio of second moment of payments and mean payment

Figure 4.1: Results on fixed rate pricing.

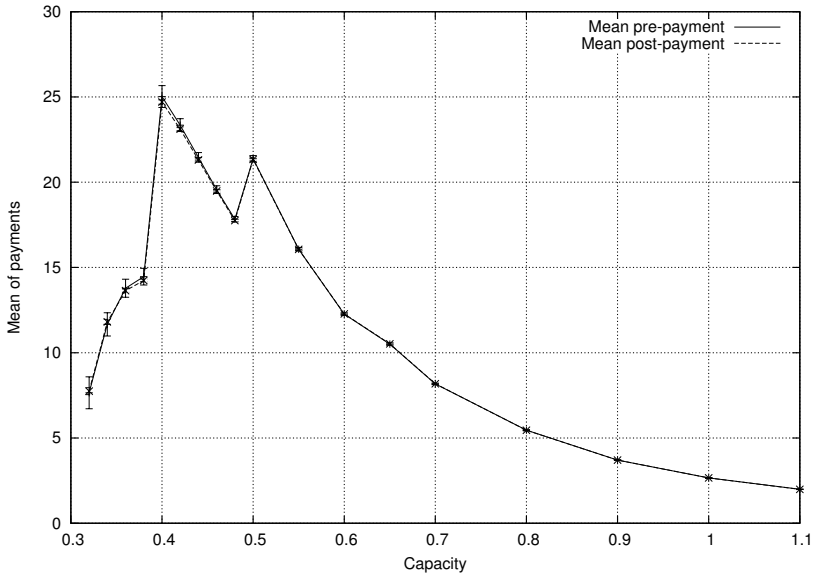



(a) Mean payments under VCG auctions from simulations

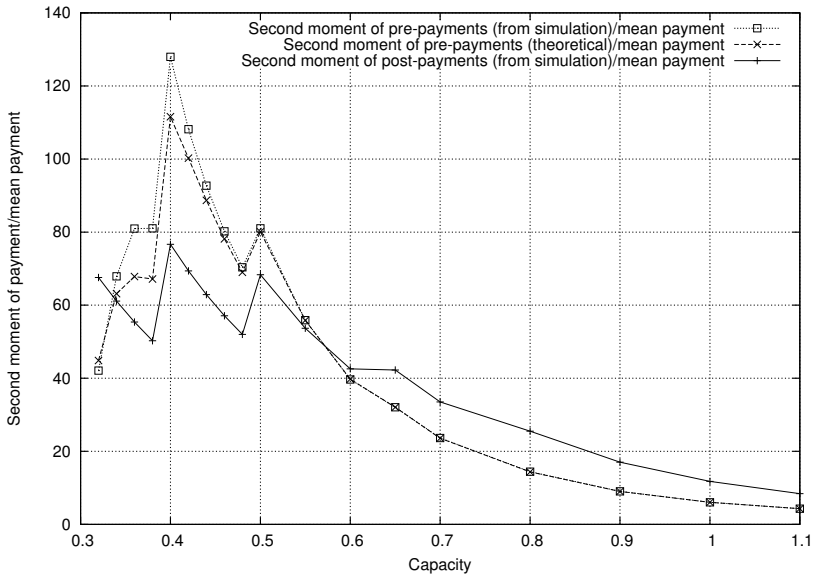



(b) Ratio of second moment of payments and mean payment

Figure 4.2: Results on VCG auction based pricing.




(a) Mean payments under congestion based pricing from simulations




(b) Ratio of second moment of payments and mean payment

Figure 4.3: Results on congestion based pricing.

## 4.5 Summary

The main focus of this chapter is on studying pricing under insensitive allocations. Specifically, we analyze the processor sharing discipline, a well-studied and often used model in networks. We provide insights into the structure of three pricing models: fixed rate pricing, VCG auctions, and congestion-based pricing. We show that the revenue collected by the operator under the three models relate to the increasing moments of the number of users in the system. Under our assumptions, this results in

$$\bar{R}_F \leq \bar{R}_V \leq \bar{R}_L,$$

in an order sense. We also propose a pre-payment mechanism where a user is charged upfront on arrival. Besides having an easier implementation, the revenue earned by the operator in pre-payments is less volatile under fixed-rate pricing and VCG auctions while generating the same mean revenue. The pre-payment mechanism is thus preferable over the post-payment mechanism under these two pricing models.

A criticism of our model is that the users enter the system irrespective of the per unit-time price and the absence of call admission. However, introducing call admission invalidates the insensitivity property. One approach for modelling price based behaviour for users may be by considering Poisson thinning where the arrival rates satisfy the insensitivity property.

The simulation results in Section 4.4 model a resource with a single class of users where the coefficients $\{A_{0,k}\}$ are analytically obtained. For multi-class users, ascertaining the coefficients $\{A_{m,k}\}$ is more difficult (see [81]). In this work, we have tried to understand the role of pricing under insensitivity in order to obtain insights and statistical robustness. Further work is needed to advance towards more realistic pricing models.

# Chapter 5

# Conclusion

In this dissertation, we focused on two problems: improving delay performance in multihop wireless networks and studying pricing under insensitive allocations.

In Chapter 2, we identified three reasons for the poor delay performance of the back-pressure algorithm. Two of the reasons were related to the network flow volumes and the network topology, both of which were considered to be immutable. We rectified the third reason which is the routing of packets akin to a random walk by introducing a framework to prioritize select flows and links to guide packets towards the sink. We emphasized and achieved improved delay performance without a reduction in the capacity of the network.

The back-pressure algorithm strives to maintain stability and does not explicitly minimize delay. In non-stationary environments, our generalized framework was a solution to reduce delays. In Chapter 3, a static mean delay minimization problem was formulated for stationary environments. We saw that the static implementation delivered improved delay performance similar to the remaining hops weighted back-pressure algorithm from Chapter 2. The implementation of static solutions also had a lower per timeslot complexity.

We also proved the non-convexity of the delay function for wireless networks — a result that restricted our analysis to local optima. We believe static solutions to be a step towards distributed routing and scheduling algorithms for multihop wireless networks.

Chapter 4 focused on three pricing models under the processor sharing discipline. We provided the operator's mean revenue, which is important for making functional decisions. As an initial attempt at understanding user behaviour, user payments were also characterized. A pre-payment scheme of charging expected costs was proposed as an alternative to a post-payment mechanism where the users paid the exact cost. The pre-payment scheme generated the same long-term revenue as the post-payment mechanism. Under fixed rate pricing and VCG auctions, we saw that a pre-payment scheme is preferable over charging exact payments since the payments in the former mechanism were predictable with greater confidence.

## 5.1 Extensions

The framework we develop in Chapter 2, i.e., weighted back-pressure algorithms, are of interest even outside the context of delay. Exploring the class of other network metrics that can be improved by simply assigning appropriate weights is of its own importance. Our observations have shown that the weighted back-pressure algorithms are throughput optimal without the adaptive extension. It will be interesting to show whether multiplicative weights do not reduce the capacity region or if an example with random arrivals indeed exists for which the capacity is reduced.

In the static formulation, we primarily focused on two implementations for static scheduling which do not include current network state information in deciding on the schedule to activate. Extensions to make the algorithm more opportunistic may further improve the delay. Implementing routing

parameters may be another aspect to explore for static formulations. We also assumed that the traffic arrival rates were known a priori. Extensions may consider adaptively learning traffic arrival rates, $\lambda$, and adapting routing and scheduling parameters by an online algorithm. Showing that such an algorithm converges will be a key feature of the new algorithm.

We explore three pricing models in Chapter 4. We assume that the users do not behave strategically and do not collude. Studying user incentives would be of interest. In a similar vein, we assume the log utility function in VCG auctions and congestion based pricing. Under this function, 0 allocation to a user has negative infinity utility and thus the user has incentive to always participate. It will be interesting to broaden the class of utilities under processor-sharing. A natural extension to our model is allowing the operator to reject arrivals when the QoS constraint is not met but call admission violates our assumptions.

As noted already, introducing a fee for entering the game or fees relating to the sojourn times are immediate extensions of our formulation and techniques. We have analyzed the case of a single server. As future work, it will be interesting to consider cloud computing operators who possess multiple servers. This extension will add routing of newly arriving users to one of $M$ servers efficiently as a new dimension to the existing formulation.

# Bibliography

[1] S. Birmiwal, R. R. Mazumdar, and S. Sundaram, "Predictable revenue under processor sharing," in *Proceedings of the 46th Annual Conference on Information Sciences and Systems (CISS)*, 2012.

[2] S. Birmiwal, J. Nair, D. Manjunath, and R. R. Mazumdar, "Delay minimization in multihop wireless networks: Static scheduling does it," in *10th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2012.

[3] S. Birmiwal, R. R. Mazumdar, and S. Sundaram, "Processor sharing and pricing implications," in *24th International Teletraffic Congress*, 2012.

[4] 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, 1992.

[5] L. Ying, S. Shakkottai, A. Reddy, and S. Liu, "On combining shortest-path and back-pressure routing over multihop wireless networks," *IEEE/ACM Transactions on Networking*, vol. 19, no. 3, pp. 841–854, June 2011.

[6] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time varying wireless networks," in *Proceedings of IEEE INFOCOM*, vol. 1, 2003, pp. 745–755.

[7] L. Georgiadis, M. J. Neely, and L. Tassiulas, *Resource allocation and cross layer control in wireless networks*, ser. Foundations and Trends in Networking. Now Publishers, 2006.

[8] X. J. Lin and N. B. Shroff, "Joint rate control and scheduling in multihop wireless networks," in *Proceedings of IEEE Conference on Decision and Control*, Paradise Island, Bahamas, December 2004, pp. 1484–1489.

[9] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Transactions on Communications*, vol. 47, no. 8, pp. 1260 –1267, August 1999.

[10] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches," in *Proceedings of IEEE INFOCOM*, vol. 2, 1998, pp. 533–539.

[11] M. J. Neely, "Delay analysis for maximal scheduling with flow control in wireless networks with bursty traffic," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1146–1159, August 2009.

[12] 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, pp. 505–521, 2006.

[13] X. Lin and N. B. Shroff, "The impact of imperfect scheduling on cross-layer rate control in multihop wireless networks," *IEEE/ACM Transactions on Networking*, vol. 14, no. 2, pp. 302–315, April 2006.

[14] C. Joo, X. Lin, and N. B. Shroff, "Performance limits of greedy maximal matching in multi-hop wireless networks," in *Proceedings of 46th IEEE Conference on Decision and Control*, December 2007, pp. 1128 –1133.

[15] ——, "Understanding the capacity region of the greedy maximal scheduling algorithm in multi-hop wireless networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1132–1145, August 2009.

[16] G. Sharma, R. Mazumdar, and N. Shroff, "Delay and capacity trade-offs in mobile ad hoc networks: A global perspective," in *Proceedings of IEEE INFOCOM*, 2006, pp. 1–12.

[17] L. B. Le, K. Jagannathan, and E. Modiano, "Delay analysis of maximum weight scheduling in wireless ad hoc networks," in *Proceedings of Conference on Information Sciences and Systems (CISS)*, March 2009, pp. 389–394.

[18] G. R. Gupta, S. Sanghavi, and N. B. Shroff, "Node weighted scheduling," in *Proceedings of ACM SIGMETRICS/Performance*, 2009, pp. 97–108.

[19] P. Gupta and T. Javidi, "Towards throughput and delay-optimal routing for wireless ad-hoc networks," in *Proceedings of the Asilomar Conference on Signals, Systems and Computers*, November 2007, pp. 249–254.

[20] L. Huang and M. J. Neely, "Delay reduction via Lagrange multipliers in stochastic network optimization," *IEEE Transactions on Automatic Control*, vol. 56, no. 4, pp. 842–857, April 2011.

[21] G. R. Gupta and N. B. Shroff, "Delay analysis for wireless networks with single hop traffic and general interference constraints," *IEEE Transactions on Networking*, vol. 18, no. 2, pp. 393–405, April 2010.

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

[23] B. Sadiq, S. J. Baek, and G. de Veciana., "Delay-optimal opportunistic scheduling and approximations: The log rule," in *Proceedings of IEEE INFOCOM*, April 2009, pp. 1692 – 1700.

[24] M. Lotfinezhad, B. Liang, and E. S. Sousa, "On stability region and delay performance of linear memory randomized scheduling for randomized scheduling for time varying networks," *IEEE Transactions on Networking*, vol. 17, no. 6, pp. 1860–1873, December 2009.

[25] S. Jagabathula and D. Shah, "Optimal delay scheduling in networks with arbitrary constraints," in *Proceedings of ACM SIGMETRIC/Performance*, 2008, pp. 395–406.

[26] L. Huang and M. J. Neely, "Delay efficient scheduling via redundant constraints in multihop networks," *Performance Evaluation*, vol. 68, no. 8, pp. 670–689, 2011.

[27] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, R. Vijayakumar, and P. Whiting, "Scheduling in a queuing system with asynchronously varying service rates," *Probability in the Engineering and Informational Sciences*, vol. 18, no. 2, pp. 191–217, 2004.

[28] 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.

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

[30] D. Shah, D. N. C. Tse, and J. N. Tsitsiklis, "Hardness of low delay network scheduling," *IEEE Transactions on Information Theory*, vol. 57, no. 12, pp. 7810–7817, 2011.

[31] M. Lotfinezhad and P. Marbach, "Throughput-optimal random access with order-optimal delay," in *Proceedings of IEEE INFOCOM*, April 2011, pp. 2867–2875.

[32] T. Roughgarden, *The Price of Anarchy*. MIT Press, 2005.

[33] D. Shah, J. N. Tsitsiklis, and Y. Zhong, "Qualitative properties of $\alpha$-weighted scheduling policies," in *Proceedings of ACM SIGMETRICS*, 2010, pp. 239–250.

[34] J.-H. Hoepman, S. Kutten, and Z. Lotker, "Efficient distributed weighted matching on trees," in *13th Coll. on Structural Information and Communication Complexity*, 2006.

[35] L. Fratta, M. Gerla, and L. Kleinrock, "The flow deviation network method: An approach to store-and-forward communication network design," *Networks*, vol. 3, no. 2, pp. 97–133, 1973.

[36] R. Gallager, "A minimum delay routing algorithm using distributed computation," *IEEE Transactions on Communications*, vol. 25, no. 1, pp. 73–85, January 1977.

[37] D. Bertsekas, E. Gafni, and R. Gallager, "Second derivative algorithms for minimum delay distributed routing in networks," *IEEE Transactions on Communications*, vol. 32, no. 8, pp. 911–919, August 1984.

[38] J. Tsitsiklis and D. Bertsekas, "Distributed asynchronous optimal routing in data networks," *IEEE Transactions on Automatic Control*, vol. 31, no. 4, pp. 325–332, April 1986.

[39] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Prentice Hall, 1992.

[40] M. Schwartz and C. Cheung, "The gradient projection algorithm for multiple routing in message-switched networks," *IEEE Transactions on Communications*, vol. 24, no. 4, pp. 449–456, April 1976.

[41] D. P. Bertsekas, *Nonlinear Programming.* Athena Scientific, 1999.

[42] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 3rd ed. Wiley-Interscience, May 2006.

[43] S. Anily and A. Federgruen, "Simulated annealing methods with general acceptance probabilities," *Journal of Applied Probability*, vol. 24, no. 3, pp. 657–667, 1987.

[44] I. O. Bohachevsky, M. E. Johnson, and M. L. Stein, "Generalized simulated annealing for function optimization," *Technometrics*, vol. 28, no. 3, pp. 209–217, 1986.

[45] M. Locatelli, "Convergence properties of simulated annealing for continuous global optimization," *Journal of Applied Probability*, vol. 33, pp. 1127–1140, 1996.

[46] K. M. Cheh, J. B. Goldberg, and R. G. Askin, "A note on the effect of neighborhood structure in simulated annealing," *Computers & Operations Research*, vol. 18, no. 6, pp. 537–547, 1991.

[47] Y. Xi and E. M. Yeh, "Node-based optimal power control, routing, and congestion control in wireless networks," *IEEE Transactions on Information Theory*, vol. 54, no. 9, pp. 4081–4106, September 2008.

[48] V. S. Borkar and P. R. Kumar, "Dynamic cesaro-wardrop equilibration in networks," *IEEE Transactions on Automatic Control*, vol. 48, no. 3, 2003.

[49] J. Nair and D. Manjunath, "Distributed iterative optimal resource allocation with concurrent updates of routing and flow control variables," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, 2009.

[50] F. Kelly, A. Maulloo, and D.K.H.Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *Journal of Operational Research Society*, vol. 49, no. 3, pp. 237–252, 1998.

[51] Y. Xi and E. M. Yeh, "Optimal distributed power control and routing in wireless networks," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, 2006, pp. 2506–2511.

[52] E. Wei, A. Azdaglar, A. Eryilmaz, and A. Jadbabaie, "A distributed newton method for dynamic network utility maximization with delivery contracts," in *Proceedings of the 46th Annual Conference on Information Sciences and Systems (CISS)*, 2012.

[53] A. Eryilmaz, A. Azdaglar, D. Shah, and E. Modiano, "Distributed cross-layer algorithms for the optimal control of multihop wireless networks," *IEEE/ACM Transactions on Networking*, vol. 18, no. 2, 2010.

[54] J. F. Nash, Jr., "The bargaining problem," *Econometrica*, vol. 18, no. 2, 1950.

[55] H. Yaiche, R. R. Mazumdar, and C. Rosenberg, "A game theoretic framework for bandwidth allocation and pricing in broadband networks," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 667–678, 2000.

[56] F. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, pp. 33–37, 1997.

[57] T. Bonald and A. Proutiere, "Insensitive bandwidth sharing in data networks," *Queueing Systems*, vol. 44, no. 1, pp. 69–100, 2003.

[58] R. L. Grossman, "The case for cloud computing," *IT Professional*, vol. 11, no. 2, pp. 23–27, 2009.

[59] E. Altman, K. Avrachenkov, and U. Ayesta, "A survey on discriminatory processor sharing," *Queueing Systems: Theory and Applications*, vol. 54, pp. 53–63, 2006.

[60] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, October 2000.

[61] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown, "Processor sharing flows in the internet," in *Thirteenth International Workshop on Quality of Service (IWQoS)*, 2005, pp. 271–285.

[62] S. Yashkov, "Processor-sharing queues: Some progress in analysis," *Queueing Systems*, vol. 2, no. 1, pp. 1–17, 1987.

[63] Y. Xue, B. Li, and K. Nahrstedt, "Price-based resource allocation in wireless ad hoc networks," in *International Workshop on Quality of Service (IWQoS)*, 2003, pp. 79–96.

[64] J. Huang, R. A. Berry, and M. L. Honig, "Auction-based spectrum sharing," in *Proceedings of WiOpt '04*, 2004, pp. 405–418.

[65] D. Niyato and E. Hossain, "Competitive pricing in heterogeneous wireless access networks: Issues and approaches," *IEEE Network*, vol. 22, no. 6, pp. 4–11, 2008.

[66] S. Maharjan, Y. Zhang, and S. Gjessing, "Economic approaches for cognitive radio networks: A survey," *Wireless Personal Communications*, vol. 57, no. 1, pp. 33–51, 2010.

[67] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *The Journal of Finance*, vol. 16, pp. 8–37, 1961.

[68] T. Roughgarden and M. Sundararajan, "Is efficiency expensive?" in *Third Workshop on Sponsored Search Auctions*, 2007.

[69] F. M. Menzes and P. K. Monteiro, *An introduction to auction theory*. Oxford University Press, 2005.

[70] S. Yang and B. Hajek, "VCG-Kelly mechanisms for allocation of divisible goods: Adapting VCG mechanisms to one-dimensional signals," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 6, pp. 1237–1243, 2007.

[71] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2003.

[72] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Athena Scientific, 1999.

[73] N. S. Walton, "Insensitive, maximum stable allocations converge to proportional fairness," *Queueing Systems*, vol. 68, pp. 51–60, 2011.

[74] P. Whittle, "Partial balance and insensitivity," *Journal of Applied Probability*, vol. 22, no. 1, pp. 168–176, 1985.

[75] F. P. Kelly, *Reversibility and stochastic networks*. Chichester: John Wiley & Sons Ltd., 1979.

[76] T. Bonald and A. Proutiere, "Insensitive bandwidth sharing," in *Proceedings of IEEE GLOBECOM 2002*, vol. 3, 2002, pp. 2659–2663.

[77] ——, "Insensitivity in processor-sharing networks," *Performance Evaluation*, vol. 49, no. 1/4, pp. 193–209, 2002.

[78] J. W. Roberts, "A survey on statistical bandwidth sharing," *Computer Networks and ISDN Systems*, vol. 45, no. 3, pp. 3199–332, 2004.

[79] P. Brémaud, "A Swiss Army Formula of Palm Calculus," *Journal of Applied Probability*, vol. 30, pp. 40–51, 1993.

[80] F. Baccelli and P. Brémaud, *Elements of Queueing Theory: Palm Martingale Calculus and Stochastic Recurrences*, 2nd ed. Springer Verlag, 2003.

[81] K. M. Rege and B. Sengupta, "A decomposition theorem and related results for the discriminatory processor sharing queue," *Queueing Systems*, vol. 18, pp. 333–351, 1994.

[82] B. Sengupta and D. L. Jagerman, "A conditional response time of the M/M/1 processor-sharing queue," *AT&T Technical Journal*, vol. 64, pp. 409–421, 1985.