DETERMINISTIC MATHEMATICAL OPTIMIZATION IN STOCHASTIC NETWORK CONTROL

by

Longbo Huang

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(ELECTRICAL ENGINEERING)

August 2011

Dedication

To my mom and dad: Ximei Lin and Yingzhong Huang

To uncle Liangjun Hu

Acknowledgments

First, I would like to thank my advisor, Professor Michael Neely. To me, he is an ideal advisor, and I feel extremely fortunate to be able to work with him. He is passionate, energetic, rigorous and highly supportive. He gave me the complete freedom in choosing what I wanted to work on, and always encouraged me to pursue further into the topics. He also tried his best to help me get around the many difficulties I faced. I still remember when I wrote my first paper on pricing, he spent so much time helping me that he could have written the paper from scratch himself. When I started my Ph.D., I wished to learn as much from him as possible. Now I think I have learned a lot from him already, but I realize that he still has a lot more that I can learn from.

The second person I want to thank is Professor Bhaskar Krishnamachari, who in many ways acted as my second advisor. I took Bhaskar's EE650 course in Spring 2006, and benefited a lot from his guidance on how to conduct networking research. He has given me enormous encouragement and help throughout the years. After more than five years, I still remember a sentence he said to me in one of the EE650 course project meetings: "A positive mind is the most important thing." This sentence has helped me go through many of the down times I had.

I would like to thank Professor David Kempe, who taught me two wonderful algorithm courses and served as my dissertation committee. In fact, he taught the classes so well that I was once seriously thinking about becoming a CS student and working on algorithms. I am also very grateful for his detailed comments on editing my thesis. I also want to thank Professor Adam Wierman, who has given me a lot of advice on how to improve my research results and develop my academic skills. I am extremely grateful for his help and encouragement during my job searching process. I am also very grateful for all the great guidance from my other committee members: Professor Rahul Jain, Professor Giuseppe Caire and Professor Sheldon Ross.

I want to say thank you to all my friends, in particular, Anita, Mayumi, Gerrie and Milly, Rahul, Chih-ping, Scott, Yuan Yao, Abhshek, Xijin, Frank, Dileep, Yuan Lin, Erica, Carson, Hua and Ying. It is you who made my life at USC much more memorable and be filled with joy. I remember all the wonderful moments I had with you.

I want to express my special thanks to uncle Liangjun Hu. My life trajectory would have been much different should I not meet him when I was in college, and was struggling in finding my own direction for the future.

Finally, the development of this thesis would not have been possible without the constant love and support from my family – my dad and mom, Yingzhong Huang and Ximei Lin, my two brothers, Longzhang and Longjin, and Xiaojun. They have been believing in me and gave me the courage to try out things that I myself was not sure if I would be able to accomplish. Thank you, and I only wish I had a better way to express my gratefulness to you.

Table of Contents

List of Figures

Abstract

In this thesis, we extend the recently developed Lyapunov optimization technique (also known as Max-Weight or Backpressure) for stochastic queueing networks in two important directions: (1) guaranteeing small network delay; and (2) resolving underflows.

To achieve our objective, we first establish an explicit connection between the Lyapunov technique and a randomized dual subgradient method. Based on this connection, we develop a novel exponential attraction result, which states that the network queue backlog under a Lyapunov algorithm deviates from a certain fixed point with a probability that decreases exponentially in the deviation distance. Inspired by the exponential attraction result, we develop three delay-efficient algorithms and show that they achieve near-optimal utility-delay tradeoffs for a general class of multi-hop communication networks. One of the algorithms has also been implemented on a sensor network testbed and was shown to be able to guarantee very small network delay in practical systems.

We later consider the problem of resolving underflows in general complex network scheduling problems. In this case, we propose the weight perturbation technique and develop the Perturbed Max-Weight algorithm (PMW). We show that PMW effectively resolves underflow constraints without sacrificing utility performance. We then apply the perturbation technique to construct utility optimal scheduling algorithms for two

important classes of networks – stochastic processing networks and energy harvesting networks.

The results developed in this thesis highlight the importance of Lagrange multiplier engineering in queueing networks. Specifically, our results show that the queues under the Lyapunov technique indeed correspond to the Lagrange multiplier values under the randomized dual subgradient method. This not only helps us better understand the Lyapunov technique, but also gives us general guidelines on how should one design its algorithm to achieve the desire properties of the queues.

Chapter 1

Introduction

Controlled stochastic queueing networks have been one of the most common and general models for studying complex networked systems. For instance, they can be used to model data networks, transportation networks, manufacturing networks, cellular networks, and social networks. Therefore, developing an optimal control theory for controlled stochastic queueing networks is one of the most fundamental problems in the network science research.

Among the many techniques that have been used for algorithm design, the *Lyapunov optimization technique* (below we mainly focus on the Quadratic Lyapunov function based Algorithm (QLA), which is also known as Backpressure or Max-Weight), first proposed in the seminal work by Tassiulas and Ephremides [TE92], has gradually become the state-of-the-art tool for solving controlled stochastic queueing network problems. The Lyapunov technique is currently receiving much attention in both the theoretical and experimental communities, due to (i) its ability to achieve optimal performance, (ii) its robustness against time-varying network conditions, and (iii) its mathematical tractability. However, the Lyapunov technique has the following two main limitations: (i) It has an unsatisfying

network delay performance for communication network problems; and (ii) The technique does not directly apply to complex networks with "no-underflow" constraints, meaning the network requires some queues to have a sufficiently large size to perform particular actions. An example of a network with "no-underflow" constraints are energy harvesting networks that require "energy queues" to have enough energy to make wireless transmissions. Another example is a product assembly or processing network that combines two or more types of objects before delivery to the next stage, and hence requires the number of objects of each type to be sufficiently large.

In this thesis, we extend Lyapunov optimization theory to resolve the above two limitations. Specifically, we present a systematic way of designing Lyapunov-type (or Max-Weight type) delay-efficient algorithms for general communication network optimization problems, as well as a novel approach to handle the no-underflow constraints. These two extensions greatly broaden the range of problems that can be solved by the Lyapunov technique. Resolving these two problems is highly nontrivial. For the network delay problem, current Lyapunov algorithms are typically analyzed by Lyapunov drift arguments, which only provide upper bounds of the network delay and the results depend heavily on the choice of the Lyapunov function; this lacks a systematic approach. For the underflow problem, the "no-underflow" constraint couples all the actions in time, i.e., the current action may affect the feasibility of a future action. Thus, previous works in this area typically use dynamic programming as the main tool and suffer greatly from the "curse-of-dimensionality." To accomplish our tasks, we establish a novel mathematical programming view of the Lyapunov technique. This connection between the classic mathematical programming theory and the Lyapunov technique not only provides

us with new insights into the Lyapunov method, but also suggests new ways to design delay-efficient/underflow-capable optimal network control algorithms.

1.1 The general stochastic network optimization problem and the QLA (Max-Weight) algorithm

In this thesis, we consider the following general stochastic network optimization problem: We are given a discrete-time stochastic network. The network state, which characterizes the randomness in the network, such as the network channel condition or the random number of arrivals, is time varying according to some probability law. At every time slot, the network controller performs some action chosen from some *feasible* action set based on the observed network state. The chosen action incurs a cost, [1] but also serves some amount of traffic and possibly generates new traffic for the network. This traffic causes congestion, and thus leads to backlogs at nodes in the network. The goal of the controller is to minimize its time average cost subject to the constraint that the time average total backlog in the network is finite.

This setting is very general. Many existing network optimization works fall into this category, and many techniques have been used to study this problem (see [YC08] for a survey). Also note that this framework can be used to model many stability problems, which correspond to the case where the cost incurred by any action is the same. Out of the many proposed approaches, the class of algorithms built upon quadratic Lyapunov functions (called Quadratic Lyapunov function based Algorithms (QLA), also known as

[1] Since cost minimization is mathematically equivalent to utility maximization, in the sense that any cost minimization problem can be written as a utility maximization problem and vice versa, below we will use cost and utility interchangeably.

Max-Weight or Backpressure) have been receiving much recent attention. These QLA algorithms were first proposed in the seminal work by Tassiulas et al. [TE92] for network scheduling, and were later extended into the network utility optimization context by Neely et al. [NML08]. They have been proven to be capable of solving a wide range of communication network problems, for instance, throughput optimal routing [NMR05], network energy minimization [Nee06c], network utility maximization with fairness consideration [NML08], network pricing [HN10c] and cognitive radio applications [UN08]. These QLA algorithms are easy to construct and implement, greedy in nature, robust to network condition changes, and most importantly, they *do not require any statistical knowledge* of the complicated underlying stochastic process in the network. In the network utility optimization context, QLA offers a scalar control parameter $V \geq 1$ to control the distance to the optimal utility. It has been shown in [GNT06] that when the network state is i.i.d., QLA algorithms can guarantee a time average utility that is within $O(1/V)$ to the optimal. Therefore, as V grows large, the time average utility can be pushed arbitrarily close to the optimal. In the pure routing and scheduling context, QLA (in this context, we will follow the convention to call it the Max-Weight algorithm and mainly refer to the Dynamic Routing and Power Control algorithm (DRPC) in [NMR05]) is known to be *throughput-optimal* in that it can stabilize the network whenever it is possible to stabilize it [NMR05].

However, though being a powerful tool, QLA has two main drawbacks: (i) it usually incurs a large network delay. Specifically, when achieving the $O(1/V)$ close-to-optimal utility, one can only guarantee that the incurred network delay is $O(V)$; and (ii) it cannot be directly applied to complex network problems that involve "no-underflow"

constraints. In this thesis, we extend Lyapunov network optimization theory to resolve the two aforementioned limitations in a *systematic* way. The development of the results in this thesis also provide us with more insights into designing optimal network control algorithms.

1.2 Organization of the thesis

This thesis consists of the following components:

• Part I - Optimal Dynamic Pricing [HN10c]: This part consists of Chapter 2, and is devoted to constructing optimal dynamic pricing algorithms for networks with random demand and service opportunities. This problem, though being a special case of the general utility maximization framework defined in Chapter 3, is indeed a problem of its own interest. We develop an optimal online pricing and service scheduling policy for network access points under demand and service opportunity uncertainties, and prove the intriguing "Optimality of Two Prices" theorem, which states that it is always sufficient and sometimes necessary to use a "regular price" together with a "sale price" for achieving optimal revenue.

• Part II - Delay-Efficient Scheduling for Communication Networks [HN09] [HN11a] [HN10d] [HNar] [HN10a] [HMNK11]: This part consists of Chapters 3, 4 and 5, and presents a systematic solution for developing delay-efficient scheduling algorithms for general data network problems. In this part, we first describe the general network utility optimization model in Chapter 3 and discuss the applicability and the limitations of the model. Then in Chapter 4, we develop delay-efficient algorithms for the general network

utility optimization problem defined in Chapter 3. Specifically, we first relate the QLA algorithm to a corresponding *deterministic* mathematical program. We then prove that the backlog vector under QLA is exponentially attracted to some attractor, which is the dual optimal solution of the deterministic mathematical program. Using this exponential attraction result, we develop the Fast-QLA algorithm (FQLA) and show that FQLA guarantees an $O([\log(V)]^2)$ network delay when the utility is pushed to within $O(1/V)$ of the optimal value. This contrasts with the $O(V)$ delay incurred by QLA with the same $O(1/V)$ utility performance. We then develop the LIFO-Backpressure technique and the LIFO^p-Backpressure method, which admit a simpler implementation compared to FQLA, and achieve the same $[O(1/V), O([\log(V)]^2)]$ utility-delay tradeoff. Lastly, we develop the novel "redundant constraint" method for stochastic network stability problems in Chapter 5, which can be viewed as complementing Chapter 4 by exploring delay reduction opportunities with respect to the network size.

• Part III - Utility Optimal Scheduling for Complex Networks [HN10b] [HN11b]: This part contains Chapters 6 and 7. It proposes a general framework for modeling scheduling problems in general complex networks that involve "no-underflow" constraints. Such networks are very common in practice, and scheduling problems in such networks are much more complicated than those in traditional communication networks. We provide a systematic way for designing and analyzing algorithms for scheduling problems in such networks. In Chapter 6, we extend Lyapunov theory to networks that involve the "no-underflow" constraints by developing the novel Perturbed Max-Weight (PMW) technique, which is greedy and has low implementation complexity, and applying it to general stochastic processing networks. In Chapter 7, we apply the methodology developed in

Chapter 6 to energy harvesting networks. In this case, we develop an *online* algorithm that requires *no* knowledge of the random harvestable energy process and show that our algorithm can achieve a close-to-optimal utility using *finite* energy storage devices.

• Part IV - Conclusion and Future Work: Finally, we conclude our thesis in Chapter 8, and outline a few future research directions.

1.3 Notation

Here we introduce the notations used in this thesis:

- $\mathbb{R}$: the set of real numbers
- $\mathbb{R}_+$ (or $\mathbb{R}_-$): the set of nonnegative (or non-positive) real numbers
- $\mathbb{R}^n$ (or $\mathbb{R}_+^n$): the set of n dimensional *column* vectors, with each element being in $\mathbb{R}$ (or $\mathbb{R}_+$)
- **bold** symbols $\boldsymbol{a}$ and $\boldsymbol{a}^T$: *column* vector and its transpose
- $\boldsymbol{a} \succeq \boldsymbol{b}$: vector $\boldsymbol{a}$ is entrywise no less than vector $\boldsymbol{b}$
- $||\boldsymbol{a} - \boldsymbol{b}||$: the Euclidean distance of $\boldsymbol{a}$ and $\boldsymbol{b}$
- $||\boldsymbol{a}||_\infty$: the sup norm of $\boldsymbol{a}$
- $\mathbf{0}$: column vector with all elements being 0
- $\log(\cdot)$: the natural log function
- $\big[x\big]^+$: $\max[x, 0]$
- i.i.d.: independently and identically distributed

Part I

Optimal Dynamic Pricing

Chapter 2

Network access point pricing: the optimality of two prices

In this chapter, we consider the profit maximization problem of an access point (AP) in a wireless mesh network. The results in this chapter are based in part on our conference and journal papers [HN07] and [HN10c].

In this network, mobile users connect to the mesh network via the AP. The AP receives the user data and transmits it to the larger network via a wireless link. Time is slotted with integral slot boundaries $t \in \{0, 1, 2, \ldots\}$, and every timeslot the AP chooses an admission price $p(t)$ (cost per unit packet) and announces this price to all present mobile users. The users react to the current price by sending data, which is queued at the AP. While the AP gains revenue by accepting this data, it in turn has to deliver all the admitted packets by transmitting them over its wireless link. Therefore, it incurs a transmission cost for providing this service (for example, the cost might be proportional to the power consumed due to transmission). The mission of the AP is to find strategies for both packet admission and packet transmission so as to maximize its time average profit while ensuring queue stability.

This problem falls into our general utility optimization framework in Chapter 3, and hence the general delay improvement results developed in Chapter 4 can also be applied here. However, we note that this pricing problem also contains many aspects, such as the two-price result and the demand blind pricing mode, that are interesting in their own right, and hence can indeed be viewed as a problem of independent interest. It will also serve as a demonstration of how the Lyapunov technique works.

2.1 Network model

We consider the network as shown in Fig 2.1. The network is assumed to operate in slotted time, i.e. $t \in \{0, 1, 2, ...\}$.

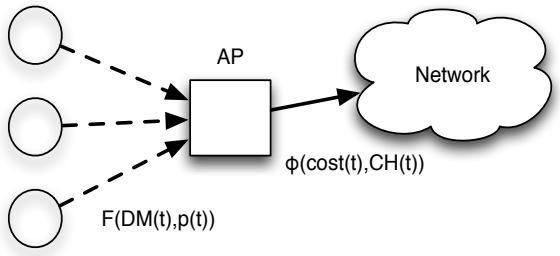

Figure 2.1: An Access Point (AP) that connects mobile users to a larger network.

2.1.1 Arrival model: the demand function

We first describe the packet arrival model. Let $DM(t)$ be the demand state at time t. $DM(t)$ might be the number of present mobile users, or could represent the current demand situation, such as the demand being "High," "Medium" or "Low." We assume that $DM(t)$ evolves according to a finite state ergodic Markov chain with state space $\mathcal{M}$. Let π_m represent the steady state probability that $DM(t) = m$. The value of $DM(t)$ is

assumed known at the beginning of each slot t, although the transition and steady state probabilities are potentially unknown.

Every timeslot, the AP first makes a *business decision* by deciding whether or not to allow new data (this decision can be based on knowledge of the current $DM(t)$ state). Let $Z(t)$ be a 0/1 variable for this decision, defined as:[1]

$$Z(t) = \begin{cases} 1 & \text{if the AP allows new data on slot } t, \\ 0 & \text{else.} \end{cases} \tag{2.1}$$

If the AP chooses $Z(t) = 1$, it then chooses a per-unit price $p(t)$ for incoming data and advertises this price to the mobile users. We assume that price is restricted to a set of price options $\mathcal{P}$, so that $p(t) \in \mathcal{P}$ for all t. We assume that the set $\mathcal{P}$ includes the constraint that prices are non-negative and bounded by some finite maximum price p_{max}. Let $R(t)$ be the total number of packets that are sent by the mobile users in reaction to this price. The income earned by the AP on slot t is thus $Z(t)R(t)p(t)$.

The arrival $R(t)$ is a random variable that depends on the demand state $DM(t)$ and the current price $p(t)$ via a demand function $F(DM(t), p(t))$:

$$F : (DM(t), p(t)) \rightarrow \mathbb{E}\big\{R(t)\big\}. \tag{2.2}$$

Specifically, the demand function maps $DM(t)$ and $p(t)$ into the *expected value* of arrivals $\mathbb{E}\big\{R(t)\big\}$. We further assume that there is a maximum value R_{max}, so that $R(t) \leq R_{max}$ for all t, regardless of $DM(t)$ and $p(t)$. The higher order statistics for $R(t)$ (beyond its expectation and its maximum value) are arbitrary. The random variable $R(t)$ is assumed to be conditionally independent of past history given the current $DM(t)$ and $p(t)$. The

[1]The $Z(t)$ decisions are introduced to allow stability even in the possible situation where user demand is so high that incoming traffic would exceed transmission capabilities, even if price were set to its maximum value p_{max}.

demand function $F(m,p)$ is only assumed to satisfy $0 \leq F(m,p) \leq R_{max}$ for all $m \in \mathcal{M}$ and all $p \in \mathcal{P}$.

Example: In the case when $DM(t)$ represents the number of mobile users in range of the AP at time t, a useful example model for $F(DM(t), p(t))$ is:

$$F(DM(t), p(t)) = A(DM(t))\hat{F}(p(t)),$$

where $\hat{F}(p)$ is the expected number of packets sent by a single user in reaction to price p, a curve that is possibly obtained via empirical data; and $A(DM(t))$ is a non-negative function of $DM(t)$, e.g. $A(DM(t)) = \theta DM(t)$, $\theta \geq 0$, which represents the "effective number of participating users" generated by the $DM(t)$ present users. In this case, we assume that the $A(DM(t))$ is bounded by some value A_{max} and the maximum number of packets sent by any single user is bounded by some value R_{max}^{single}, so that $R_{max} = A_{max}R_{max}^{single}$.

In Section 2.4, we show that this type of demand function (i.e, $F(m,p) = A(m)\hat{F}(p)$) leads to an interesting situation where the AP can make "*demand state blind*" pricing decisions, where prices are chosen without knowledge of $DM(t)$.

2.1.2 Transmission model: the rate-cost function

Let $CH(t)$ represent the channel condition of the wireless link from AP to the mesh network on slot t. We assume that the channel state process $CH(t)$ is a finite state ergodic Markov chain with state space $\mathcal{CH}$. Let π_{ch} represent the steady state probability that $CH(t) = ch$. The transition and steady state probabilities of $CH(t)$ are potentially unknown to the AP, although we assume that the AP knows the current $CH(t)$ value at the beginning of each t.

Every slot t, the AP decides how much resource to allocate for transmission. We model this decision completely by its *cost* to the AP, denoted as $\text{cost}(t)$. We assume that $\text{cost}(t)$ is chosen within some set of costs $\mathcal{C}$, and that $\mathcal{C}$ includes the constraint $0 \leq \text{cost} \leq C_{max}$ for some finite maximum cost C_{max}. The transmission rate is then determined by $\text{cost}(t)$ and the channel state $CH(t)$ according to the *rate-cost*[2] function $\mu(t) = \Phi\left(\text{cost}(t), CH(t)\right)$. In our problem, we assume that $\Phi(0, CH(t)) = 0$ for all $CH(t)$. Further, we assume that there is a finite maximum transmission rate, so that:

$$\Phi\left(\text{cost}(t), CH(t)\right) \leq \mu_{max} \quad \text{for all } \text{cost}(t), CH(t), t. \tag{2.3}$$

We assume that packets can be continuously split, so that $\mu(t) = \Phi\left(\text{cost}(t), CH(t)\right)$ determines the portion of packets that can be sent over the link from AP to the network on slot t (for this reason, the rate function can also be viewed as taking units of *bits*). Of course, the set $\mathcal{C}$ can be restricted to a finite set of costs that correspond to integral units for $\Phi(\text{cost}(t), CH(t))$ in systems where packets cannot be split.

2.1.3 Queueing dynamics and other notations

Let $q(t)$ be the queue backlog of the AP at time t, in units of packets.[3] Note that this is a *single commodity* problem as we do not distinguish packets from different users.[4] We assume the following queueing dynamics for $q(t)$:

$$q(t+1) = \max[q(t) - \mu(t), 0] + Z(t)R(t), \tag{2.4}$$

where $\mu(t) = \Phi(\text{cost}(t), CH(t))$. And we use the queue stability criterion as follows.

$$\overline{q} \triangleq \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{q(\tau)\} < \infty. \tag{2.5}$$

[2]This is essentially the same as the rate-power curve in [Nee06c].
[3]The packet units can be fractional. Alternatively, the backlog could be expressed in units of *bits*.
[4]Our analysis can be extended to treat multi-commodity models.

2.2 Related work

Many existing works on revenue maximization can be found. Work in [PT00] [LS05] models the problem of maximizing revenue as a dynamic program. Work in [AOS04] and [MB02] model revenue maximization as static optimization problems. A game theoretic perspective is considered in [BS02], where equilibrium results are obtained. Works [MW06], [LCL07], and [SS06] also use game theoretic approaches with the goal of obtaining efficient strategies for both the AP and the users. The paper [FP03] looks at the problem from a mechanism design perspective, and [KA03], [ZDT07] consider profit maximization with Qos guarantees. Early work on network pricing in [MMV95], [Kel97a], and [LL99] consider throughput-utility maximization rather than revenue maximization. There, prices play the role of Lagrange multipliers, and are used mainly to facilitate better utilization of the shared network resource. This is very different from the revenue maximization problem that we consider, where the service provider is only interested in its own profit. Indeed, the revenue maximization problem can be much more complex due to non-convexity issues.

The above prior work does not directly solve the profit maximization problem for APs in a wireless network for one or more of the following reasons: (1) Most works consider time-invariant systems, i.e., the network condition does not change with time. (2) Works that model the problem as an optimization problem rely heavily on the assumption that the user utility function or the demand function is concave. (3) Many of the prior works adopt the flow rate allocation model, where a single fixed operating point is obtained and used for all time. However, in a wireless network, the network condition can easily

change due to channel fading and/or node mobility, so that a fixed resource allocation decision may not be efficient. Also, although the utility functions can generally be assumed to be concave, it is easy to construct examples where the demand function is not non-concave/non-convex even if users have concave utility functions. Indeed, profit maximization problems are often non-convex in nature. Hence, they are generally hard to solve, even in the static case where the channel condition, user condition, and demand function is fixed for all time. It is also common to look for single-price solutions in these static network problems. Our results show that single-price solutions are not always optimal, and that even for static problems the AP can only maximize time average profit by providing a "regular" price some fraction of the time, and a "reduced price" at other times. (4) The utility maximization problems use very different models than revenue maximization problems. Thus, algorithms developed there cannot be directly applied here. (5) Most network pricing work considers flow allocation that neglects the packet-based nature of the traffic, and neglects issues of queueing delay. Below, we will use the Lyapunov optimization technique to develop efficient algorithms for solving this general pricing problem.

2.3 Characterizing the maximum profit

In this section, we characterize the optimal average profit that is achievable over the class of all possible control polices that stabilize the queue at the AP. We show that it suffices for the AP to use only *two prices* for every demand state $DM(t)$ to maximize its profit.

2.3.1 The maximum profit

To describe the maximum average profit, we use an approach that is similar to the analysis of the capacity region in [NMR05], [Nee03] and the minimum average power for stability problem in [Nee06c]. Note that in [NMR05], [Nee03] and [Nee06c], the arrival rate is taken as a given parameter, while in our case, the AP needs to balance between the profit from data admission and the cost for packet transmission. The following theorem shows that one can achieve arbitrarily close to the optimality over the class of stationary randomized pricing and transmission scheduling strategies with the following structure: Every slot the AP observes $DM(t) = m$, and makes a business decision $Z(t)$ by independently and randomly choosing $Z(t) = 1$ with probability $\phi^{(m)}$ (for some $\phi^{(m)}$ values defined for each $m \in \mathcal{M}$). If $Z(t) = 1$, then the AP allocates a price randomly from a countable *collection of prices* $\{p_1^{(m)}, p_2^{(m)}, p_3^{(m)}, \ldots\}$, with probabilities $\{\alpha_k^{(m)}\}_{k=1}^{\infty}$. Similarly, the AP observes $CH(t) = ch$ and makes a transmission decision by choosing cost(t) randomly from a set of costs $\{cost_k^{(ch)}\}_{k=1}^{\infty}$ with probabilities $\{\beta_k^{(ch)}\}_{k=1}^{\infty}$.

Theorem 1. *(Maximum Profit with Stability) The optimal average profit for the AP, with its queue being stable, is given by* $Profit_{av}^{opt}$*, where* $Profit_{av}^{opt}$ *is defined as the following:*

$$Profit_{av}^{opt} = \sup\left\{Income_{av} - Cost_{av}\right\} \tag{2.6}$$

$$s.t.\ \ Income_{av} = \mathbb{E}_m\left\{\phi^{(m)} \sum_{k=1}^{\infty} \alpha_k^{(m)} F(m, p_k^{(m)}) p_k^{(m)}\right\} \tag{2.7}$$

$$Cost_{av} = \mathbb{E}_{ch}\left\{\sum_{k=1}^{\infty} \beta_k^{(ch)} cost_k^{(ch)}\right\} \tag{2.8}$$

$$\lambda_{av} = \mathbb{E}_m\left\{\phi^{(m)} \sum_{k=1}^{\infty} \alpha_k^{(m)} F(m, p_k^{(m)})\right\} \tag{2.9}$$

$$\mu_{av} = \mathbb{E}_{ch}\left\{\sum_{k=1}^{\infty} \beta_k^{(ch)} \Phi\left(cost_k^{(ch)}, ch\right)\right\} \tag{2.10}$$

$$\mu_{av} \geq \lambda_{av} \tag{2.11}$$

$$0 \leq \phi^{(m)} \leq 1 \qquad \forall m \in \mathcal{M} \tag{2.12}$$

$$p_k^{(m)} \in \mathcal{P} \qquad \forall k, \forall m \in \mathcal{M} \tag{2.13}$$

$$cost_k^{(ch)} \in \mathcal{C}, \qquad \forall k, \forall ch \in \mathcal{CH} \tag{2.14}$$

$$\sum_{k=1}^{\infty} \alpha_k^{(m)} = 1 \qquad \forall m \in \mathcal{M} \tag{2.15}$$

$$\sum_{k=1}^{\infty} \beta_k^{(ch)} = 1 \qquad \forall ch \in \mathcal{CH} \tag{2.16}$$

where sup{} denotes the supremum, $\mathbb{E}_{ch}$ *and* $\mathbb{E}_m$ *denote the expectation over the steady state distribution for* $CH(t)$ *and* $DM(t)$*, respectively, and* $\phi^{(m)}$, $\alpha_k^{(m)}$, $p_k^{(m)}$, $\beta_k^{(ch)}$, *and* $cost_k^{(ch)}$ *are auxiliary variables with the interpretation given in the text preceding Theorem 1.*

The proof of Theorem 1 contains two parts. Part I shows that no algorithm that stabilizes the AP can achieve an average profit that is larger than Profit_{av}^{opt}. Part II shows that we can achieve a profit of at least $\rho\text{Profit}_{av}^{opt}$ (for any ρ such that $0 < \rho < 1$) with a particular stationary randomized algorithm that also yields average arrival and transmission rates λ_{av} and μ_{av} that satisfy $\lambda_{av} < \mu_{av}$. The formal proof is similar to the proof in [HN07] and is omitted here. The following important corollary to Theorem 1 is simpler and is useful for analysis of the online algorithm described in Section 2.4.

Corollary 1. *For any* $Profit_{av}^{nopt} = Profit_{av}^{opt} - \epsilon^* > 0$,[5] *where* $\epsilon^* > 0$, *there exists a control algorithm* $STAT^*$ *that makes stationary and randomized business and pricing decisions* $Z^*(t)$ *and* $p^*(t)$ *depending only on the current demand state* $DM(t)$ *(and independent*

[5]The case when $\text{Profit}_{av}^{nopt} = 0$ can trivially be satisfied and thus not considered here.

of queue backlog), and makes stationary randomized transmission decisions $cost^*(t)$ *depending only on the current channel state* $CH(t)$ *(and independent of queue backlog) such that:*

$$\mathbb{E}\{Z^*(t)R^*(t)\} \leq \mathbb{E}\{\mu^*(t)\}, \tag{2.17}$$

$$\mathbb{E}\{Z^*(t)p^*(t)F(DM(t),p^*(t))\} - \mathbb{E}\{cost^*(t)\} = Profit_{av}^{nopt}, \tag{2.18}$$

where $\mu^*(t) = \Phi(cost^*(t), CH(t))$*. The above expectations are taken with respect to the steady state distributions for* $DM(t)$ *and* $CH(t)$*. Specifically:*

$$\begin{aligned} \mathbb{E}\big\{Z^*(t)R^*(t)\big\} &= \mathbb{E}_m\{Z^*(t)F(m,p^*(t))\}, \\ \mathbb{E}\big\{\mu^*(t)\big\} &= \mathbb{E}_{ch}\{\Phi(cost^*(t),ch)\}. \qquad \square \end{aligned}$$

2.3.2 The optimality of two prices

The following two theorems show that instead of considering a countably infinite collection of prices $\{p_1^{(m)}, p_2^{(m)}, \ldots\}$ for the stationary algorithm of Corollary 1, it suffices to consider only *two* price options for each distinct demand state $DM(t) \in \mathcal{M}$.

Theorem 2. *Let* $(\lambda^{(m)*}, Income^{(m)*})$ *represent any rate-income tuple formed by a stationary randomized algorithm that chooses* $Z(t) \in \{0,1\}$ *and* $p(t) \in \mathcal{P}$*, so that:*

$$\begin{aligned} \mathbb{E}\big\{Z(t)F(DM(t),p(t)) \,|\, DM(t)=m\big\} &= \lambda^{(m)*}, \\ \mathbb{E}\big\{Z(t)p(t)F(DM(t),p(t)) \,|\, DM(t)=m\big\} &= Income^{(m)*}, \end{aligned}$$

then:

a) $(\lambda^{(m)*}, Income^{(m)*})$ *can be expressed as a convex combination of at most three points in the set* $\Omega^{(m)}$*, defined:*

$$\Omega^{(m)} \triangleq \{(ZF(m,p), ZpF(m,p)) \mid Z \in \{0,1\}, p \in \mathcal{P}\}.$$

b) If $(\lambda^{(m)*}, Income^{(m)*})$ *is on the boundary of the convex hull of* $\Omega^{(m)}$*, then it can be expressed as a convex combination of at most two elements of* $\Omega^{(m)}$*, corresponding to at most two business-price tuples* (Z_1, p_1)*,* (Z_2, p_2)*.*

c) If the demand function $F(m,p)$ *is continuous in* p *for each* $m \in \mathcal{M}$*, and if the set of price options* $\mathcal{P}$ *is* connected*, then any* $(\lambda^{(m)*}, Income^{(m)*})$ *point (possibly not on the boundary of the convex hull of* $\Omega^{(m)}$*) can be expressed as a convex combination of at most two elements of* $\Omega^{(m)}$*.*

Proof. Part (a): It is known that for any vector random variable $\vec{X}$ that takes values within a set Ω, the expected value $\mathbb{E}\{\vec{X}\}$ is in the convex hull of Ω (see, for example, Appendix 4.B in [Nee03]). Therefore, the 2-dimensional point $(\lambda^{(m)*}; \text{Income}^{(m)*})$ is in the convex hull of the set $\Omega^{(m)}$. By Caratheodory's theorem (see, for example, [BNO03]), any point in the convex hull of the 2-dimensional set $\Omega^{(m)}$ can be achieved by a convex combination of at most three elements of $\Omega^{(m)}$.

Part (b): We know from part (a) that $(\lambda^{(m)*}, \text{Income}^{(m)*})$ can be expressed as a convex combination of at most three elements of $\Omega^{(m)}$ (say, $\boldsymbol{\omega}_1$, $\boldsymbol{\omega}_2$, and $\boldsymbol{\omega}_3$). Suppose these elements are distinct. Because $(\lambda^{(m)*}, \text{Income}^{(m)*})$ is on the boundary of the convex hull of $\Omega^{(m)}$, it cannot be in the interior of the triangle formed by $\boldsymbol{\omega}_1$, $\boldsymbol{\omega}_2$, and $\boldsymbol{\omega}_3$. Hence, it must be on an edge of the triangle, so that it can be reduced to a convex combination of two or fewer of the $\boldsymbol{\omega}_i$ points.

Part (c): We know from part (a) that $(\lambda^{(m)*}, \text{Income}^{(m)*})$ is in the convex hull of the 2-dimensional set $\Omega^{(m)}$. An extension to Caratheodory's theorem in [HR51] shows that any such point can be expressed as a convex combination of at most *two* points in $\Omega^{(m)}$ if $\Omega^{(m)}$ is the union of at most two connected components. The set $\Omega^{(m)}$ can clearly be written:

$$\Omega^{(m)} = \{(0;0)\} \cup \{(F(m,p); pF(m,p)) \mid p \in \mathcal{P}\},$$

which corresponds to the cases $Z = 0$ and $Z = 1$. Let $\hat{\Omega}^{(m)}$ represent the set on the right-hand side (RHS) of the above union, so that $\Omega^{(m)} = \{(0;0)\} \cup \hat{\Omega}^{(m)}$. Because the $F(m,p)$ function is continuous in p for each $m \in \mathcal{M}$, the set $\hat{\Omega}^{(m)}$ is the image of the connected set $\mathcal{P}$ through the continuous function $(F(m,p), pF(m,p))$, and hence is itself connected [Mun00]. Thus, $\Omega^{(m)}$ is the union of at most two connected components. It follows that $(\lambda^{(m)*}; \text{Income}^{(m)*})$ can be achieved via a convex combination of at most two elements in $\Omega^{(m)}$. □

Theorem 3. *(Optimality of Two Prices) Let $(\lambda^*, Income^*)$ represent the rate-income tuple corresponding to any stationary randomized policy $Z^*(t)$, $p^*(t)$, $cost^*(t)$, possibly being the policies of Corollary 1 that achieve any near optimal profit $Profit_{av}^{nopt}$. Specifically, assume that the algorithm yields an average profit $Profit_{av}^*$ (defined by the left hand side of (2.18)), and that:*

$$\begin{aligned} \lambda^* &= \mathbb{E}_m\{Z^*(t)F(m,p^*(t))\}, \\ Income^* &= \mathbb{E}_m\{Z^*(t)p^*(t)F(m,p^*(t))\}. \end{aligned}$$

Then for each $m \in \mathcal{M}$, there exists two business-price tuples $(Z_1^{(m)}, p_1^{(m)})$ and $(Z_2^{(m)}, p_2^{(m)})$ and two probabilities $q_1^{(m)}, q_2^{(m)}$ (where $q_1^{(m)} + q_2^{(m)} = 1$) such that:

$$\begin{aligned} \lambda^* &= \sum_{m\in\mathcal{M}} \pi_m \sum_{i=1}^{2} \left[q_i^{(m)} Z_i^{(m)} F(m, p_i^{(m)}) \right], \\ \textit{Income}^* &\leq \sum_{m\in\mathcal{M}} \pi_m \sum_{i=1}^{2} \left[q_i^{(m)} Z_i^{(m)} p_i^{(m)} F(m, p_i^{(m)}) \right]. \end{aligned}$$

That is, a new stationary randomized pricing policy can be constructed that yields the same average arrival rate λ^* and an average income that is greater than or equal to Income*, but which uses at most two prices for each state $m \in \mathcal{M}$.[6]

Proof. For the stationary randomized policy $Z^*(t)$ and $p^*(t)$, define:

$$\begin{aligned} \lambda^{(m)*} &\triangleq \mathbb{E}\big\{ Z^*(t) F(m, p^*(t)) \mid DM(t) = m \big\}, \\ \text{Income}^{(m)*} &\triangleq \mathbb{E}\big\{ Z^*(t) p^*(t) F(m, p^*(t)) \mid DM(t) = m \big\}. \end{aligned}$$

Note that the point $(\lambda^{(m)*}, \text{Income}^{(m)*})$ can be expressed as a convex combination of at most three points $\boldsymbol{\omega}_1^{(m)}$, $\boldsymbol{\omega}_2^{(m)}$, $\boldsymbol{\omega}_3^{(m)}$ in $\Omega^{(m)}$ (from Theorem 2 part (a)). Then $(\lambda^{(m)*}, \text{Income}^{(m)*})$ is inside (or on an edge of) the triangle formed by $\boldsymbol{\omega}_1^{(m)}$, $\boldsymbol{\omega}_2^{(m)}$, $\boldsymbol{\omega}_3^{(m)}$. Thus, for some value $\delta \geq 0$ the point $(\lambda^{(m)*}, \text{Income}^{(m)*} + \delta)$ is on an *edge* of the triangle. Hence, the point $(\lambda^{(m)*}, \text{Income}^{(m)*} + \delta)$ can be achieved by a convex combination of at most two of the $\boldsymbol{\omega}_i^{(m)}$ values. Hence, for each $m \in \mathcal{M}$, we can find a convex combination of two elements of $\Omega^{(m)}$, defining a stationary randomized pricing policy with two business-price choices $(Z_1^{(m)}, p_1^{(m)})$, $(Z_2^{(m)}, p_2^{(m)})$ and two probabilities $q_1^{(m)}$, $q_2^{(m)}$. This new policy yields exactly the same average arrival rate λ^*, and has an average income that is greater than or equal to Income*. □

[6] Because the new average income is greater than or equal to Income*, the new average profit is greater than or equal to Profit_{av}^* when this new pricing policy is used together with the old cost$^*(t)$ scheduling policy.

Most work in network pricing has focused on achieving optimality over the class of single-price solutions, and indeed in some cases it can be shown that optimality can be achieved over this class (so that two prices are not needed). However, such optimality requires special properties of the demand function ([HN10c] provides a sufficient condition for the existence of a single optimal price). Instead, Theorem 3 shows that for *any* demand function $F(m, p)$, the AP can optimize its average profit by using only two prices for every demand state $m \in \mathcal{M}$. We note that there are similar logical arguments about using finite price options to achieve good performance in the economic literature. For example, [McA02] shows that under certain conditions, the social value of using two price classes is at least half of the optimal value. However, we note problems there typically consider selling a certain amount of goods in a given time interval, e.g., [BC03], or assume excessive demand will be lost, e.g., [CFK95], thus are different from our problem, which can be viewed as queueing the excessive demand and serve them later.

Theorem 3 is also related to a classical result of Markov decision theory that bounds the number of required modes for constrained optimization over the class of stationary policies [Alt99]. Indeed, using a more detailed argument as in [Alt99] together with the stationarity and separability of pricing and transmission scheduling that arise from Theorems 1 and 2, our two-price result can likely be extended to show there exists a policy that achieves maximum revenue (or arbitrarily close to it) where most demand states $m \in \mathcal{M}$ use only *one* price, while at most one demand state requires two prices. In fact, the following example shows that the number two is *tight*, in that a single fixed price does not always suffice to achieve optimality.

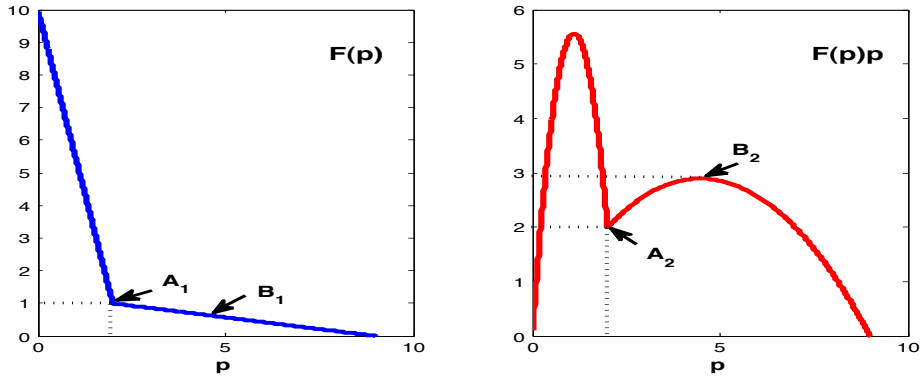

Figure 2.2: $A_1 = (2, 1)$, $B_1 = (\frac{9}{2}, \frac{9}{14})$, $A_2 = (2, 2)$ and $B_2 = (\frac{9}{2}, \frac{81}{28})$.

2.3.3 Example demonstrating necessity of two prices

For simplicity, we consider a static situation where the transmission rate is equal to $\mu = 1$ with zero cost for all t (so that $\Phi(\text{cost}(t), CH(t)) = 1$ for all $CH(t)$ and all $\text{cost}(t)$, including $\text{cost}(t) = 0$). The demand state $DM(t)$ is also assumed to be fixed for all time, so that $F(m, p)$ can be simply written as $F(p)$. Let $\mathcal{P}$ represent the interval $0 \leq p \leq p_{max}$, with $p_{max} = 9$. We consider the following $F(p)$ function:

$$F(p) = \begin{cases} 10 - \frac{9}{2}p & 0 \leq p \leq 2, \\ \frac{9}{7} - \frac{1}{7}p & 2 < p \leq 9. \end{cases} \tag{2.19}$$

Note that the demand curve (2.19) is convex and monotone. Indeed, it can represent a market demand generated by two groups of customers having demands $F(p) = \frac{61}{7} - \frac{61p}{14}$, $0 \leq p \leq 2$ and $F(p) = \frac{9}{7} - \frac{p}{7}$, $0 \leq p \leq 9$. Such demand functions are common in the microeconomic literature, e.g., [BZ99], for modeling real world problems. The $F(p)$ and $pF(p)$ functions corresponding to (2.19) are plotted in Fig. 2.2. Now consider the situation when the AP only uses one price. First we consider the case when $Z(t) = 1$ for all time. Since $\mu = 1$, in order to stabilize the queue, the AP has to choose a price p such that $\lambda = F(p) \leq 1$. Thus we obtain that p has to be greater than 2 (points A_1

and A_2 in Fig. 2.2 show $F(p)$ and $F(p)p$ for $p = 2$).[7] It is easy to show that in this case the best single-price is $p = \frac{9}{2}$ (point B_1 and B_2 in Fig. 2.2 shows its $F(p)$ and $F(p)p$), which yields an average profit of $Profit_{single} = 81/28 \approx 2.8929$. However, we see that in this case the average arrival rate $F(p)$ is only $9/14 \approx 0.6429$, which is only about 65% of μ. Now consider an alternative scheme that uses two prices $p_1 = \frac{13}{9}$ and $p_2 = \frac{9}{2}$, [8] with probabilities of $\frac{1}{10}$ and $\frac{9}{10}$, respectively. Then the resulting profit is

$$\begin{aligned} Profit_{Two} &= \frac{1}{10}F(p_1)p_1 + \frac{9}{10}F(p_2)p_2 \\ &= \frac{1}{10}\cdot\frac{7}{2}\cdot\frac{13}{9} + \frac{9}{10}\cdot\frac{9}{2}\cdot\frac{9}{14} \approx 3.1091, \end{aligned}$$

which is strictly larger than $Profit_{single}$. Further, the resulting arrival rate is only

$$\lambda_{Two} = \frac{1}{10}F(p_1) + \frac{9}{10}F(p_2) = \frac{1}{10}\cdot\frac{7}{2} + \frac{9}{10}\cdot\frac{9}{14} \approx 0.9286,$$

which is strictly less than $\mu = 1$. Therefore the queue is stable under this scheme [GNT06].

Now consider the case when the AP uses a varying $Z(t)$ and a single fixed price. From Theorem 1 we see that this is equivalent to using a probability $0 < \phi < 1$ to decide whether or not to allow new data for all time.[9] In order to stabilize the queue, the AP has to choose a price p such that $F(p)\phi < \mu$. Thus the average profit in this case would be $F(p)p\phi < p\mu$. If $p \leq 2$, then $F(p)p\phi < 2 \cdot 1 = 2$ (note that this is indeed just an upper bound); else if $2 < p \leq 9$, $F(p)p\phi < F(\frac{9}{2}) \cdot \frac{9}{2} = 81/28$. Both are less than $Profit_{Two}$ obtained above. Therefore, no single price policy is optimal.

It is interesting to note that the demand curve (2.19) actually has two unit-elasticity points (which are usually profit maximization points in the economic literature) [BZ99]:

[7] Throughout this chapter, numbers of this type are numerical results and are accurate enough for our arguments.

[8] These are the two prices corresponding to the two local optimum points in the $F(p)p$ curve.

[9] The case when ϕ=0 is trivial and thus is excluded.

$p = \frac{10}{9}$ and $p = \frac{9}{2}$. However, none of them alone achieves the optimal profit under the capacity constraint. Furthermore, the optimal revenue is not achieved by any time sharing between them. This indeed highlights the importance of Theorem 3 and the need of an efficient algorithm.

2.4 Achieving the maximum profit

Even though Theorem 2 and 3 show the possibility of achieving the optimum average profit by using only two prices for each demand state, in practice, we still need to solve the problem in Theorem 1. This often involves a very large number of variables and would require the exact demand state and channel state distributions, which are usually hard to obtain. To overcome these difficulties, here we develop the dynamic Pricing and Transmission Scheduling Algorithm (PTSA). The algorithm offers a control parameter $V > 0$ that determines the tradeoff between the queue backlog and the proximity to the optimal average profit. For simplicity, we assume that $\mathcal{P}$ is compact and $F(m, p)$ is continuous in $p \in \mathcal{P}$. Likewise, we assume that $\mathcal{C}$ is compact and $\Phi(cost, ch)$ is continuous in $cost \in \mathcal{C}$.[10]

Admission Control: Every slot t, the AP observes the current backlog $q(t)$ and the user demand $DM(t)$ and chooses the price $p(t)$ to be the solution of the following problem:

$$\begin{aligned} \max: \quad & VF(DM(t), p)p - 2q(t)F(DM(t), p) \\ \text{s.t.} \quad & p \in \mathcal{P}. \end{aligned} \tag{2.20}$$

[10]These assumptions are only made to ensure the existence of a well defined *max* in equations (2.20) and (2.21). Without these assumptions, the algorithm and the analysis can similarly be described and obtained, but are more involved.

If for all $p \in \mathcal{P}$ the resulting maximum is less than or equal to zero, the AP sends the "CLOSED" signal ($Z(t) = 0$) and does not accept new data. Else, it sets $Z(t) = 1$ and announces the chosen $p(t)$.

Cost/Transmission: Every slot t, the AP observes the current channel state $CH(t)$ and backlog $q(t)$ and chooses cost(t) to be the solution of the following problem:

$$\begin{aligned} \max: \quad & 2q(t)\Phi(cost, CH(t)) - V cost \\ \text{s.t.} \quad & cost \in \mathcal{C}. \end{aligned} \tag{2.21}$$

The AP then sends out packets according to $\mu(t) = \Phi\left(\text{cost}(t), CH(t)\right)$.

The control policy is thus decoupled into separate algorithms for pricing and transmission scheduling. Note from (2.20) that a larger $q(t)$ increases the negative term $-2q(t)F(DM(t), p)$ in the optimization metric, and hence tends to lead to a higher price $p(t)$. Intuitively, such a slow down of the packet arrival helps alleviate the congestion in the AP. Note that the metric in (2.20) can be written as $F(DM(t), p)\big(Vp - 2q(t)\big)$. This is positive only if p is larger than $2q(t)/V$. Thus, we have the following simple fact:

Lemma 1. *Under the PTSA algorithm, if* $2q(t)/V > p_{max}$*, then* $Z(t) = 0$. □

Notice that PTSA only requires the AP to solve the problems (2.20) and (2.21), which use current $DM(t)$ and $CH(t)$ states but do not require statistical knowledge of how these states evolve. While these problems may be non-convex, we note that they are both optimizing a function of *one* variable over an interval, and hence can be easily be solved to obtain highly accurate solutions. For instance, if the pricing set $\mathcal{P}$ contains 100 pricing options, the pricing decision is made just by comparing the metric in (2.20) over each option. Alternatively, for continuous price options, the function typically has

a small number of sub-intervals over which it is piecewise linear or convex, so that the solution can be obtained by comparing the optimums over each sub-interval.

2.4.1 Performance results

In this section we evaluate the performance of PTSA. The following theorem summarizes the performance results:

Theorem 4. *PTSA stabilizes the AP and achieves the following bounds (assuming* $q(0) = 0$*):*

$$q(t) \leq q_{max} \triangleq V p_{max}/2 + R_{max}, \qquad \forall\, t \tag{2.22}$$

$$Profit_{av} \geq Profit_{av}^{opt} - \frac{\tilde{B}}{V}, \tag{2.23}$$

where:

$$Profit_{av} \triangleq \liminf_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\big\{ Z(\tau)p(\tau)R(\tau) - cost(\tau) \big\},$$

and where $Profit_{av}^{opt}$ *is the optimal profit characterized by (2.6) in Theorem 1, and* $\tilde{B}$ *is defined in equation (2.39) of the proof, and* $\tilde{B} = O(\log(V))$.

Because $\tilde{B}/V = O(\log(V)/V)$, the V parameter can be increased to push the profit arbitrarily close to the optimum value, while the worst case backlog bound grows linearly with V. In fact, we can see from (2.20) and (2.21) that these results are quite intuitive: when using a larger V, the AP is more inclined to admit packets (setting $p(t)$ to a smaller value and only requiring $p(t) \geq 2q(t)/V$). Also, a larger V implies that the AP is more careful in choosing the transmission opportunities (indeed, $\Phi\left(\text{cost}(t), CH(t)\right)$ must be more cost effective, i.e. larger than $V\text{cost}(t)/2q(t)$). Therefore a larger V would yield a better profit, at the cost of larger backlog. The proof of Theorem 4 is given in 2.4.3.

2.4.2 Discussion of worst case delay

Note that in the special case of a fixed $\mu(t) = \mu$ for all t, the worst case delay of any packet is upper bounded by $(\frac{1}{2}Vp_{max} + R_{max})/\mu$. This is a very useful result. For instance, if the users also require the worst case delay to be no more than some constant D, the AP can choose V to be such that $D \geq (\frac{1}{2}Vp_{max} + R_{max})/\mu$ (provided this inequality is achievable). Then the delay requirement is met and the revenue lost is less than $\tilde{B}/V = O(\log V/V)$. This is due to the fact that the delay constrained optimal revenue is no more than $Profit^*$, while PTSA gets within $\tilde{B}/V$ of $Profit^*$. This is a unique feature of our algorithm, previous results on QoS pricing are usually obtained based on queueing approximations, e.g., [KA03], [ZDT07].

2.4.3 Proof of performance

We first prove (2.22) in Theorem 4:

Proof: ((2.22) in Theorem 4) We prove this by induction. It is easy to see that (2.22) is satisfied at time 0. Now assume that $q(t) \leq Vp_{max}/2 + R_{max}$ for some integer slot $t \geq 0$. We will prove that $q(t+1) \leq Vp_{max}/2 + R_{max}$. We have the following two cases:

(a) $q(t) \leq Vp_{max}/2$: In this case, $q(t+1) \leq Vp_{max}/2 + R_{max}$ by the definition of R_{max}.

(b) $q(t) > Vp_{max}/2$: In this case, $2q(t)/V > p_{max}$. By Lemma 1 the AP will decide not to admit any new data. Therefore $q(t+1) \leq q(t) \leq Vp_{max}/2 + R_{max}$. □

In the following we prove (2.23) in Theorem 4 via a Lyapunov analysis, using the framework of [GNT06]. First define the Lyaponov function $L(q(t))$ to be: $L(q(t)) \triangleq$

$q^2(t)$. Define the one-step *unconditional Lyapunov drift* as $\Delta(t) \triangleq \mathbb{E}\{L(q(t+1)) - L(q(t))\}$. Squaring both sides of (2.4) and rearranging the terms, we see that the drift satisfies:

$$\Delta(t) \leq 2B^2 - \mathbb{E}\{2q(t)\big[\Phi(\text{cost}(t), CH(t)) - Z(t)R(t)\big]\}, \tag{2.24}$$

where $B \triangleq \max[R_{max}, \mu_{max}]$. For a given number $V > 0$, we subtract from both sides the instantaneous profit (scaled by V) and rearrange terms to get:

$$\begin{aligned} &\Delta(t) - V\mathbb{E}\big\{Z(t)p(t)R(t) - \text{cost}(t)\big\} \\ &\qquad \leq 2B^2 - \mathbb{E}\big\{2q(t)\Phi(\text{cost}(t), CH(t)) - V\text{cost}(t)\big\} \\ &\qquad\quad - \mathbb{E}\big\{Z(t)\big[Vp(t)R(t) - 2q(t)R(t)\big]\big\}. \end{aligned} \tag{2.25}$$

Now we see that the PTSA algorithm is designed to *minimize the RHS of the drift expression (2.25) over all alternative control decisions that could be chosen on slot t.* Thus, we have that the drift of PTSA satisfies:

$$\begin{aligned} &\Delta^P(t) - V\mathbb{E}\big\{Z^P(t)p^P(t)R^P(t) - cost^P(t)\big\} \\ &\qquad \leq 2B^2 - \mathbb{E}\big\{2q^P(t)\Phi(cost^*(t), CH(t)) - Vcost^*(t)\big\} \\ &\qquad\quad -\mathbb{E}\big\{Z^*(t)\big[Vp^*(t)R^*(t) - 2q^P(t)R^*(t)\big]\big\}, \end{aligned} \tag{2.26}$$

where the decisions $Z^*(t)$, $p^*(t)$, and $cost^*(t)$ (and the resulting random arrival $R^*(t)$) correspond to any other feasible control action that can be implemented on slot t (subject to the same constraints $p^*(t) \in \mathcal{P}$ and $cost^*(t) \in \mathcal{C}$). Note that we have used notation $\Delta^P(t)$, $Z^P(t)$, $p^P(t)$, $R^P(t)$, and $cost^P(t)$ on the left hand side of the above inequality to emphasize that this left hand side corresponds to the variables associated with the PTSA policy. Note also that, because the PTSA policy has been implemented up to slot t, the queue backlog on the RHS at time t is the backlog associated with the PTSA algorithm and hence is also denoted $q^P(t)$. We emphasize that the RHS of the drift inequality

(2.26) has been modified *only* in those control variables that can be chosen on slot t. Note further that $R^*(t)$ is a random variable that is conditionally independent of the past given the $p^*(t)$ price and the current value of $DM(t)$.

Now consider the alternative control policy $STAT^*$ described in Corollary 1, which chooses decisions $Z^*(t), p^*(t)$ and $cost^*(t)$ on slot t as a pure function of the observed $DM(t)$ and $CH(t)$ states and yields:

$$Profit_{av}^{nopt} = \mathbb{E}_m\{Z^*(t)R^*(t)p^*(t)\} - \mathbb{E}_{ch}\{cost^*(t)\}, \quad (2.27)$$

$$\lambda_{av}^* \triangleq \mathbb{E}_m\{Z^*(t)R^*(t)\} \leq \mu_{av}^* \triangleq \mathbb{E}_{ch}\{\mu^*(t)\}, \quad (2.28)$$

where $Profit_{av}^{nopt} = Profit_{av}^{opt} - \epsilon^*$ and $Profit_{av}^{opt}$ is the optimal average profit defined in Theorem 1, $\mu^*(t) = \Phi(cost^*(t), CH(t))$, and $R^*(t)$ is the random arrival for a given $p^*(t)$ and $DM(t)$. Recall that $\mathbb{E}_m\{\}$ and $\mathbb{E}_{ch}\{\}$ denote expectations over the steady state distributions for $DM(t)$ and $CH(t)$, respectively. Of course, the expectations in (2.27) and (2.28) cannot be directly used in the RHS of (2.26) because the $DM(t)$ and $CH(t)$ distributions at time t may not be the same as their steady state distributions. However, regardless of the initial condition of $DM(0)$ and $CH(0)$ we have:

$$\lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\{Z^*(\tau)p^*(\tau)R^*(\tau) - cost^*(\tau)\} = Profit_{av}^{nopt}. \quad (2.29)$$

Let $f^P(t)$ represent a short-hand notation for the left-hand side of (2.26), and define $g^*(t)$ as the RHS of (2.26), so that:

$$g^*(t) \triangleq 2B^2 - \mathbb{E}\{2q^P(t)[\mu^*(t) - Z^*(t)R^*(t)]\} - V\mathbb{E}\{Z^*(t)p^*(t)R^*(t) - cost^*(t)\}, \quad (2.30)$$

where we have rearranged terms and have used $\mu^*(t)$ to represent $\Phi(cost^*(t), CH(t))$. Thus, the inequality (2.26) is equivalent to $f^P(t) \leq g^*(t)$. To compute a simple upper bound on $g^*(t)$, note that for any integer $d \geq 0$, we have:

$$q^P(t-d) - d\mu_{max} \leq q^P(t) \leq q^P(t-d) + dR_{max}$$

These inequalities hold since the backlog at time t is no smaller than the backlog at time $t-d$ minus the maximum departures during the interval from $t-d$ to t, and is no larger than the backlog at time $t-d$ plus the largest possible arrivals during this interval. Plugging these two inequalities directly into the definition of $g^*(t)$ in (2.30) yields:

$$\begin{aligned} g^*(t) \quad \leq \quad & 2B^2 + 2d(\mu_{max}^2 + R_{max}^2) - \mathbb{E}\big\{2q^P(t-d)[\mu^*(t) - Z^*(t)R^*(t)]\big\} \\ & -V\mathbb{E}\big\{Z^*(t)p^*(t)R^*(t) - cost^*(t)\big\}. \end{aligned} \tag{2.31}$$

Also note that (by the law of iterated expectations):

$$\begin{aligned} & \mathbb{E}\big\{q^P(t-d)\big[\mu^*(t) - Z^*(t)R^*(t)\big]\big\} \\ & = \mathbb{E}\bigg\{q^P(t-d)\mathbb{E}\big\{\big[\mu^*(t) - Z^*(t)R^*(t)\big] \mid \chi(t-d)\big\}\bigg\}, \end{aligned} \tag{2.32}$$

where $\chi(t) \triangleq [DM(t), CH(t), q(t)]$ is the joint demand state, channel state, and queue state of the system. Since $DM(t)$ and $CH(t)$ are Markovian and both have well defined steady state distributions, and the $STAT^*$ policy makes $p^*(t)$ and $cost^*(t)$ decisions as a stationary and random function of the observed $DM(t)$ and $CH(t)$ states (and independent of queue backlog), we see that the resulting processes $\mu^*(t)$ and $Z^*(t)R^*(t)$ are Markovian and have well defined steady state averages. Further, they converge exponentially fast to their steady state values [Ros96]. Of course, we know the steady state averages are

given by μ^*_{av} and λ^*_{av}, respectively. Therefore there exist positive constants θ_1, θ_2, and $0 < \kappa_1, \kappa_2 < 1$, such that:

$$\mathbb{E}\big\{\mu^*(t) \mid \chi(t-d)\big\} \geq \mu^*_{av} - \theta_1\kappa_1^d, \tag{2.33}$$

$$\mathbb{E}\big\{Z^*(t)R^*(t) \mid \chi(t-d)\big\} \leq \lambda^*_{av} + \theta_2\kappa_2^d. \tag{2.34}$$

Plugging (2.33) and (2.34) into (2.32) yields:

$$\mathbb{E}\big\{q^P(t-d)\big[\mu^*(t) - Z^*(t)R^*(t)\big]\big\} \geq -\mathbb{E}\bigg\{q^P(t-d)\left[\theta_1\kappa_1^d + \theta_2\kappa_2^d\right]\bigg\}, \tag{2.35}$$

where we have used the fact that $\lambda^*_{av} \leq \mu^*_{av}$ (from (2.28)). Plugging (2.35) directly into (2.31) yields:

$$g^*(t) \leq B_1 + 2\mathbb{E}\big\{q^P(t-d)(\theta_1\kappa_1^d + \theta_2\kappa_2^d)\big\} - V\mathbb{E}\big\{Z^*(t)p^*(t)R^*(t) - cost^*(t)\big\}, \tag{2.36}$$

where $B_1 \triangleq 2B^2 + 2d(\mu^2_{max} + R^2_{max})$. However, the queue backlog under PTSA is always bounded by q_{max} (by (2.22) in Theorem 4). We now choose d large enough so that $\theta_i\kappa_i^d \leq 1/(2q_{max})$ for $i \in \{1,2\}$. Specifically, by choosing:

$$d \triangleq \left\lceil \max_{i=1,2}\left\{\frac{\log\big(2\theta_i q_{max}\big)}{\log\big(1/\kappa_i\big)}\right\}\right\rceil, \tag{2.37}$$

we have $2q_{max}[\theta_1\kappa_1^d + \theta_2\kappa_2^d] \leq 2$. Inequality (2.36) becomes:

$$g^*(t) \leq B_1 + 2 - V\mathbb{E}\big\{Z^*(t)p^*(t)R^*(t) - cost^*(t)\big\}. \tag{2.38}$$

Now define $\tilde{B}$ as follows:

$$\tilde{B} \triangleq B_1 + 2 = (2d+2)(R^2_{max} + \mu^2_{max}) + 2, \tag{2.39}$$

where d is defined in (2.37). Because $q_{max} = Vp_{max}/2 + R_{max}$ (by (2.22) in Theorem 4), the value of d is $O(\log(V))$, and hence $\tilde{B} = O(\log(V))$. Recalling that $f^P(t) \leq g^*(t)$, where $f^P(t)$ is the left hand side of (2.26), we have:

$$\Delta^P(t) - V\mathbb{E}\big\{Z^P(t)p^P(t)R^P(t) - cost^P(t)\big\} \leq \tilde{B} - V\mathbb{E}\big\{Z^*(t)p^*(t)R^*(t) - cost^*(t)\big\}.$$

The above inequality holds for all t. Summing both sides over $\tau \in \{0, 1, \ldots, t-1\}$ and using $\Delta^P(t) = \mathbb{E}\big\{L(q^P(t+1)) - L(q^P(t))\big\}$, we get:

$$\mathbb{E}\big\{L(q^P(t))\big\} - \mathbb{E}\big\{L(q^P(0))\big\} - V\sum_{\tau=0}^{t-1}\mathbb{E}\big\{Z^P(\tau)p^P(\tau)R^P(\tau) - cost^P(\tau)\big\}$$
$$\leq \tilde{B}t - V\sum_{\tau=0}^{t-1}\mathbb{E}\big\{Z^*(\tau)p^*(\tau)R^*(\tau) - cost^*(\tau)\big\}.$$

Dividing by Vt, using the fact that $L(q^P(t)) \geq 0$, $L(q(0)) = 0$, and taking limits yields:

$$\liminf_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\big\{Z^P(\tau)p^P(\tau)R^P(\tau) - cost^P(\tau)\big\} \geq$$
$$Profit_{av}^{nopt} - \tilde{B}/V = Profit_{av}^{opt} - \epsilon^* - \tilde{B}/V, \tag{2.40}$$

where we have used (2.29). The left hand side of (2.40) is the liminf time average profit of the PTSA algorithm. Now let $\epsilon^* \to 0$ completes the proof of Theorem 4.

2.4.4 Demand blind pricing

In the special case when the demand function $F(m, p)$ takes the form of $F(m, p) = A(m)\hat{F}(p)$ for $A(m) \geq 0$, PTSA can in fact choose the current price *without* looking at the current demand state $DM(t)$. To see this, note in this case that (2.20) can be written as:

$$\begin{aligned} \max: \quad & A(DM(t))\big[V\hat{F}(p)p - 2q(t)\hat{F}(p)\big] \\ \text{s.t.} \quad & p \in \mathcal{P}. \end{aligned} \tag{2.41}$$

Thus we see that the price set by the AP under PTSA is *independent* of $DM(t)$. So in this case, PTSA can make decisions just by looking at the queue backlog value $q(t)$. This will be very useful if acquiring the demand state incurs some cost to the AP.

2.5 Simulation

In this section, we provide simulation results for the PTSA algorithm. We compare two types of arrival processes. In the first case, the arrival $R(t)$ is deterministic and is exactly equal to $F(m, p(t))$. In the other case, we assume that $R(t)$ is a Bernoulli random variable, i.e., $R(t) = 2F(m, p(t))$ or $R(t) = 0$ with equal probability. [11]

Now we provide our simulation results. We assume that $\mathcal{M} = \{\text{Low}, \text{High}\}$, $\mathcal{CH} = \{\text{Good}, \text{Bad}\}$. The demand curve for $DM(t) = \text{Low}$ is given by:

$$F(\text{Low}, p) = \begin{cases} 4 & 0 \leq p \leq 1, \\ -6p + 10 & 1 < p \leq \frac{3}{2}, \\ -\frac{2}{17}p + \frac{20}{17} & \frac{3}{2} < p \leq 10. \end{cases} \tag{2.42}$$

The demand curve for $DM(t) = \text{High}$ is given by:

$$F(\text{High}, p) = \begin{cases} 10 - p & 0 \leq p \leq 2, \\ -6p + 20 & 2 < p \leq 3, \\ -\frac{1}{7}p + \frac{17}{7} & 3 < p \leq 10. \end{cases} \tag{2.43}$$

The rate-cost curve is given by :

$$\Phi(\text{cost}(t), CH(t)) = log(1 + \gamma_{CH(t)}\text{cost}(t)), \tag{2.44}$$

where $0 \leq \text{cost} \leq 10$, $\gamma_{CH(t)} = 2$ if $CH(t) = \text{Good}$ and $\gamma_{CH(t)} = 1$ else. Both the demand state and the channel are assumed to vary according to the two-state Markov Chain in Fig. 2.3.

Fig. 2.4 shows the backlog and profit performance of PTSA under this dynamic setting. We see that the profit converges quickly to the optimum value and the backlog is no larger than the worst case bound. Let's now also look at the prices chosen by PTSA.

[11] For simplicity here, we assume that $R(t)$ can take fractional values. Alternatively, we could restrict packet sizes to integral units and make the probabilities be such that $\mathbb{E}\{R(t) \mid m, p(t)\} = F(m, p(t))$.

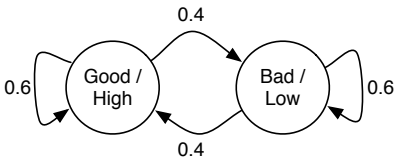

Figure 2.3: The two-state Markov chain.

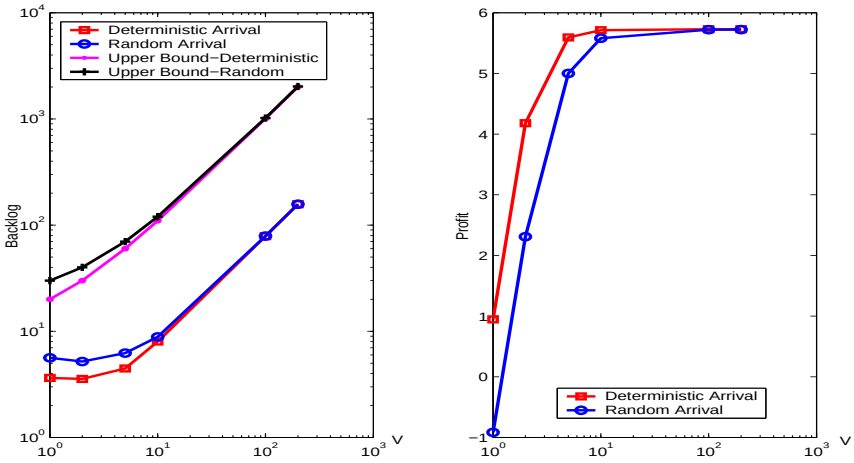

Figure 2.4: Average backlog and average profit v.s. V.

We see in Fig. 2.5 that in fact PTSA quickly determines the optimum prices for each state, and consequently determines the optimum share of the two different demand states. In this case, we see that for each demand state, only one price is chosen. In [HN07], an actual two price phenomenon was observed, in which case for every demand state, two prices are used to achieve the optimal profit. Figure 2.6 also shows the prices sequence under the case when $F(\text{High}, p) = 1 - \frac{1}{10}p^2$ and $F(\text{Low}, p) = \frac{10}{1+p^2}$ for $0 \leq p \leq 10$. In this case, we see that prices fluctuate within some interval over the whole period of time. However, within small time intervals, we again observe the similar one-price-per-state phenomenon as in the previous case.

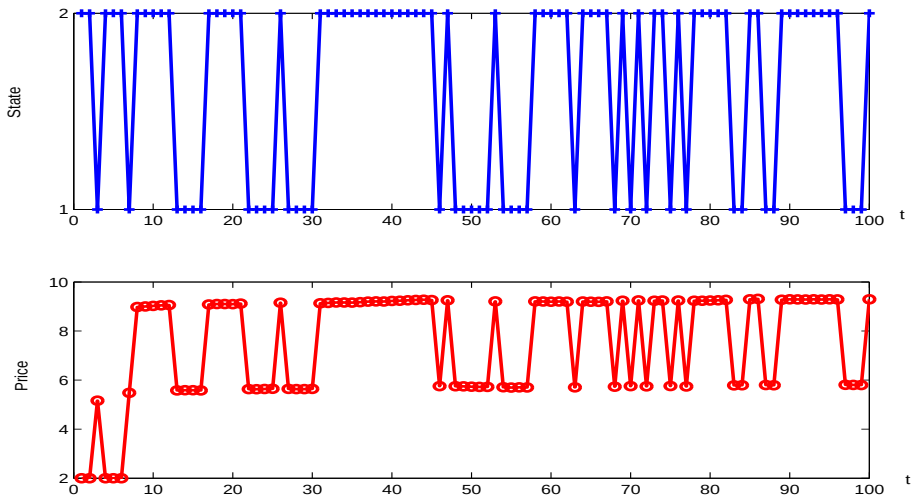

Figure 2.5: Prices chosen according to demand state $DM(t)$ for V=100

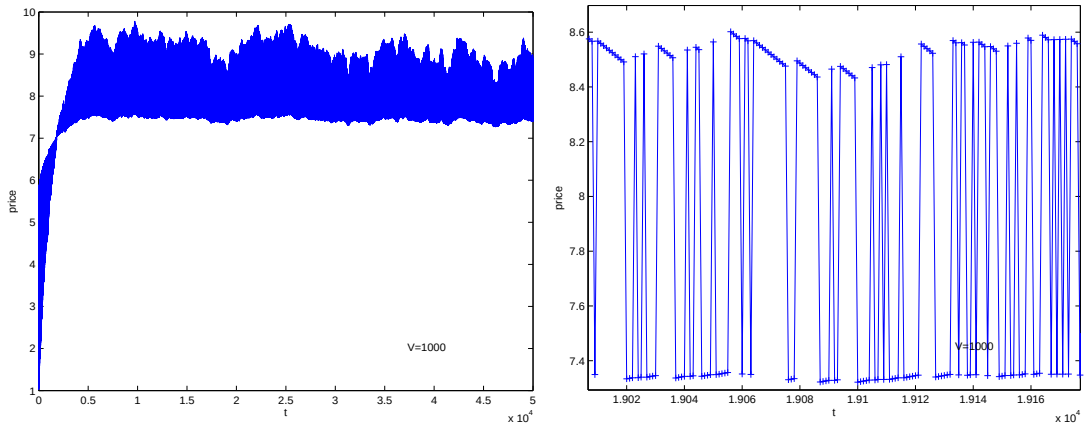

Figure 2.6: LEFT: prices chosen in the first 5×10^4 slots; RIGHT: prices chosen over an interval of 200 slots (V=1000)

2.6 Discussion

In practice, we often see stores going on sale from time to time. It is sometimes believed that going on sale is a way of performing price discrimination [BZ99] between customers that are willing to wait and those who are not, so as to reap profit from different groups of customers. However, Theorem 2 shows that indeed, there is a fundamental reason for stores to go on sale: for achieving the maximum time average profit. We also observe

that the prices generated by the PTSA algorithm, e.g., Fig. 2.6 are quantitatively similar to the time-varying prices that one can usually observe in practice. These results are rather interesting and suggest that using dynamic models like the one considered here may better capture the real world economic behaviors.

In [HN10c], the above model was also extended to treat the case when different customers have different service channels, e.g., different unlink channels from mobile users to the AP. In that case, the Multi-channel PTSA (MPTSA) algorithm has been developed and was shown to achieve the same $[O(\log(V)/V), O(V)]$ profit-delay tradeoff. The game theoretic issue concerning customers anticipating the prices has also been discussed. Examples are given to show that the dynamic prices generated by PTSA can in some cases bypass the effect of user anticipation and help the AP achieve optimal profit in this case.

2.7 Chapter summary

In this chapter, we consider a network access point pricing problem. We prove the "Optimality of Two Prices" theorem, which states that it suffices to use only two prices to achieve optimal profit under *arbitrary* demand functions. We then develop the PTSA algorithm, which is greedy and easy to implement, and prove that PTSA is able to achieve within $O(\log(V)/V)$ of the optimal profit while ensuring a *deterministic* $\Theta(V)$ congestion guarantee.

Part II

Delay-Efficient Scheduling for Communication Networks

Chapter 3

The general network model: the stochastic problem

In this chapter, we present the general network utility optimization problem, called *the stochastic problem*, which can be used to model a wide class of network utility optimization problems, e.g., [NML08], [Nee06c], [UN08], and contains as a special case the network pricing problem in Chapter 2. This model is used in Chapters 4 and 5 for developing delay-efficient algorithms.

3.1 System model

In this section, we specify the general network model we use. We consider a network controller that operates a network with the goal of minimizing the time average cost, subject to the queue stability constraint. The network is assumed to operate in slotted time, i.e., $t \in \{0, 1, 2, ...\}$. We assume that there are $r \geq 1$ *queues* in the network.

3.1.1 Network state

We assume that there are a total of M different random network states, and define $\mathcal{S} = \{s_1, s_2, \ldots, s_M\}$ as the set of possible states. Each particular state s_i indicates the

current network parameters, such as a vector of channel conditions for each link, or a collection of other relevant information about the current network channels and arrivals. Let $S(t)$ denote the network state at time t. We assume that $S(t)$ evolves according to some general probability law, under which there exists a steady state distribution of $S(t)$. For example, $S(t)$ can be i.i.d. every time slot, or it can be a finite state irreducible and aperiodic Markov process. We use π_{s_i} to denote its steady state probabilities of being in state s_i. We assume that the network controller can observe $S(t)$ at the beginning of every slot t, but the statistics of the $S(t)$ process, including the π_{s_i} probabilities and transition probabilities (if $S(t)$ is a Markov process), are not necessarily known. Note that if $S(t)$ contains multiple components, e.g., if $S(t)$ is a vector of channel states of the network, the components can be correlated with each other. See Section 3.1.4 for an example.

3.1.2 The cost, traffic and service

At each time t, after observing $S(t) = s_i$, the controller chooses an action $x(t)$ from a set $\mathcal{X}^{(s_i)}$, i.e., $x(t) = x^{(s_i)}$ for some $x^{(s_i)} \in \mathcal{X}^{(s_i)}$. The set $\mathcal{X}^{(s_i)}$ is called the feasible action set for network state s_i and is assumed to be time-invariant and compact for all $s_i \in \mathcal{S}$. The cost, traffic and service generated by the chosen action $x(t) = x^{(s_i)}$ are as follows:

(a) The chosen action has an associated cost given by the cost function $f(t) = f(s_i, x^{(s_i)}) : \mathcal{X}^{(s_i)} \to \mathbb{R}_+$ (or $\mathcal{X}^{(s_i)} \to \mathbb{R}_-$ in the case of reward maximization problems);

(b) The amount of traffic generated to queue j by the action is determined by the traffic function $A_j(t) = A_j(s_i, x^{(s_i)}) : \mathcal{X}^{(s_i)} \to \mathbb{R}_+$, in units of packets;

(c) The amount of service allocated to queue j is given by the rate function $\mu_j(t) = \mu_j(s_i, x^{(s_i)}) : \mathcal{X}^{(s_i)} \to \mathbb{R}_+$, in units of packets;

Note that $A_j(t)$ includes both the exogenous arrivals from outside the network to queue j, and the endogenous arrivals from other queues, i.e., the transmitted packets from other queues, to queue j. (See Section 3.1.3 and 3.1.4 for further explanations.) We assume that the functions $f(s_i, \cdot)$, $\mu_j(s_i, \cdot)$ and $A_j(s_i, \cdot)$ are time-invariant, their magnitudes are uniformly upper bounded by some constant $\delta_{max} \in (0, \infty)$ for all s_i, j, and they are known to the network operator. We also assume that there exists a set of actions $\{x_k^{(s_i)}\}_{i=1,...,M}^{k=1,2,...,r+2}$ with $x_k^{(s_i)} \in \mathcal{X}^{(s_i)}$, a set of probabilities $\{\vartheta_k^{(s_i)}\}_{i=1,...,M}^{k=1,2,...,r+2}$ and a positive constant $\epsilon > 0$ such that:

$$\sum_{s_i} \pi_{s_i} \Big\{ \sum_k \vartheta_k^{(s_i)} [\mu_j(s_i, x_k^{(s_i)}) - A_j(s_i, x_k^{(s_i)})] \Big\} \leq -\epsilon, \ \forall \ j. \tag{3.1}$$

That is, the constraints are feasible with ϵ-slack. Thus, there exists a stationary randomized policy that stabilizes all queues (where $\vartheta_k^{(s_i)}$ represents the probability of choosing action $x_k^{(s_i)}$ when $S(t) = s_i$). In the following, we use:

$$\boldsymbol{A}(t) = (A_1(t), A_2(t), ..., A_r(t))^T, \tag{3.2}$$

$$\boldsymbol{\mu}(t) = (\mu_1(t), \mu_2(t), ..., \mu_r(t))^T, \tag{3.3}$$

to denote the arrival and service vectors at time t. It is easy to see from above that if we define:

$$B = \sqrt{r}\delta_{max}, \tag{3.4}$$

then since $|A_j(t)|, |\mu_j(t)| \leq \delta_{max}$ for all j, we have $\|\boldsymbol{A}(t) - \boldsymbol{\mu}(t)\| \leq B$ for all t.

3.1.3 Queueing, average cost and the stochastic problem

Let $\boldsymbol{q}(t) = (q_1(t), ..., q_r(t))^T \in \mathbb{R}_+^r$, $t = 0, 1, 2, ...$ be the queue backlog vector process of the network, in units of packets. We assume the following queueing dynamics:

$$q_j(t+1) = \max\big[q_j(t) - \mu_j(t), 0\big] + A_j(t) \quad \forall j, \tag{3.5}$$

and $\boldsymbol{q}(0) = \mathbf{0}$. By using (3.5), we assume that when a queue does not have enough packets to send, null packets are transmitted. In this chapter, we adopt the following notion of queue stability:

$$\mathbb{E}\big\{\sum_{j=1}^{r} q_j\big\} \triangleq \limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{j=1}^{r}\mathbb{E}\big\{q_j(\tau)\big\} < \infty. \tag{3.6}$$

We use Π to denote the policy that the network controller uses to choose actions, and use f_{av}^{Π} to denote the time average cost induced by Π, defined as:

$$f_{av}^{\Pi} \triangleq \limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\big\{f^{\Pi}(\tau)\big\}, \tag{3.7}$$

where $f^{\Pi}(\tau)$ is the cost incurred at time τ by policy Π. We call an action-seeking policy *feasible* if at every time slot t it only chooses actions from the feasible action set $\mathcal{X}^{S(t)}$. We then call a feasible action-seeking policy under which (3.6) holds a *stable* policy, and use f_{av}^{*} to denote the optimal time average cost over all stable policies. In every slot, the network controller observes the current network state and chooses a control action, with the goal of minimizing the time average cost subject to network stability. This goal can be mathematically stated as:

$$\min: \quad f_{av}^{\Pi}, \quad s.t. \ (3.6).$$

In the rest of the chapter, we refer to this problem as *the stochastic problem*. Note that our model can also be used to model stability problems, where the network is trying to support certain traffic flows. This special case is further studied in Chapter 5.

3.1.4 Examples of the model

3.1.4.1 The cccess point pricing problem

Now we describe how the AP pricing problem in Chapter 2 fits into our general model. In the pricing problem, there is only a single queue in the network. Hence we have $r = 1$. The network state, the action, the cost function, and the arrival and service functions are as follows: [1]

1. The network state is $S(t) = (DM(t), CH(t))$, i.e., the (Demand, Channel) pair, and for $s_i = (m, ch)$, π_{s_i} is the steady state probability that $(DM(t), CH(t)) = (m, ch)$.

2. The action $x^{(s_i)}$ at time t under a state s_i is a tuple of $(Z^{(s_i)}, p^{(s_i)}, cost^{(s_i)})$.

3. For each $s_i = (m, ch)$,

   (a) the cost function is $f(s_i, x^{(s_i)}) = cost^{(s_i)} - Z^{(s_i)} F(m, p^{(s_i)}) p^{(s_i)}$,

   (b) the arrival function is $A(s_i, x^{(s_i)}) = Z^{(s_i)} F(m, p^{(s_i)})$,

   (c) the service function is $\mu(s_i, x^{(s_i)}) = \Phi(cost^{(s_i)}, ch)$.

It is also easy to see that in this case, $\delta_{max} = \max[R_{max}, \mu_{max}]$.

3.1.4.2 A 2-queue energy minimization example

We provide another example to illustrate our model. Consider the 2-queue network in Fig. 3.1. In every slot, the network operator decides whether or not to allocate one unit of power to serve packets at each queue, so as to support all arriving traffic, i.e., maintain

[1] Note that in the AP pricing problem, the arrival function and cost function indeed represent the *expected* value of the actual numbers. This can easily be incorporated into the analysis of our general model.

queue stability, with minimum energy expenditure. The number of arriving packets $R(t)$, is i.i.d. over slots, being either 2 or 0 with probabilities 5/8 and 3/8, respectively. Each channel state $CH_1(t)$ or $CH_2(t)$ can be either "G=good" or "B=bad." However, the two channels are correlated, so that $(CH_1(t), CH_2(t))$ can only be in the channel set $\mathcal{CH} = \{(B,B),(B,G),(G,G)\}$. We assume that $(CH_1(t), CH_2(t))$ is i.i.d. over slots and takes every value in $\mathcal{CH}$ with probability $\frac{1}{3}$. When a link's channel state is good, one unit of power can serve 2 packets over the link, otherwise it can only serve one. We assume that power can be allocated to both channels without affecting each other.

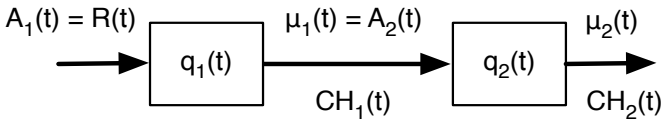

Figure 3.1: A 2-queue system

In this case, the network state $S(t)$ is a triple $(R(t), CH_1(t), CH_2(t))$ and is i.i.d. There are six possible network states. At each state s_i, the action $x^{(s_i)}$ is a pair (x_1, x_2), with x_j being the amount of energy spent at queue j, and $(x_1, x_2) \in \mathcal{X}^{(s_i)} = \{0/1, 0/1\}$. The cost function is $f(s_i, x^{(s_i)}) = x_1 + x_2$, $\forall s_i$. The network states, the traffic functions and service rate functions are summarized in Fig. 3.2. Note here that $A_1(t) = R(t)$ is part of $S(t)$ and is independent of $x^{(s_i)}$; while $A_2(t) = \mu_1(t)$ and hence depends on $x^{(s_i)}$. Also note that $A_2(t)$ equals $\mu_1(t)$ instead of $\min[\mu_1(t), q_1(t)]$ due to our idle fill assumption in Section 3.1.3.

S(t)	$R(t)$	$CH_1(t)$	$CH_2(t)$	$A_1(t)$	$A_2(t)$	$\mu_1(t)$	$\mu_2(t)$
s_1	0	B	B	0	x_1	x_1	x_2
s_2	0	B	G	0	x_1	x_1	$2x_2$
s_3	0	G	G	0	$2x_1$	$2x_1$	$2x_2$
s_4	2	B	B	2	x_1	x_1	x_2
s_5	2	B	G	2	x_1	x_1	$2x_2$
s_6	2	G	G	2	$2x_1$	$2x_1$	$2x_2$

Figure 3.2: Network state, traffic and rate functions

3.2 Discussion of the model

We note that our system model does not explicitly take any queueing delay constraints into account. This is mainly due to the difficulty of charactering the exact queueing delay under control policies in a large network [YZC09]. While problems with explicit queueing constraints can in principle be formulated and solved using the dynamic programming approach [Ber07], algorithms there usually require accurate statistical knowledge of the complicated underlying random process and easily run into the "curse of dimensionality" problem, hence are less practical when compared to the QLA algorithms developed under our system model.

However, though we did not specify the queueing delay constraints, we still get network congestion bounds as a by-product of the Lyapunov analysis. And these delay bounds are very useful. For example, suppose that our objective is to maximize a certain utility subject to some delay constraint. Since it is difficult to capture the exact delay, we can first solve a "relaxed" problem, i.e., replace the delay constraint with the queue stability constraint (3.6). Then, if we can achieve a near-optimal utility with the guarantee that the network delay is no more than the required value, we indeed obtain a control policy

that achieves a utility close to the optimal delay-constrained utility, and ensures the delay requirement. (Section 2.4.2 provides such an example.)

Chapter 4

Achieving the near-optimal $[O(1/V), O([\log(V)]^2)]$ utility-delay tradeoff

In this chapter, we design delay-efficient algorithms for the general stochastic problem defined in Chapter 3, under the assumptions that the network state $S(t)$ is i.i.d. and Markovian, and that the utility function $f(\cdot, \cdot)$ is not a common constant. The results in this chapter are based in part on our conference and journal papers [HN09] [HN11a] [HN10d] [HNar] [HN10a] [HMNK11].

The i.i.d. case has been extensively studied, e.g., in [GNT06], and it has been shown that QLA achieves an $[O(1/V), O(V)]$ utility-delay tradeoff for the problem. Two recent papers ([Nee07] and [Nee06b]) instead construct algorithms based on exponential Lyapunov functions and achieve an $[O(1/V), O(\log(V))]$ tradeoff for the downlink energy minimization problem and the network flow utility maximization problem. The Markovian case, however, has rarely been studied, and results in this case are limited to achieving the $[O(1/V), O(V)]$ tradeoffs for problems in single-hop networks, e.g. [Nee09a].

Rather than using the exponential Lyapunov function approach, we focus on using simpler quadratic Lyapunov functions for constructing delay-efficient algorithms under

both i.i.d. and Markovian network states. We carry out our task in the following steps. In step one, we construct a *deterministic* mathematical program and establish a novel connection between the QLA algorithm and a randomized dual subgradient method applied to the dual problem of the deterministic problem. We then show in the second step that under the QLA algorithm, the network backlog vector is exponentially attracted to some attractor, which is the dual optimal solution of the deterministic problem. In our third step, we construct the Fast-QLA (FQLA) algorithm to subtract out a Lagrange multiplier from the network under QLA and achieve the $[O(1/V), O([\log(V)]^2)]$ utility-delay performance. [1] In the fourth step, we further show that the requirement of having certain knowledge of the optimal Lagrange multiplier in FQLA can be removed when we apply QLA with a Last-In-First-Out (LIFO) queueing rule (called LIFO-Backpressure). Finally, we develop the LIFO^p-Backpressure rule, which is a generalized version of LIFO-Backpressure and uses the LIFO discipline with probability p. We show that LIFO^p-Backpressure achieves a similar tradeoff for a p fraction of the traffic.

The techniques developed in this chapter are general, and apply to general multi-hop network problems with Markovian network states. As we will see, the connection between the Lyapunov technique and the deterministic mathematical program, and the constructions of the three delay-efficient algorithms, i.e., FQLA, LIFO-Backpressure and LIFO^p-Backpressure, indeed provide us with new insights into this important Lyapunov technique and will be helpful for future algorithm design in stochastic networks. In the

[1] One will see from the analysis in this chapter that, we can indeed prove the $[O(1/V), O(\log(V))]$ utility-delay tradeoffs for the proposed algorithms. However, for the ease of analysis and notation, we instead choose to prove our results for a slightly sub-optimal tradeoff.

following, we first provide some intuition of the deterministic mathematical program using the access point (AP) pricing problem considered in Chapter 2. We then turn to the general network setting and develop the general results there.

4.1 QLA and the deterministic problem

In this section, we first provide a simple example explaining the connection between the Lyapunov technique and the mathematical program. We then review the Quadratic Lyapunov functions based Algorithms (QLA) [GNT06] for solving the general stochastic problem defined in Chapter 3. After that, we define the *deterministic problem* and its dual problem. We also describe the ordinary subgradient method (OSM) that can be used to solve the dual. The dual problem and OSM will also be used later for our analysis of the steady state backlog behavior under QLA.

4.1.1 The pricing example

Here we first take a look at the relationship between QLA and a certain mathematical program via the AP pricing problem in Chapter 2.

Recall that the network state for the pricing system is $S(t) = [DM(t), CH(t)]$ and π_s is the steady state probability that $S(t) = s = [m, ch]$. Consider the following deterministic mathematical program:

$$\begin{aligned} \max: \quad & V \sum_s \pi_s \left[Z^{(s)} F(s, p^{(s)}) p^{(s)} - \text{cost}^{(s)} \right] \\ \text{s.t.} \quad & \sum_s \pi_s Z^{(s)} F(s, p^{(s)}) \leq \sum_s \pi_s \Phi(\text{cost}^{(s)}, s) \\ & p^{(s)} \in \mathcal{P}, \quad \forall\ s, \end{aligned} \tag{4.1}$$

$$Z^{(s)} \in \{0,1\}, \quad \forall\ s,$$

$$\text{cost}^{(s)} \in \mathcal{C} \quad \forall\ s.$$

Note that here we have written $F(m,p)$ and $\Phi(cost, ch)$ both as functions of s. The dual problem of (4.1) can easily be obtained to be: [2]

$$\min: \ \ g(\gamma), \quad \text{s.t. } \gamma \geq 0, \tag{4.2}$$

where:

$$g(\gamma) = \sup_{p^{(s)},\text{cost}^{(s)},Z^{(s)}} \sum_s \pi_s \Big\{ VZ^{(s)}F(s,p^{(s)})p^{(s)} - V\text{cost}^{(s)} \\ -\gamma\big[Z^{(s)}F(s,p^{(s)}) - \Phi(\text{cost}^{(s)},s)\big]\Big\}.$$

Now consider solving (4.2) using the Randomized Incremental Subgradient Method (RISM) [BNO03] as follows:

RISM: Initialize $\gamma(0) = 0$ and $\{\alpha^t \geq 0\}_{t=0}^{\infty}$; at iteration t, observe $\gamma(t)$, choose a random state $S(t) \in \mathcal{S}$ according to some probability law.

(1) If $S(t) = s$, find $p^{(s)}, \text{cost}^{(s)}, Z^{(s)}$ that solves the following:

$$\max: \quad VZ^{(s)}F(s,p^{(s)})p^{(s)} - V\text{cost}^{(s)} - \gamma(t)\big[Z^{(s)}F(s,p^{(s)}) - \Phi(\text{cost}^{(s)},s)\big] \tag{4.3}$$

$$\text{s.t.} \quad p^{(s)} \in \mathcal{P}, \text{cost}^{(s)} \in \mathcal{C}, Z^{(s)} \in \{0,1\}$$

(2) Using the $p^{(s)}, \text{cost}^{(s)}, Z^{(s)}$ found, update $\gamma(t)$ according to:

$$\gamma(t+1) = \max\Big[\gamma(t) - \alpha^t\Phi(\text{cost}^{(s)},s), 0\Big] + \alpha^t Z^{(s)}F(s,p^{(s)}).$$

Now it is easy to see that (4.3) is the same as:

$$\max: \Big[VZ^{(s)}F(s,p^{(s)})p^{(s)} - Z^{(s)}F(s,p^{(s)})\gamma(t)\Big] + \Big[\gamma(t)\Phi(\text{cost}^{(s)},s) - V\text{cost}^{(s)}\Big] \tag{4.4}$$

$$\text{s.t.} \quad p^{(s)} \in \mathcal{P}, Z^{(s)} \in \{0,1\}, \text{cost}^{(s)} \in \mathcal{C}.$$

[2] To facilitate reading, we have provided a very brief review on how to obtain the dual problem for a mathematical program in the Appendix .

We also recall that the PTSA algorithm in Chapter 2 works as follows:

Admission Control: Every slot t, the AP observes the current backlog $q(t)$ and the user demand $DM(t)$ and chooses the price $p(t)$ to be the solution of the following problem:

$$\begin{aligned} \max: \quad & VF(DM(t),p)p - 2q(t)F(DM(t),p) \\ s.t. \quad & p \in \mathcal{P}. \end{aligned} \tag{4.5}$$

If for all $p \in \mathcal{P}$ the resulting maximum is less than or equal to zero, the AP sends the "CLOSED" signal ($Z(t) = 0$) and does not accept new data. Else, it sets $Z(t) = 1$ and announces the chosen $p(t)$.

Cost/Transmission: Every slot t, the AP observes the current channel state $CH(t)$ and backlog $q(t)$ and chooses $\text{cost}(t)$ to be the solution of the following problem:

$$\begin{aligned} \max: \quad & 2q(t)\Phi(\text{cost}, CH(t)) - V \cdot \text{cost} \\ \text{s.t.} \quad & \text{cost} \in \mathcal{C}. \end{aligned} \tag{4.6}$$

The AP then sends out packets according to $\mu(t) = \Phi\left(\text{cost}(t), CH(t)\right)$.

Now let $S(t)$ in RISM be exactly the same as that in the AP pricing system, and let $\alpha^t = 1$ for all t. By comparing (4.4) to (4.5) and (4.6), we see that PTSA is indeed equivalent to RISM applied to the dual problem of (4.1) with $\gamma(t) = 2q(t)$.

In the following, we show that this connection between the deterministic mathematical program and QLA indeed holds for *any* network utility optimization problem that falls into the framework described in Chapter 3. As we will see, this connection not only enables a detailed analysis of the network backlog behavior under QLA, but also suggests a very natural way to guarantee an $O([\log(V)]^2)$ delay bound while achieving the $O(1/V)$ utility performance.

In the rest of this chapter, we study the case when $S(t)$ is i.i.d.. However, the QLA algorithm and the deterministic problem will *remain the same* under general ergodic $S(t)$ processes. The connection between the two also holds for general ergodic $S(t)$ processes. See Section 4.8 for further discussion.

4.1.2 The QLA algorithm

To solve the general stochastic problem in Chapter 3 with i.i.d. $S(t)$ using QLA, we first define a quadratic Lyapunov function $L(\boldsymbol{q}(t)) = \frac{1}{2}\sum_{j=1}^{r} q_j^2(t)$. We then define the one-slot conditional Lyapunov drift: $\Delta(\boldsymbol{q}(t)) = \mathbb{E}\big\{L(\boldsymbol{q}(t+1)) - L(\boldsymbol{q}(t)) \,|\, \boldsymbol{q}(t)\big\}$, where the expectation is taken over the random network state $S(t)$ and the possible random actions. From (3.5), we obtain the following:

$$\Delta(\boldsymbol{q}(t)) \leq B^2 - \mathbb{E}\big\{\sum_{j=1}^{r} q_j(t)\big[\mu_j(t) - A_j(t)\big] \,|\, \boldsymbol{q}(t)\big\}.$$

Now, adding to both sides the term $V\mathbb{E}\big\{f(t) \,|\, \boldsymbol{q}(t)\big\}$, where $V \geq 1$ is a scalar control variable, we obtain:

$$\Delta(\boldsymbol{q}(t)) + V\mathbb{E}\big\{f(t) \,|\, \boldsymbol{q}(t)\big\} \leq B^2 - \mathbb{E}\Big\{ - Vf(t) + \sum_{j=1}^{r} q_j(t)\big[\mu_j(t) - A_j(t)\big] \,|\, \boldsymbol{q}(t)\Big\}. \quad (4.7)$$

The QLA algorithm is then obtained by choosing an action x at every time slot t to *minimize the RHS of (4.7)* given $\boldsymbol{q}(t)$. Specifically, the QLA algorithm works as follows:

<u>*QLA:*</u> At every time slot t, observe the current network state $S(t)$ and the backlog $\boldsymbol{q}(t)$. If $S(t) = s_i$, choose $x^{(s_i)} \in \mathcal{X}^{(s_i)}$ that solves the following:

$$\begin{aligned} \max \quad & -Vf(s_i, x) + \sum_{j=1}^{r} q_j(t)\big[\mu_j(s_i, x) - A_j(s_i, x)\big] \quad (4.8) \\ \text{s.t.} \quad & x \in \mathcal{X}^{(s_i)}. \end{aligned}$$

Depending on the problem structure, (4.8) can usually be decomposed into separate parts that are easier to solve, e.g., [Nee06c], [HN10c]. Also, it can be shown, as in [GNT06] that when $S(t)$ is i.i.d.,

$$f_{av}^{QLA} = f_{av}^{*} + O(1/V), \quad \overline{q}^{QLA} = O(V), \tag{4.9}$$

where f_{av}^{QLA} and $\overline{q}^{QLA}$ are the expected average cost and the expected time average network backlog size under QLA, respectively.

It has been shown, e.g., in [Nee06c], [NML08], [GNT06], that an $O(V)$ network delay is typically incurred when an $O(1/V)$ close-to-optimal utility is achieved for network utility optimization problems. In this case, two recent papers [Nee07] and [Nee06b] use a more complicated algorithm design approach based on exponential Lyapunov functions to achieve within $O(1/V)$ of the optimal utility with only $O(\log(V))$ delay under i.i.d. network randomness. However, the exponential Lyapunov approach cannot be easily extended to the case when the network state is not i.i.d., whereas the methods we are going to present work under more general network state processes.

4.1.3 The deterministic problem

Consider *the deterministic problem* as follows:

$$\begin{aligned} \min: \quad & \mathcal{F}(\boldsymbol{x}) \triangleq V \sum_{s_i} \pi_{s_i} f(s_i, x^{(s_i)}) \\ \text{s.t.} \quad & \mathcal{A}_j(\boldsymbol{x}) \triangleq \sum_{s_i} \pi_{s_i} A_j(s_i, x^{(s_i)}) \leq \mathcal{B}_j(\boldsymbol{x}) \triangleq \sum_{s_i} \pi_{s_i} \mu_j(s_i, x^{(s_i)}) \quad \forall\, j \\ & x^{(s_i)} \in \mathcal{X}^{(s_i)} \quad \forall\, i = 1, 2, ..., M, \end{aligned} \tag{4.10}$$

where π_{s_i} is the probability of $S(t) = s_i$ and $\boldsymbol{x} = (x^{(s_1)}, ..., x^{(s_M)})^T$. The dual problem of (4.10) can be obtained as follows:

$$\begin{aligned} \max \quad & g(\boldsymbol{\gamma}) \\ \text{s.t.} \quad & \boldsymbol{\gamma} \succeq \mathbf{0}, \end{aligned} \tag{4.11}$$

where $g(\boldsymbol{\gamma})$ is called the dual function and is defined as:

$$\begin{aligned} g(\boldsymbol{\gamma}) = \inf_{x^{(s_i)} \in \mathcal{X}^{(s_i)}} \Big\{ & V \sum_{s_i} \pi_{s_i} f(s_i, x^{(s_i)}) \\ & + \sum_j \gamma_j \big[\sum_{s_i} \pi_{s_i} A_j(s_i, x^{(s_i)}) - \sum_{s_i} \pi_{s_i} \mu_j(s_i, x^{(s_i)}) \big] \Big\}. \end{aligned} \tag{4.12}$$

We note that $g(\boldsymbol{\gamma})$ can also be written in the following *separable* form, which is more useful for our later analysis.

$$g(\boldsymbol{\gamma}) = \inf_{x^{(s_i)} \in \mathcal{X}^{(s_i)}} \sum_{s_i} \pi_{s_i} \Big\{ V f(s_i, x^{(s_i)}) + \sum_j \gamma_j \big[A_j(s_i, x^{(s_i)}) - \mu_j(s_i, x^{(s_i)}) \big] \Big\}. \tag{4.13}$$

Here $\boldsymbol{\gamma} = (\gamma_1, ..., \gamma_r)^T$ is the *Lagrange multiplier* of (4.10). It is well known that $g(\boldsymbol{\gamma})$ in (4.12) is concave in the vector $\boldsymbol{\gamma}$, and hence the problem (4.11) can usually be solved efficiently, particularly when cost functions and rate functions are separable over different network components. It is also well known that in many situations, the optimal value of (4.11) is the same as the optimal value of (4.10), and in this case we say that there is no duality gap [BNO03]. (See Appendix A for more discussions.)

We note that the deterministic problem (4.10) is not necessarily convex, as the sets $\mathcal{X}^{(s_i)}$ are not necessarily convex, and the functions $f(s_i, \cdot)$, $A_j(s_i, \cdot)$ and $\mu_j(s_i, \cdot)$ are not necessarily convex. Therefore, there may be a duality gap between the deterministic problem (4.10) and its dual (4.11). Furthermore, solving the deterministic problem (4.10) may not solve the stochastic problem. This is because at every network state, the stochastic problem may require time sharing over more than one action, but the solution to the

deterministic problem gives only a fixed operating point per network state. However, we have shown in [HN10a] that the dual problem (4.11) gives the exact value of Vf^*_{av}, where f^*_{av} is the optimal time average cost for the stochastic problem, even if (4.10) is non-convex.

Among the many algorithms that can be used to solve (4.11), the following is the most common one (for performance, see [BNO03]). We call it the *ordinary subgradient method* (OSM):

<u>*OSM:*</u> Initialize $\boldsymbol{\gamma}(0)$; at every iteration t, observe $\boldsymbol{\gamma}(t)$,

1. Find $x^{(s_i)}_{\boldsymbol{\gamma}(t)} \in \mathcal{X}^{(s_i)}$ for $i \in \{1, ..., M\}$ that achieves the infimum of the right-hand side of (4.12).

2. Using the $\boldsymbol{x}_{\boldsymbol{\gamma}(t)} = (x^{(s_1)}_{\boldsymbol{\gamma}(t)}, x^{(s_2)}_{\boldsymbol{\gamma}(t)}, ..., x^{(s_M)}_{\boldsymbol{\gamma}(t)})^T$ found, update:

$$\gamma_j(t+1) = \max\left[\gamma_j(t) - \alpha^t \sum_{s_i} \pi_{s_i}\big[\mu_j(s_i, x^{(s_i)}_{\boldsymbol{\gamma}(t)}) - A_j(s_i, x^{(s_i)}_{\boldsymbol{\gamma}(t)})\big], 0\right]. \tag{4.14}$$

We use $x^{(s_i)}_{\boldsymbol{\gamma}(t)}$ to highlight its dependency on $\boldsymbol{\gamma}(t)$. $\alpha^t > 0$ is called the *step size* at iteration t. In the following, we always assume that $\alpha^t = 1$ when referring to OSM. Note that if there is only one network state, QLA and OSM will choose the same action given $\boldsymbol{\gamma}(t) = \boldsymbol{q}(t)$, and they differ only by (3.5) and (4.14). The term $\boldsymbol{G}_{\boldsymbol{\gamma}(t)} = (G_{\boldsymbol{\gamma}(t),1}, G_{\boldsymbol{\gamma}(t),2}, ..., G_{\boldsymbol{\gamma}(t),r})^T$, with:

$$\begin{aligned} G_{\boldsymbol{\gamma}(t),j} &= \mathcal{A}_j(\boldsymbol{x}_{\boldsymbol{\gamma}(t)}) - \mathcal{B}_j(\boldsymbol{x}_{\boldsymbol{\gamma}(t)}) \\ &= \sum_{s_i} \pi_{s_i}\big[-\mu_j(s_i, x^{(s_i)}_{\boldsymbol{\gamma}(t)}) + A_j(s_i, x^{(s_i)}_{\boldsymbol{\gamma}(t)})\big], \end{aligned} \tag{4.15}$$

is called the *subgradient* of $g(\boldsymbol{\gamma})$ at $\boldsymbol{\gamma}(t)$. It is well-known, e.g., in [BNO03], that for any other $\hat{\boldsymbol{\gamma}} \in \mathbb{R}^r$, we have:

$$(\hat{\boldsymbol{\gamma}} - \boldsymbol{\gamma}(t))^T \boldsymbol{G}_{\boldsymbol{\gamma}(t)} \geq g(\hat{\boldsymbol{\gamma}}) - g(\boldsymbol{\gamma}(t)). \tag{4.16}$$

Using $\|\boldsymbol{G}_{\boldsymbol{\gamma}(t)}\| \leq B$, we note that (4.16) also implies:

$$g(\hat{\boldsymbol{\gamma}}) - g(\boldsymbol{\gamma}) \leq B\|\hat{\boldsymbol{\gamma}} - \boldsymbol{\gamma}\| \quad \forall\, \hat{\boldsymbol{\gamma}}, \boldsymbol{\gamma} \in \mathbb{R}^r. \tag{4.17}$$

We are now ready to study the steady state behavior of $\boldsymbol{q}(t)$ under QLA. To simplify notation and highlight the scaling effect of the scalar V in QLA, we use the following notation:

1. We use $g_0(\boldsymbol{\gamma})$ and $\boldsymbol{\gamma}_0^*$ to denote the dual objective function and an optimal solution of (4.11) when $V = 1$; and use $g(\boldsymbol{\gamma})$ and $\boldsymbol{\gamma}_V^*$ (also called the optimal Lagrange multiplier) for their counterparts with general $V \geq 1$;

2. We use $x_{\boldsymbol{q}(t)}^{(s_i)}$ to denote an action chosen by QLA for a given $\boldsymbol{q}(t)$ and $S(t) = s_i$; and use $\boldsymbol{x}_{\boldsymbol{\gamma}(t)} = (x_{\boldsymbol{\gamma}(t)}^{(s_1)}, ..., x_{\boldsymbol{\gamma}(t)}^{(s_M)})^T$ to denote a solution chosen by OSM for a given $\boldsymbol{\gamma}(t)$.

To simplify the analysis, we assume the following throughout: [3]

Assumption 1. *$\boldsymbol{\gamma}_V^* = (\gamma_{V1}^*, ..., \gamma_{Vr}^*)^T$ is unique for all $V \geq 1$.*

Note that Assumption 1 is not very restrictive. In fact, it holds in many network utility optimization problems, e.g., [ES07]. In many cases, we also have $\boldsymbol{\gamma}_V^* \neq \mathbf{0}$. Moreover, for the assumption to hold for all $V \geq 1$, it suffices to have just $\boldsymbol{\gamma}_0^*$ being unique. This is shown in the following lemma.

Lemma 2. *$\boldsymbol{\gamma}_V^* = V\boldsymbol{\gamma}_0^*$.*

Proof. From (4.13) we see that:

$$g(\boldsymbol{\gamma})/V = \inf_{x^{(s_i)} \in \mathcal{X}^{(s_i)}} \sum_{s_i} \pi_{s_i} \Big\{ f(s_i, x^{(s_i)}) + \sum_j \hat{\gamma}_j \big[A_j(s_i, x^{(s_i)}) - \mu_j(s_i, x^{(s_i)})\big] \Big\},$$

[3] See Section 4.6 for discussions on the importance of this assumption.

where $\hat{\gamma}_j = \frac{\gamma_j}{V}$. The RHS is exactly $g_0(\hat{\boldsymbol{\gamma}})$, and so is maximized at $\hat{\boldsymbol{\gamma}} = \boldsymbol{\gamma}_0^*$. Thus $g(\boldsymbol{\gamma})$ is maximized at $V\boldsymbol{\gamma}_0^*$. □

4.2 Backlog vector behavior under QLA

In this section we study the backlog vector behavior under QLA of the stochastic problem. We first look at the case when $g_0(\boldsymbol{\gamma})$ is "locally polyhedral." We show that $\boldsymbol{q}(t)$ is mostly within $O(\log(V))$ distance from $\boldsymbol{\gamma}_V^*$ in this case, even when $S(t)$ evolves according to a more general time homogeneous Markovian process. We then consider the case when $g_0(\boldsymbol{\gamma})$ is "locally smooth," and show that $\boldsymbol{q}(t)$ is mostly within $O(\sqrt{V}\log(V))$ distance from $\boldsymbol{\gamma}_V^*$. As we will see, these two results also explain how QLA functions. The choices of these two types of $g_0(\boldsymbol{\gamma})$ functions are made based on their practical generality. See Section 4.2.3 for further discussion.

4.2.1 When $g_0(\cdot)$ is "locally polyhedral"

In this section, we study the backlog vector behavior under QLA for the case where $g_0(\boldsymbol{\gamma})$ is *locally polyhedral* with parameters ϵ, L, i.e., there exist $\epsilon, L > 0$, such that for all $\boldsymbol{\gamma} \succeq \mathbf{0}$ with $\|\boldsymbol{\gamma} - \boldsymbol{\gamma}_0^*\| < \epsilon$, the dual function $g_0(\boldsymbol{\gamma})$ satisfies:

$$g_0(\boldsymbol{\gamma}_0^*) \geq g_0(\boldsymbol{\gamma}) + L\|\boldsymbol{\gamma}_0^* - \boldsymbol{\gamma}\|. \tag{4.18}$$

We show that in this case, even if $S(t)$ is a general time homogeneous Markovian process, the backlog vector will mostly be within $O(\log(V))$ distance to $\boldsymbol{\gamma}_V^*$. Hence the same is also true when $S(t)$ is i.i.d.

To start, we assume *for this subsection* that $S(t)$ evolves according to a time homogeneous Markovian process. Now we define the following notation. Given t_0, define $\mathcal{T}_{s_i}(t_0, k)$ to be the set of slots at which $S(\tau) = s_i$ for $\tau \in [t_0, t_0 + k - 1]$. For a given $\nu > 0$, define the *convergent interval* T_ν [NMR05] for the $S(t)$ process to be the smallest number of slots such that for any t_0, regardless of past history, [4] we have:

$$\sum_{i=1}^{M} \left| \pi_{s_i} - \frac{\mathbb{E}\big\{ ||\mathcal{T}_{s_i}(t_0, T_\nu)|| \,|\, \mathcal{H}(t_0) \big\}}{T_\nu} \right| \leq \nu; \tag{4.19}$$

here $||\mathcal{T}_{s_i}(t_0, T_\nu)||$ is the cardinality of $\mathcal{T}_{s_i}(t_0, T_\nu)$, and $\mathcal{H}(t_0) = \{S(\tau)\}_{\tau=0}^{t_0-1}$ denotes the network state history up to time t_0. For any $\nu > 0$, such a T_ν must exist for any stationary ergodic processes with finite state space. Thus, T_ν exists for $S(t)$ in particular. When $S(t)$ is i.i.d. every slot, we have $T_\nu = 1$ for all $\nu \geq 0$, as $\mathbb{E}\big\{ ||\mathcal{T}_{s_i}(t_0, 1)|| \,|\, \mathcal{H}(t_0) \big\} = \pi_{s_i}$. Intuitively, T_ν represents the time needed for the process to reach its "near" steady state.

The following theorem summarizes the main results. Recall that B is defined in (3.4) as the upper bound of the magnitude change of $\boldsymbol{q}(t)$ in a slot, which is a function of the network size r and δ_{max}.

Theorem 5. *If $g_0(\boldsymbol{\gamma})$ is locally polyhedral with constants $\epsilon, L > 0$, independent of V, then under QLA,*

(a) There exist constants $\nu > 0$, $D \geq \eta > 0$, all independent of V, such that $D = D(\nu), \eta = \eta(\nu)$, and whenever $\|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\| \geq D$, we have: [5]

$$\mathbb{E}\big\{ \|\boldsymbol{q}(t + T_\nu) - \boldsymbol{\gamma}_V^*\| \,|\, \boldsymbol{q}(t) \big\} \leq \|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\| - \eta. \tag{4.20}$$

[4] In the Markov chain literature, this is called the *total variation mixing time* [LPW08].

[5] Note that if $\|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\| \geq D$, then $\|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\| \geq \eta$. Thus the right-hand side of (4.20) is nonnegative.

In particular, the constants ν, D and η that satisfy (4.20) can be chosen as follows: Choose ν as any constant such that $0 < \nu < L/B$. Then choose η as any value such that $0 < \eta < T_\nu(L - B\nu)$. Finally, choose D as: [6]

$$D = \max\left[\frac{(T_\nu^2 + T_\nu)B^2 - \eta^2}{2T_\nu(L - \frac{\eta}{T_\nu} - B\nu)}, \eta\right]. \tag{4.21}$$

(b) For the constants ν, D, η given in (a), there exist some constants $c^*, \beta^* > 0$, independent of V, such that:

$$\mathcal{P}(D, m) \leq c^* e^{-\beta^* m}, \tag{4.22}$$

where $\mathcal{P}(D, m)$ is defined as:

$$\mathcal{P}(D, m) \triangleq \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} Pr\{\|\boldsymbol{q}(\tau) - \boldsymbol{\gamma}_V^*\| > D + m\}. \tag{4.23}$$

Part (a) shows that when (4.18) is satisfied, if the current backlog vector is far away from $\boldsymbol{\gamma}_V^*$, the action of QLA will "push" it towards $\boldsymbol{\gamma}_V^*$. Specifically, the expected distance between them will decrease by η every time. Part (b) then shows that under such a "driving force," the queue vector will mostly be "attracted" to $\boldsymbol{\gamma}_V^*$. Indeed, if $m = \frac{\log(V)}{\beta^*}$, by (4.22) we have $\mathcal{P}(D, m) \leq \frac{c^*}{V}$. Also if a steady state distribution of $\|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\|$ exists under QLA, e.g., when $q_j(t)$ only takes integer values for all j, in which case $\boldsymbol{q}(t)$ is a discrete time Markov chain with countably infinite states and the limit of $\frac{1}{t}\sum_{\tau=0}^{t-1} \Pr\{\|\boldsymbol{q}(\tau) - \boldsymbol{\gamma}_V^*\| > D + m\}$ exists as $t \to \infty$, then one can replace $\mathcal{P}(D, m)$ with the steady state probability that $\boldsymbol{q}(t)$ deviates from $\boldsymbol{\gamma}_V^*$ by an amount of $D + m$, i.e., $\Pr\{\|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\| > D + m\}$.

Therefore, Theorem 5 can be viewed as showing that when (4.18) is satisfied, for a large V, the backlog $\boldsymbol{q}(t)$ under QLA will mostly be within $O(\log(V))$ distance from $\boldsymbol{\gamma}_V^*$.

[6] It can be seen from (4.17) and (4.18) that $B \geq L$. Thus $T_\nu B > T_\nu L > \eta$.

This implies that the average backlog will roughly be $\sum_j \gamma^*_{Vj}$, which is typically $\Theta(V)$ by Lemma 2. It also allows us to build FQLA upon QLA to "subtract out" roughly $\sum_j \gamma^*_{Vj}$ data from the network and reduce network delay. Theorem 5, together with Theorem 8 below, highlight the important fact that the steady state behavior of the network backlog vector $\boldsymbol{q}(t)$ is closely related to the structure of $g_0(\boldsymbol{\gamma})$. We note that (4.18) is not very restrictive. In fact, if $g_0(\boldsymbol{\gamma})$ is polyhedral (e.g., $\mathcal{X}^{(s_i)}$ is finite for all s_i), with a unique optimal solution $\boldsymbol{\gamma}^*_0 \succeq \mathbf{0}$, then (4.18) can be satisfied (see Section 4.3.6 for an example). To prove the theorem, we need the following lemma.

Lemma 3. *For any $\nu > 0$, under QLA, we have for all t,*

$$\begin{aligned}\mathbb{E}\big\{\|\boldsymbol{q}(t+T_\nu) - \boldsymbol{\gamma}^*_V\|^2 \mid \boldsymbol{q}(t)\big\} \quad \leq \quad & \|\boldsymbol{q}(t) - \boldsymbol{\gamma}^*_V\|^2 + (T_\nu^2 + T_\nu)B^2 \\ & -2T_\nu\big(g(\boldsymbol{\gamma}^*_V) - g(\boldsymbol{q}(t))\big) + 2T_\nu\nu B\|\boldsymbol{\gamma}^*_V - \boldsymbol{q}(t)\|.\end{aligned} \tag{4.24}$$

Proof. See Section 4.10.1. □

We now use Lemma 3 to prove Theorem 5.

Proof. (Theorem 5) Part (a): We first show that if (4.18) holds for $g_0(\boldsymbol{\gamma})$ with L, then it also holds for $g(\boldsymbol{\gamma})$ with the same L. To this end, suppose (4.18) holds for $g_0(\boldsymbol{\gamma})$ for all $\boldsymbol{\gamma}$ satisfying $\|\boldsymbol{\gamma} - \boldsymbol{\gamma}^*_0\| < \epsilon$. Then for any $\boldsymbol{\gamma} \succeq \mathbf{0}$ such that $\|\boldsymbol{\gamma} - \boldsymbol{\gamma}^*_V\| < \epsilon V$, we have $\|\boldsymbol{\gamma}/V - \boldsymbol{\gamma}^*_0\| < \epsilon$, hence:

$$g_0(\boldsymbol{\gamma}^*_0) \geq g_0(\boldsymbol{\gamma}/V) + L\|\boldsymbol{\gamma}^*_0 - \boldsymbol{\gamma}/V\|.$$

Multiplying both sides by V, we get:

$$Vg_0(\boldsymbol{\gamma}^*_0) \geq Vg_0(\boldsymbol{\gamma}/V) + LV\|\boldsymbol{\gamma}^*_0 - \boldsymbol{\gamma}/V\|.$$

Now using $\boldsymbol{\gamma}_V^* = V\boldsymbol{\gamma}_0^*$ and $g(\boldsymbol{\gamma}) = Vg_0(\boldsymbol{\gamma}/V)$, we have for all $\|\boldsymbol{\gamma} - \boldsymbol{\gamma}_V^*\| < \epsilon V$:

$$g(\boldsymbol{\gamma}_V^*) \geq g(\boldsymbol{\gamma}) + L\|\boldsymbol{\gamma}_V^* - \boldsymbol{\gamma}\|. \tag{4.25}$$

Since $g(\boldsymbol{\gamma})$ is concave, we see that (4.25) indeed holds for all $\boldsymbol{\gamma} \succeq \mathbf{0}$. Now for a given $\eta > 0$, if:

$$(T_\nu^2 + T_\nu)B^2 - 2T_\nu\big(g(\boldsymbol{\gamma}_V^*) - g(\boldsymbol{q}(t))\big) + 2T_\nu\nu B\|\boldsymbol{\gamma}_V^* - \boldsymbol{q}(t)\| \\ \leq \eta^2 - 2\eta\|\boldsymbol{\gamma}_V^* - \boldsymbol{q}(t)\|, \tag{4.26}$$

then by (4.24), we have:

$$\mathbb{E}\big\{\|\boldsymbol{q}(t+T_\nu) - \boldsymbol{\gamma}_V^*\|^2 \,|\, \boldsymbol{q}(t)\big\} \leq (\|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\| - \eta)^2,$$

which then by Jensen's inequality implies:

$$(\mathbb{E}\big\{\|\boldsymbol{q}(t+T_\nu) - \boldsymbol{\gamma}_V^*\| \,|\, \boldsymbol{q}(t)\big\})^2 \leq (\|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\| - \eta)^2.$$

Thus (4.20) follows whenever (4.26) holds and $\|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\| \geq \eta$. It suffices to choose D and η such that $D \geq \eta$ and that (4.26) holds whenever $\|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\| \geq D$. Now note that (4.26) can be rewritten as the following inequalty:

$$g(\boldsymbol{\gamma}_V^*) \geq g(\boldsymbol{q}(t)) + (B\nu + \frac{\eta}{T_\nu})\|\boldsymbol{\gamma}_V^* - \boldsymbol{q}(t)\| + \mathcal{Y}, \tag{4.27}$$

where $\mathcal{Y} = \frac{(T_\nu^2+T_\nu)B^2-\eta^2}{2T_\nu}$. Choose any $\nu > 0$ independent of V such that $B\nu < L$ and choose $\eta \in (0, T_\nu(L - B\nu))$. By (4.25), if:

$$L\|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\| \geq (B\nu + \frac{\eta}{T_\nu})\|\boldsymbol{\gamma}_V^* - \boldsymbol{q}(t)\| + \mathcal{Y}, \tag{4.28}$$

then (4.27) holds. Now choosing D as defined in (4.21), we see that if $\|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\| \geq D$, then (4.28) holds, which implies (4.27), and equivalently (4.26). We also have $D \geq \eta$, hence (4.20) holds.

Part (b): Now we show that (4.20) implies (4.22). Choose constants ν, D and η that are independent of V in (a). Denoting $Y(t) = \|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\|$, we see then whenever

$Y(t) \geq D$, we have $\mathbb{E}\big\{Y(t+T_\nu) - Y(t) \,|\, \boldsymbol{q}(t)\big\} \leq -\eta$. It is also easy to see that $|Y(t+T_\nu) - Y(t)| \leq T_\nu B$, as B is defined in (3.4) as the upper bound of the magnitude change of $\boldsymbol{q}(t)$ in a slot. Define $\tilde{Y}(t) = \max\big[Y(t) - D, 0\big]$. We see that whenever $\tilde{Y}(t) \geq T_\nu B$, we have:

$$\mathbb{E}\big\{\tilde{Y}(t+T_\nu) - \tilde{Y}(t) \,|\, \boldsymbol{q}(t)\big\} = \mathbb{E}\big\{Y(t+T_\nu) - Y(t) \,|\, \boldsymbol{q}(t)\big\} \leq -\eta. \tag{4.29}$$

Now define a Lyapunov function of $\tilde{Y}(t)$ to be $L(\tilde{Y}(t)) = e^{w\tilde{Y}(t)}$ with some $w > 0$, and define the T_ν-slot conditional *drift* to be:

$$\begin{aligned} \Delta_{T_\nu}(\tilde{Y}(t)) &\triangleq \mathbb{E}\big\{L(\tilde{Y}(t+T_\nu)) - L(\tilde{Y}(t)) \,|\, \boldsymbol{q}(t)\big\} \\ &= \mathbb{E}\big\{e^{w\tilde{Y}(t+T_\nu)} - e^{w\tilde{Y}(t)} \,|\, \boldsymbol{q}(t)\big\}. \end{aligned} \tag{4.30}$$

It is shown in Section 4.10.2 that by choosing $w = \frac{\eta}{T_\nu^2 B^2 + T_\nu B\eta/3}$, we have for all $\tilde{Y}(t) \geq 0$:

$$\Delta_{T_\nu}(\tilde{Y}(t)) \leq e^{2wT_\nu B} - \frac{w\eta}{2} e^{w\tilde{Y}(t)}. \tag{4.31}$$

Taking expectation on both sides over $\boldsymbol{q}(t)$, we have:

$$\mathbb{E}\big\{e^{w\tilde{Y}(t+T_\nu)} - e^{w\tilde{Y}(t)}\big\} \leq e^{2wT_\nu B} - \frac{w\eta}{2}\mathbb{E}\big\{e^{w\tilde{Y}(t)}\big\}. \tag{4.32}$$

Summing (4.32) over $t \in \{t_0, t_0 + T_\nu, ..., t_0 + (N-1)T_\nu\}$ for some $t_0 \in \{0, 1, ..., T_\nu - 1\}$, we have:

$$\mathbb{E}\big\{e^{w\tilde{Y}(t_0+NT_\nu)} - e^{w\tilde{Y}(t_0)}\big\} \leq Ne^{2wT_\nu B} - \sum_{j=0}^{N-1} \frac{w\eta}{2}\mathbb{E}\big\{e^{w\tilde{Y}(t_0+jT_\nu)}\big\}.$$

Rearranging the terms, we have:

$$\sum_{j=0}^{N-1} \frac{w\eta}{2}\mathbb{E}\big\{e^{w\tilde{Y}(t_0+jT_\nu)}\big\} \leq Ne^{2wT_\nu B} + \mathbb{E}\big\{e^{w\tilde{Y}(t_0)}\big\}.$$

Summing the above over $t_0 \in \{0, 1, ..., T_\nu - 1\}$, we obtain:

$$\sum_{t=0}^{NT_\nu-1} \frac{w\eta}{2}\mathbb{E}\big\{e^{w\tilde{Y}(t)}\big\} \leq NT_\nu e^{2wT_\nu B} + \sum_{t_0=0}^{T_\nu-1} \mathbb{E}\big\{e^{w\tilde{Y}(t_0)}\big\}.$$

Dividing both sides with NT_ν, we obtain:

$$\frac{1}{NT_\nu}\sum_{t=0}^{NT_\nu-1}\frac{w\eta}{2}\mathbb{E}\{e^{w\tilde{Y}(t)}\} \le e^{2wT_\nu B} + \frac{1}{NT_\nu}\sum_{t_0=0}^{T_\nu-1}\mathbb{E}\{e^{w\tilde{Y}(t_0)}\}. \tag{4.33}$$

Taking the limsup as N goes to infinity, which is equivalent to letting $t = NT_\nu$ go to infinity, we obtain:

$$\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\frac{w\eta}{2}\mathbb{E}\{e^{w\tilde{Y}(\tau)}\} \quad \le \quad e^{2wT_\nu B}. \tag{4.34}$$

Using the fact that $\mathbb{E}\{e^{w\tilde{Y}(\tau)}\} \ge e^{wm}\Pr\{\tilde{Y}(\tau) > m\}$, we get:

$$\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\frac{w\eta}{2}e^{wm}\Pr\{\tilde{Y}(\tau) > m\} \quad \le \quad e^{2wT_\nu B}. \tag{4.35}$$

Plug in $w = \frac{\eta}{T_\nu^2B^2+T_\nu B\eta/3}$ and use the definition of $\tilde{Y}(t)$:

$$\begin{aligned}\mathcal{P}(D,m) &\le \frac{2e^{2wT_\nu B}}{w\eta}e^{-wm} \\ &= \frac{2(T_\nu^2B^2+T_\nu B\eta/3)e^{\frac{2\eta}{T_\nu B+\eta/3}}}{\eta^2}e^{-\frac{\eta m}{T_\nu^2B^2+T_\nu B\eta/3}},\end{aligned} \tag{4.36}$$

where $\mathcal{P}(D,m)$ is defined in (4.23). Therefore (4.22) holds with:

$$\begin{aligned}c^* &= \frac{2(T_\nu^2B^2+T_\nu B\eta/3)e^{\frac{2\eta}{T_\nu B+\eta/3}}}{\eta^2}, \\ \beta^* &= \frac{\eta}{T_\nu^2B^2+T_\nu B\eta/3}.\end{aligned} \tag{4.37}$$

It is easy to see that c^* and β^* are both independent of V. □

Our approach in deriving the results in Part (b) of Theorem 5 (and Theorem 8 below) and the approach in [CL] are both based on Taylor expansion. However, [CL] can be viewed as deriving the results from the moment generating function of the target variable. Our approach can be viewed as using the difference of the moment generating functions of the intermediate variables. Our approach can also be applied to deriving contraction bounds for martingales [Dur05].

Note from (4.33) and (4.34) that Theorem 5 indeed holds for any finite $\boldsymbol{q}(0)$. We later use this fact to prove the performance of FQLA. The following theorem is a special case of Theorem 5 and gives a more direct illustration of Theorem 5. Recall that $\mathcal{P}(D,m)$ is defined in (4.23). Define:

$$\mathcal{P}^{(r)}(D,m) \triangleq \limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \Pr\{||\boldsymbol{q}(\tau)-\boldsymbol{\gamma}_V^*||_\infty > D+m\} \tag{4.38}$$

$$= \limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \Pr\{\exists\, j, |q_j(\tau)-\gamma_{Vj}^*| > D+m\} \tag{4.39}$$

Theorem 6. *If the condition in Theorem 5 holds and $S(t)$ is i.i.d., then under QLA, for any $c>0$:*

$$\mathcal{P}(D_1, cK_1\log(V)) \leq \frac{c_1^*}{V^c}, \tag{4.40}$$

$$\mathcal{P}^{(r)}(D_1, cK_1\log(V)) \leq \frac{c_1^*}{V^c}. \tag{4.41}$$

where $D_1 = \frac{2B^2}{L}+\frac{L}{4}$, $K_1 = \frac{B^2+BL/6}{L/2}$ and $c_1^* = \frac{8(B^2+BL/6)e^{\frac{L}{B+L/6}}}{L^2}$.

Proof. First we note that when $S(t)$ is i.i.d., we have $T_\nu = 1$ for $\nu = 0$. Now choose $\nu = 0$, $T_\nu = 1$ and $\eta = L/2$, then we see from (4.21) that

$$D = \max\left[\frac{2B^2-L^2/4}{L}, \frac{L}{2}\right] \leq \frac{2B^2}{L}+\frac{L}{4}.$$

Now by (4.37) we see that (4.22) holds with $c^* = c_1^*$ and $\beta^* = \frac{L/2}{B^2+BL/6}$. Thus by taking $D_1 = \frac{2B^2}{L}+\frac{L}{4}$, we have:

$$\mathcal{P}(D_1, cK_1\log(V)) \leq c^* e^{-cK_1\beta^*\log(V)}$$

$$= c_1^* e^{-c\log(V)},$$

where the last step follows since $\beta^* K_1 = 1$. Thus (4.40) follows. Equation (4.41) follows from (4.40) by using the fact that for any constant ζ, the events $\mathcal{E}_1 = \{||\boldsymbol{q}(\tau)-\boldsymbol{\gamma}_V^*||_\infty > \zeta\}$

and $\mathcal{E}_2 = \{\|\boldsymbol{q}(\tau) - \boldsymbol{\gamma}_V^*\| > \zeta\}$ satisfy $\mathcal{E}_1 \subset \mathcal{E}_2$. Thus: $\Pr\{||\boldsymbol{q}(\tau) - \boldsymbol{\gamma}_V^*||_\infty > \zeta\} \leq \Pr\{\|\boldsymbol{q}(\tau) - \boldsymbol{\gamma}_V^*\| > \zeta\}$. □

Theorem 6 can be viewed as showing that for a large V, the probability for $q_j(t)$ to deviate from the j^{th} component of $\boldsymbol{\gamma}_V^*$ is exponentially decreasing in the distance. Thus it rarely deviates from γ_{Vj}^* by more than $\Theta(\log(V))$ distance. Note that one can similarly prove the following theorem for OSM:

Theorem 7. *If the condition in Theorem 5 holds, then there exist positive constants* $D = \Theta(1)$ *and* $\eta = \Theta(1)$*, i.e, independent of* V*, such that, under OSM, if* $\|\boldsymbol{\gamma}(t) - \boldsymbol{\gamma}_V^*\| \geq D$*,*

$$\|\boldsymbol{\gamma}(t+1) - \boldsymbol{\gamma}_V^*\| \leq \|\boldsymbol{\gamma}(t) - \boldsymbol{\gamma}_V^*\| - \eta. \tag{4.42}$$

Proof. It is easy to show that under OSM, Lemma 3 holds with $\nu = 0$, $T_\nu = 1$ and without the expectation. Indeed, by (4.14), (4.15) and Lemma 5 in Section 4.10.1, we have:

$$\|\boldsymbol{\gamma}(t+1) - \boldsymbol{\gamma}_V^*\|^2 \quad \leq \quad \|\boldsymbol{\gamma}(t) - \boldsymbol{\gamma}_V^*\|^2 + 2B^2 - 2(\boldsymbol{\gamma}_V^* - \boldsymbol{\gamma}(t))^T \boldsymbol{G}_{\boldsymbol{\gamma}(t)}.$$

Now by (4.16) we have: $(\boldsymbol{\gamma}_V^* - \boldsymbol{\gamma}(t))^T \boldsymbol{G}_{\boldsymbol{\gamma}(t)} \geq g(\boldsymbol{\gamma}_V^*) - g(\boldsymbol{\gamma}(t))$. Plug this into the above equation, we obtain:

$$\|\boldsymbol{\gamma}(t+1) - \boldsymbol{\gamma}_V^*\|^2 \quad \leq \quad \|\boldsymbol{\gamma}(t) - \boldsymbol{\gamma}_V^*\|^2 + 2B^2 - 2\big(g(\boldsymbol{\gamma}_V^*) - g(\boldsymbol{\gamma}(t))\big).$$

The theorem then follows by using the same argument as in the proof of Theorem 5. □

Therefore, when there is a single network state, in which case QLA is equivalent to OSM, we see that given (4.18), the backlog process converges to a neighborhood of size $\Theta(1)$ around $\boldsymbol{\gamma}_V^*$.

4.2.2 When $g_0(\cdot)$ is "locally smooth"

In this section, we consider the backlog behavior under QLA, for the case where the dual function $g_0(\boldsymbol{\gamma})$ is "locally smooth" at $\boldsymbol{\gamma}_0^*$. Specifically, we say that the function $g_0(\boldsymbol{\gamma})$ is *locally smooth* at $\boldsymbol{\gamma}_0^*$ with parameters $\varepsilon, L > 0$ if for all $\boldsymbol{\gamma} \succeq \mathbf{0}$ such that $\|\boldsymbol{\gamma} - \boldsymbol{\gamma}_0^*\| < \varepsilon$, we have:

$$g_0(\boldsymbol{\gamma}_0^*) \geq g_0(\boldsymbol{\gamma}) + L\|\boldsymbol{\gamma} - \boldsymbol{\gamma}_0^*\|^2, \tag{4.43}$$

This condition contains the case when $g_0(\boldsymbol{\gamma})$ is twice differentiable with $\nabla g(\boldsymbol{\gamma}_0^*) = \mathbf{0}$ and $\boldsymbol{a}^T \nabla^2 g(\boldsymbol{\gamma}) \boldsymbol{a} \leq -2L\|\boldsymbol{a}\|^2$, $\forall \boldsymbol{a}$ for any $\boldsymbol{\gamma}$ with $\|\boldsymbol{\gamma}_0^* - \boldsymbol{\gamma}\| < \varepsilon$. Such a case usually occurs when the sets $\mathcal{X}^{(s_i)}, i = 1, ..., M$ are convex, i.e., a "continuous" set of actions are available. Notice that (4.43) is a looser condition than (4.18) in the neighborhood of $\boldsymbol{\gamma}_0^*$. As we will see, such structural difference of $g_0(\boldsymbol{\gamma})$ in the neighborhood of $\boldsymbol{\gamma}_0^*$ greatly affects the behavior of backlogs under QLA.

Theorem 8. *If $g_0(\boldsymbol{\gamma})$ is locally smooth at $\boldsymbol{\gamma}_0^*$ with parameters $\varepsilon, L > 0$, independent of V, then under QLA with a sufficiently large V, we have:*

(a) There exists $D = \Theta(\sqrt{V})$ such that whenever $\|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\| \geq D$, we have:

$$\mathbb{E}\big\{\|\boldsymbol{q}(t+1) - \boldsymbol{\gamma}_V^*\| \mid \boldsymbol{q}(t)\big\} \leq \|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\| - \frac{1}{\sqrt{V}}. \tag{4.44}$$

(b) $\mathcal{P}(D, m) \leq c^* e^{-\beta^* m}$, where $\mathcal{P}(D, m)$ is defined in (4.23), $c^* = \Theta(V)$ and $\beta^* = \Theta(1/\sqrt{V})$.

Theorem 8 can be viewed as showing that, when $g_0(\boldsymbol{\gamma})$ is locally smooth at $\boldsymbol{\gamma}_0^*$, the backlog vector will mostly be within $O(\sqrt{V}\log(V))$ distance from $\boldsymbol{\gamma}_V^*$. This contrasts with Theorem 5, which shows that the backlog will mostly be within $O(\log(V))$ distance

from $\boldsymbol{\gamma}_V^*$. Intuitively, this is due to the fact that under local smoothness, the drift towards $\boldsymbol{\gamma}_V^*$ is smaller as $\boldsymbol{q}(t)$ gets closer to $\boldsymbol{\gamma}_V^*$, hence a $\Theta(\sqrt{V})$ distance is needed to guarantee a drift of size $\Theta(1/\sqrt{V})$; whereas under (4.18), any nonzero $\Theta(1)$ deviation from $\boldsymbol{\gamma}_V^*$ roughly generates a drift of size $\Theta(1)$ towards $\boldsymbol{\gamma}_V^*$, ensuring that the backlog stays within $O(\log(V))$ distance from $\boldsymbol{\gamma}_V^*$.

To prove Theorem 8, we need the following corollary of Lemma 3.

Corollary 2. *If $S(t)$ is i.i.d., then under QLA,*

$$\mathbb{E}\big\{\|\boldsymbol{q}(t+1)-\boldsymbol{\gamma}_V^*\|^2 \mid \boldsymbol{q}(t)\big\} \le \|\boldsymbol{q}(t)-\boldsymbol{\gamma}_V^*\|^2 + 2B^2 - 2\big(g(\boldsymbol{\gamma}_V^*) - g(\boldsymbol{q}(t))\big).$$

Proof. When $S(t)$ is i.i.d., we have $T_\nu = 1$ for $\nu = 0$. □

Proof. (Theorem 8) Part (a): We first see that for any $\boldsymbol{\gamma}$ with $\|\boldsymbol{\gamma}-\boldsymbol{\gamma}_V^*\| < \varepsilon V$, we have $\|\boldsymbol{\gamma}/V - \boldsymbol{\gamma}_0^*\| < \varepsilon$. Therefore,

$$g_0(\boldsymbol{\gamma}_0^*) \ge g_0(\boldsymbol{\gamma}/V) + L\|\boldsymbol{\gamma}/V - \boldsymbol{\gamma}_0^*\|^2. \tag{4.45}$$

Multiplying both sides with V, we get:

$$g(\boldsymbol{\gamma}_V^*) \ge g(\boldsymbol{\gamma}) + \frac{L}{V}\|\boldsymbol{\gamma}-\boldsymbol{\gamma}_V^*\|^2. \tag{4.46}$$

Similar to the proof of Theorem 5 and by Corollary 2, we see that for (4.44) to hold, we only need $\|\boldsymbol{q}(t)-\boldsymbol{\gamma}_V^*\| \ge \frac{1}{\sqrt{V}}$ and:

$$2B^2 - 2\big(g(\boldsymbol{\gamma}_V^*) - g(\boldsymbol{q}(t))\big) \le \frac{1}{V} - \frac{2}{\sqrt{V}}\|\boldsymbol{q}(t)-\boldsymbol{\gamma}_V^*\|,$$

which can be rewritten as:

$$g(\boldsymbol{\gamma}_V^*) \ge g(\boldsymbol{q}(t))\big) + \frac{1}{\sqrt{V}}\|\boldsymbol{q}(t)-\boldsymbol{\gamma}_V^*\| + \frac{2B^2 - \frac{1}{V}}{2}. \tag{4.47}$$

By (4.46), we see that for (4.47) to hold, it suffices to have:

$$\frac{L}{V}\|\boldsymbol{q}(t)-\boldsymbol{\gamma}_V^*\|^2 \ge \frac{1}{\sqrt{V}}\|\boldsymbol{q}(t)-\boldsymbol{\gamma}_V^*\| + B^2. \tag{4.48}$$

By solving the quadratic inequality (4.48), we see that (4.48) holds whenever:

$$\|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\| \geq \frac{\frac{1}{\sqrt{V}} + \sqrt{\frac{1}{V} + \frac{4B^2L}{V}}}{2L/V} = \frac{\sqrt{V} + \sqrt{V + 4B^2LV}}{2L}.$$

Denote $D = \frac{\sqrt{V}+\sqrt{V+4B^2LV}}{2L}$. We see that when V is large, (4.44) holds for any $\boldsymbol{q}(t)$ with $D \leq \|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\| < \varepsilon V$. Now since $g(\boldsymbol{\gamma})$ is concave, one can show that (4.47) holds for all $\|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\| \geq D$. Hence (4.44) holds for all $\|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\| \geq D$, proving Part (a).

Part (b): By an argument that is similar to the proof of Theorem 5, we see that Part (b) follows with: $\beta^* = \frac{3}{3\sqrt{V}B^2+B}$ and $c^* = 2(VB^2 + B\sqrt{V}/3)e^{\frac{6}{3B\sqrt{V}+1}}$. □

Notice in this case we can also prove a similar result as Theorem 7 for OSM, with the only difference that $D = \Theta(\sqrt{V})$.

4.2.3 Discussion of the choices of $g_0(\gamma)$

In our analysis, we have focused only on the dual function $g_0(\boldsymbol{\gamma})$ being either locally polyhedral or locally smooth. These choices are made based on their practical generality. To be more precise, assume that without loss of generality that there is only one network state and the set of feasible actions is a compact subset of $\mathbb{R}^n$. In practice, this action set is usually finite due to digitization. Thus we see from the definition of $g_0(\boldsymbol{\gamma})$ that an action, if chosen given a Lagrange multiplier $\boldsymbol{\gamma}$, remains the chosen action for a range of Lagrange multipliers around $\boldsymbol{\gamma}$. Hence $g_0(\boldsymbol{\gamma})$ is polyhedral in this case. Now as the granularity of the action sets becomes finer and finer, we can expect the dual function $g_0(\boldsymbol{\gamma})$ to be "smoother and smoother," in the sense that moving from one action to another close-by action does not affect the value of $g_0(\boldsymbol{\gamma})$ by much. Eventually when the granularity is fine enough, the action set can be viewed as convex. Now if the optimal

network performance is achieved at some action not at the boundary of the action set, then we see that in a small neighborhood around $\boldsymbol{\gamma}_V^*$, we usually have a locally smooth $g_0(\boldsymbol{\gamma})$ function. Further note that in both cases, the structure of $g_0(\boldsymbol{\gamma})$ is independent of V. Hence the conditions in Theorem 5 and 8 can typically be satisfied in practice.

Though we have focused on the cases when $g(\cdot)$ is either polyhedral or smooth, our analysis approach can also be applied to dual functions satisfying other conditions. For instance,

$$g_0(\boldsymbol{\gamma}_0^*) \geq g_0(\boldsymbol{\gamma}) + L\|\boldsymbol{\gamma} - \boldsymbol{\gamma}_0^*\|^p, \tag{4.49}$$

with $p \geq 1$. In this case, we get a sufficient condition similar to (4.48):

$$\frac{L}{V^p}\|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\|^p \geq \frac{1}{V^{p/2}}\|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\| + B^2. \tag{4.50}$$

We can similarly obtain a value D, which is a function of p and V, and prove similar exponential attraction results. For some values of p, we may not be able to get closed form solutions. However, (4.50) will still be useful in practice because we can use it to compute numerical solutions.

4.2.4 The importance of the ϵ-slack condition

Throughout the above derivation, we have assumed that there exists a randomized policy that achieves the ϵ-slackness for our problem. This assumption is crucial to all the results in this thesis. Indeed, the ϵ-slack assumption guarantees that the optimal Lagrange multiplier $\boldsymbol{\gamma}_0^*$ has bounded magnitude, see, e.g., Page 524 in [BNO03]. In this case, Lemma 2 ensures that $\|\boldsymbol{\gamma}_V^*\| = \Theta(V)$, which subsequently implies that the time average backlog in the network is $\Theta(V)$. Without this slackness condition, the value $\boldsymbol{\gamma}_V^* = \infty$ will

also be a maximizer of the dual function $g(\boldsymbol{\gamma})$. [7] Thus, it is possible that the network congestion will go unbounded under QLA.

4.2.5 Implications of Theorem 5 and 8

Consider the following simple problem: an operator operates a single queue and tries to support a Bernoulli arrival, i.e., either 1 or 0 packet arrives every slot, with rate $\lambda = 0.5$ (the rate may be unknown to the operator) with minimum energy expenditure. The channel is time-invariant. The rate-power curve over the channel is given by: $\mu(t) = \log(1 + P(t))$, where $P(t)$ is the allocated power at time t. Thus to obtain a rate of $\mu(t)$, we need $P(t) = e^{\mu(t)} - 1$. In every time slot, the operator decides how much power to allocate and serves the queue at the corresponding rate, with the goal of minimizing the time average power consumption subject to queue stability. Let Φ denote the time average energy expenditure incurred by the optimal policy. It can be shown that $\Phi = e^{0.5} - 1$. To see this, note that the optimal strategy is to allocate power such that the average service rate is *exactly* equal to the arrival rate. Thus, by the convexity of the power-rate curve, the optimal strategy is to use $P(t) = e^{0.5} - 1$ for all time.

Now we look at the deterministic problem:

$$\min: \ V(e^{\mu} - 1), \qquad s.t.: \ 0.5 \leq \mu$$

In this case, the dual function is given by: $g(\gamma) = \inf_{\mu} \left\{ V(e^{\mu} - 1) + \gamma(0.5 - \mu) \right\}$. Hence by the KKT conditions [BNO03] one obtains that $\gamma_V^* = Ve^{0.5}$ and the optimal policy is to serve the queue at the constant rate $\mu^* = 0.5$. Suppose now that QLA is applied to

[7] From a queueing network point of view, this corresponds to the case when the arrival vector is on the boundary of the capacity region of the network.

the problem. Then at slot t, if $q(t) = q$, QLA chooses the power to achieve the rate $\mu(t)$ such that ($[a]^+ = \max[a, 0]$):

$$\mu(t) \in \arg\min\{V(e^{\mu} - 1) + q(0.5 - \mu)\} = \left[\log(\frac{q}{V})\right]^+. \tag{4.51}$$

which incurs an instantanous power consumption of $P(t) \approx \frac{q(t)}{V} - 1$. In this case, it can be shown that Theorem 8 applies. Thus for most of the time $q(t) \in [\gamma_V^* - \sqrt{V}, \gamma_V^* + \sqrt{V}]$, i.e., $q(t) \in [Ve^{0.5} - \sqrt{V}, Ve^{0.5} + \sqrt{V}]$. Hence it is almost always the case that: $\log(e^{0.5} - \frac{1}{\sqrt{V}}) \leq \mu(t) \leq \log(e^{0.5} + \frac{1}{\sqrt{V}})$, which implies: $0.5 - \frac{1}{\sqrt{V}} \leq \mu(t) \leq 0.5 + \frac{1}{\sqrt{V}}$. Thus by a similar argument as in [Nee07], one can show that $\overline{P} \leq \Phi + O(1/V)$, where $\overline{P}$ is the average power consumption.

Now consider the case when we can only choose to operate at $\mu \in \{0, \frac{1}{4}, \frac{3}{4}, 1\}$, with the corresponding power consumptions being: $P \in \{0, e^{\frac{1}{4}} - 1, e^{\frac{3}{4}} - 1, e - 1\}$. One can similarly obtain $\Phi = \frac{1}{2}(e^{\frac{3}{4}} + e^{\frac{1}{4}})$ and $\gamma_V^* = 2V(e^{\frac{3}{4}} - e^{\frac{1}{4}})$. In this case, Φ is achieved by time sharing the two rates $\{\frac{1}{4}, \frac{3}{4}\}$ with equal portion of time. It can also be shown that Theorem 5 applies in this case. Thus we see that under QLA, $q(t)$ is mostly within $\log(V)$ distance to γ_V^*. Hence by (4.51), we see that QLA almost always chooses between the two rates $\{\frac{1}{4}, \frac{3}{4}\}$, and uses them with almost equal frequencies. Hence QLA is also able to achieve $\overline{P} = \Phi + O(1/V)$ in this case.

The above argument can be generalized to many stochastic network optimization problems. Thus, we see that Theorem 5 and 8 not only provide us with probabilistic deviation bounds of $\boldsymbol{q}(t)$ from $\boldsymbol{\gamma}_V^*$, but also help to explain why QLA is able to achieve the desired utility performance: *under QLA, $\boldsymbol{q}(t)$ always stays close to $\boldsymbol{\gamma}_V^*$, hence the chosen action is always close to the set of optimal actions.*

4.2.6 More backlog bounds when there is a single queue

We note that when there is a single queue, i.e., $r = 1$, in the network, e.g., [HN10c], one can obtain *deterministic* upper and lower bounds of the backlog value under *arbitrary* network state distribution and the way $S(t)$ evolves. We also note that in this single queue case, one can also obtain exponential attraction results similar to Theorem 5 and 8, *without* assuming the locally polyhedral or locally smooth conditions. For details, see [HN11a].

4.3 The FQLA algorithm

In this section, we propose a family of *Fast Quadratic Lyapunov based Algorithms* (FQLA) for general stochastic network optimization problems. We first provide an example to illustrate the idea of FQLA. We then describe FQLA with known $\boldsymbol{\gamma}_V^*$, called FQLA-Ideal, and study its performance. After that, we describe the more general FQLA without such knowledge, called FQLA-General. For brevity, we only describe FQLA for the case when $g_0(\boldsymbol{\gamma})$ is locally polyhedral. FQLA for the other case is discussed in [HN11a].

4.3.1 FQLA: a single queue example

To illustrate the idea of FQLA, we first look at an example. Figure 4.1 shows a 10^4-slot sample backlog process under QLA.[8] We see that after roughly 1500 slots, $q(t)$ always stays very close to γ_V^*, which is a $\Theta(V)$ scalar in this case. To reduce delay, we can first find $\mathcal{W} \in (0, \gamma_V^*)$ such that under QLA, there exists a time t_0 so that $q(t_0) \geq \mathcal{W}$ and once

[8]This sample backlog process is one sample backlog process of queue 1 of the system considered in Section 4.3.6, under QLA with $V = 50$.

$q(t) \geq \mathcal{W}$, it remains so for all time (the solid line in Fig. 4.1 shows one for these 10^4 slots). We then place $\mathcal{W}$ fake bits (called *place-holder bits* [NU08]) in the queue at time 0, i.e., initialize $q(0) = \mathcal{W}$, and run QLA. It can be shown, as in [GNT06], that the utility performance of QLA will remain the same with this change, and the average backlog is now reduced by $\mathcal{W}$. However, such a $\mathcal{W}$ may require $\mathcal{W} = \gamma_V^* - \Theta(V)$. Thus, the average backlog may still be $\Theta(V)$.

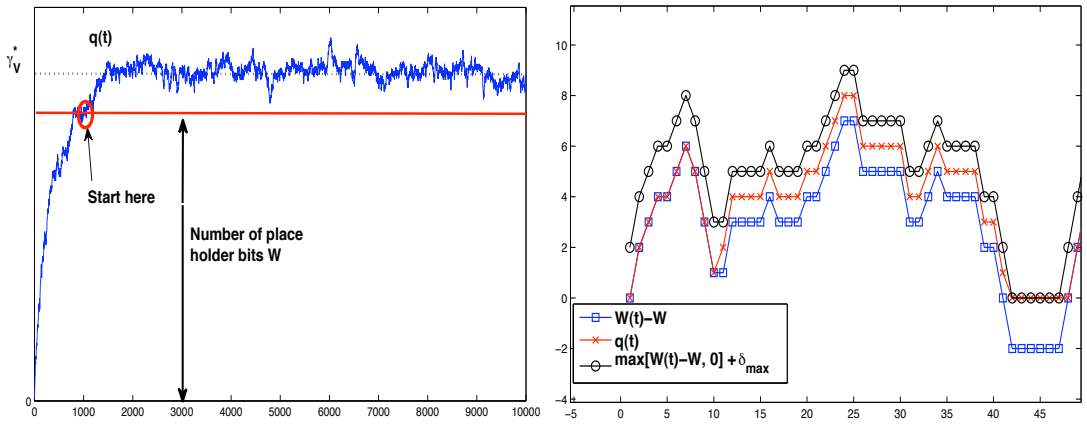

Figure 4.1: Left: A sample backlog process; Right: An example of $W(t)$ and $q(t)$.

FQLA instead finds a $\mathcal{W}$ such that in steady state, the backlog process under QLA *rarely* goes below it, and places $\mathcal{W}$ place-holder bits in the queue at time 0. FQLA then uses an auxiliary process $W(t)$, called the *virtual backlog process*, to keep track of the backlog process that should have been generated if QLA has been used. Specifically, FQLA initializes $W(0) = \mathcal{W}$. Then at every slot, QLA is run using $W(t)$ as the queue size, and $W(t)$ is updated according to QLA. With $W(t)$ and $\mathcal{W}$, FQLA works as follows: At time t, if $W(t) \geq \mathcal{W}$, FQLA performs QLA's action (obtained based on $S(t)$ and $W(t)$); else if $W(t) < \mathcal{W}$, FQLA carefully modifies QLA's action so as to maintain $q(t) \approx \max[W(t) - \mathcal{W}, 0]$ for all t (see Fig. 4.1 for an example). Similar as above, this

roughly reduces the average backlog by $\mathcal{W}$. The difference is that now we can show that $\mathcal{W} = \max[\gamma_V^* - [\log(V)]^2, 0]$ meets the requirement. Thus, it is possible to bring the average backlog down to $O([\log(V)]^2)$. Also, since $W(t)$ can be viewed as a backlog process generated by QLA, it rarely goes below $\mathcal{W}$ in steady state. Hence FQLA is almost always the same as QLA. Thus, is able to achieve an $O(1/V)$ close-to-optimal utility performance.

4.3.2 The FQLA-Ideal algorithm

In this section, we present the FQLA-Ideal algorithm. We assume that the value $\boldsymbol{\gamma}_V^* = (\gamma_{V1}^*, ..., \gamma_{Vr}^*)^T$ is known a-priori.

FQLA-Ideal:

(I) *Determining place-holder bits:* For each j, define:

$$\mathcal{W}_j = \max\left[\gamma_{Vj}^* - [\log(V)]^2, 0\right], \tag{4.52}$$

as the number of *place-holder bits* of queue j.

(II) *Place-holder-bit based action:* Initialize

$$q_j(0) = 0, \quad W_j(0) = \mathcal{W}_j, \quad \forall j.$$

For $t \geq 1$, observe the network state $S(t)$, solve (4.8) with $\boldsymbol{W}(t)$ in place of $\boldsymbol{q}(t)$. Perform the chosen action with the following modification: Let $\boldsymbol{A}(t)$ and $\boldsymbol{\mu}(t)$ be the arrival and service rate vectors generated by the action. For each queue j, do (If queue j does not have enough packets to send, null packets can be transmitted):

(a) If $W_j(t) \geq \mathcal{W}_j$: admit $A_j(t)$ arrivals, serve $\mu_j(t)$ data, i.e., update the backlog by:

$$q_j(t+1) = \max\left[q_j(t) - \mu_j(t), 0\right] + A_j(t).$$

(b) If $W_j(t) < \mathcal{W}_j$: admit $\tilde{A}_j(t) = \max\left[A_j(t) - \mathcal{W}_j + W_j(t), 0\right]$ arrivals, serve $\mu_j(t)$ data, i.e., update the backlog by:

$$q_j(t+1) = \max\left[q_j(t) - \mu_j(t), 0\right] + \tilde{A}_j(t).$$

(c) Update $W_j(t)$ by:

$$W_j(t+1) = \max\left[W_j(t) - \mu_j(t), 0\right] + A_j(t).$$

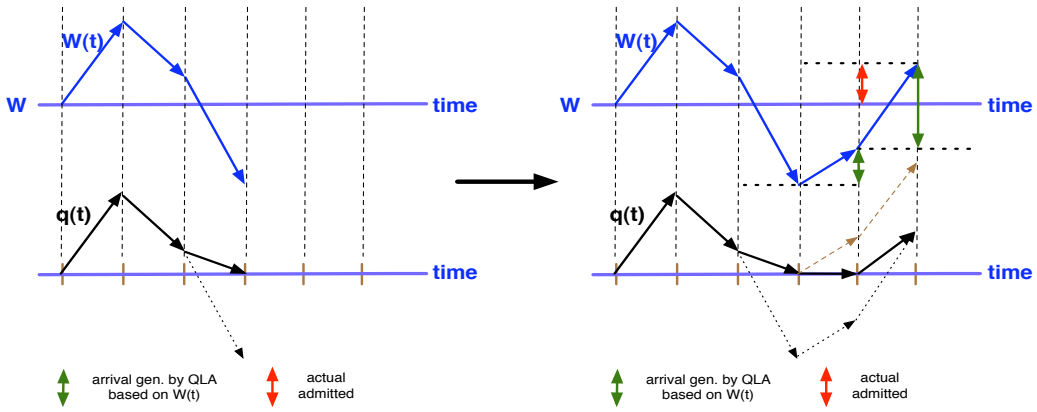

Figure 4.2: Demonstration of the FQLA algorithm for $r = 1$: FQLA is the same as QLA when $W(t) \geq \mathcal{W}$; otherwise it only admits the excessive packets.

Fig. 4.2 shows how FQLA works. From the above, we see that FQLA-Ideal is the same as QLA based on $\boldsymbol{W}(t)$ when $W_j(t) \geq \mathcal{W}_j$ for all j. When $W_j(t) < \mathcal{W}_j$ for some queue j, FQLA-Ideal admits roughly the *excessive* packets after $W_j(t)$ is brought back to be above $\mathcal{W}_j$ for the queue. Thus for problems where QLA admits an easy implementation, e.g., [Nee06c], [HN10c], it is also easy to implement FQLA. However, we also notice two different features of FQLA: (1) By (4.52), $\mathcal{W}_j$ can be 0. However, when V is large, this

happens only when $\gamma_{0j}^* = \gamma_{Vj}^* = 0$ according to Lemma 2. In this case $\mathcal{W}_j = \gamma_{Vj}^* = 0$, and queue j indeed needs zero place-holder bits. (2) Packets may be dropped in Step II-(b) upon their arrival, or after they are admitted into the network in a multihop problem. Such packet dropping is natural in many flow control problems and does not change the nature of these problems. In other problems, where such an option is not available, the packet dropping option is introduced to achieve the desired delay performance, and it can be shown that the fraction of packets dropped can be made arbitrarily small. Note that packet dropping here is to compensate for the deviation from the desired Lagrange multiplier. Thus it is different from that in [Nee06a], where packet dropping is used for drift steering.

4.3.3 Performance of FQLA-Ideal

We look at the performance of FQLA-Ideal in this section. We first have the following lemma showing the relationship between $\boldsymbol{q}(t)$ and $\boldsymbol{W}(t)$ under FQLA-Ideal. We use it later to prove the delay bound of FQLA. Note that the lemma also holds for FQLA-General described later, as FQLA-Ideal/General differ only in the way of determining $\boldsymbol{\mathcal{W}} = (\mathcal{W}_1, ..., \mathcal{W}_r)^T$.

Lemma 4. *Under FQLA-Ideal/General, we have* $\forall\ j, t$*:*

$$\max\left[W_j(t) - \mathcal{W}_j, 0\right] \leq q_j(t) \leq \max\left[W_j(t) - \mathcal{W}_j, 0\right] + \delta_{max} \tag{4.53}$$

where δ_{max} *is defined in Section 3.1.2 to be the upper bound of the number of arriving or departing packets of a queue.*

Proof. See Section 4.10.3. □

The following theorem summarizes the main performance results of FQLA-Ideal. Recall that for a given policy Π, f_{av}^{Π} denotes its average cost defined in (3.7) and $f^{\Pi}(t)$ denotes the cost induced by Π at time t.

Theorem 9. *If the condition in Theorem 5 holds and a steady state distribution exists for the backlog process generated by QLA, then with a sufficiently large V, we have under FQLA-Ideal that,*

$$\overline{q} = O([\log(V)]^2), \tag{4.54}$$

$$f_{av}^{FI} = f_{av}^{*} + O(1/V), \tag{4.55}$$

$$P_{drop} = O(1/V^{c_0 \log(V)}), \tag{4.56}$$

where $c_0 = \Theta(1)$, $\overline{q}$ is the time average network backlog, f_{av}^{FI} is the expected time average cost of FQLA-Ideal, f_{av}^{*} is the optimal time average cost and P_{drop} is the time average fraction of packets that are dropped in Step-II (b).

Proof. Since a steady state distribution exists for the backlog process generated by QLA, we see that $\mathcal{P}(D, m)$ in (4.23) represents the steady state probability of the event that the backlog vector deviates from $\boldsymbol{\gamma}_V^*$ by distance $D + m$. Now since $\boldsymbol{W}(t)$ can be viewed as a backlog process generated by QLA, with $\boldsymbol{W}(0) = \boldsymbol{\mathcal{W}}$ instead of $\mathbf{0}$, we see from the proof of Theorem 5 that Theorem 5 and 6 hold for $\boldsymbol{W}(t)$, and by [GNT06], QLA based on $\boldsymbol{W}(t)$ achieves an average cost of $f_{av}^{*} + O(1/V)$. Hence by Theorem 6, there exist constants $D_1, K_1, c_1^{*} = \Theta(1)$ so that: $\mathcal{P}^{(r)}(D_1, cK_1 \log(V)) \leq \frac{c_1^{*}}{V^c}$. By the definition of $\mathcal{P}^{(r)}(D_1, cK_1 \log(V))$, this implies that in steady state:

$$\Pr\{W_j(t) > \gamma_{Vj}^{*} + D_1 + m\} \leq c_1^{*} e^{-\frac{m}{K_1}}.$$

Now let: $Q_j(t) = \max[W_j(t) - \gamma^*_{Vj} - D_1, 0]$. We see that $\Pr\{Q_j(t) > m\} \leq c_1^* e^{-\frac{m}{K_1}}$, $\forall m \geq 0$. We thus have $\overline{Q_j} = O(1)$, where $\overline{Q_j}$ is the time average value of $Q_j(t)$. Now it can be seen from (4.52) and (4.53) that $q_j(t) \leq Q_j(t) + [\log(V)]^2 + D_1 + \delta_{max}$ for all t. Thus (4.54) follows since for a large V:

$$\overline{q_j} \leq \overline{Q_j} + [\log(V)]^2 + D_1 + \delta_{max} = \Theta([\log(V)]^2), \quad \forall\ j.$$

Now consider the average cost. To save space, we use FI for FQLA-Ideal. From above, we see that QLA based on $\boldsymbol{W}(t)$ achieves an expected average cost of $f^*_{av} + O(1/V)$. Thus it suffices to show that FQLA-Ideal performs almost the same as QLA based on $\boldsymbol{W}(t)$. First we have for all $t \geq 1$ that:

$$\frac{1}{t}\sum_{\tau=0}^{t-1} f^{FI}(\tau) = \frac{1}{t}\sum_{\tau=0}^{t-1} f^{FI}(\tau)1_{E(\tau)} + \frac{1}{t}\sum_{\tau=0}^{t-1} f^{FI}(\tau)1_{E^c(\tau)}.$$

Here $1_{E(\tau)}$ is the indicator function of the event $E(\tau)$, $E(\tau)$ is the event that FQLA-Ideal performs the same action as QLA at time τ, and $1_{E^c(\tau)} = 1 - 1_{E(\tau)}$. Taking expectation on both sides and using the fact that when FQLA-Ideal takes the same action as QLA, $f^{FI}(\tau) = f^{QLA}(\tau)$, we have:

$$\frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\big\{f^{FI}(\tau)\big\} \quad \leq \quad \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\big\{f^{QLA}(\tau)1_{E(\tau)}\big\} + \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\big\{\delta_{max}1_{E^c(\tau)}\big\}.$$

Taking the limit as t goes to infinity on both sides and using $f^{QLA}(\tau)1_{E(\tau)} \leq f^{QLA}(\tau)$, we get:

$$\begin{aligned} f^{FI}_{av} &\leq f^{QLA}_{av} + \delta_{max} \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\big\{1_{E^c(\tau)}\big\} \\ &= f^{QLA}_{av} + \delta_{max} \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \Pr\{E^c(\tau)\}. \end{aligned} \tag{4.57}$$

However, $E^c(\tau)$ is included in the event that there exists a j such that $W_j(\tau) < \mathcal{W}_j$. Therefore by (4.41) in Theorem 6, for a large V such that $\frac{1}{2}[\log(V)]^2 \geq D_1$ and $\log(V) \geq 8K_1$,

$$\begin{aligned}\lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\Pr\{E^c(\tau)\} &\leq \mathcal{P}^{(r)}(D_1, [\log(V)]^2 - D_1) \\ &= O(c_1^*/V^{\frac{1}{2K_1}\log(V)}) \\ &= O(1/V^4). \end{aligned} \tag{4.58}$$

Using this fact in (4.57), we obtain:

$$f_{av}^{FI} = f_{av}^{QLA} + O(\delta_{max}/V^4) = f_{av}^* + O(1/V),$$

where the last equality holds since $f_{av}^{QLA} = f_{av}^* + O(1/V)$. This proves (4.55). (4.56) follows since packets are dropped at time τ only if $E^c(\tau)$ happens, thus by (4.58), the fraction of time when packet dropping happens is $O(1/V^{c_0\log(V)})$ with $c_0 = \frac{1}{2K_1} = \Theta(1)$, and each time no more than $\sqrt{r}B$ packets can be dropped. □

4.3.4 The FQLA-General algorithm

Now we describe the FQLA algorithm without any a-priori knowledge of $\boldsymbol{\gamma}_V^*$, called FQLA-General. FQLA-General first runs the system for a long enough time T, such that the system enters its steady state. Then it chooses a sample of the queue vector value to estimate $\boldsymbol{\gamma}_V^*$ and uses that to decide on $\boldsymbol{\mathcal{W}}$.

FQLA-General:

(I) *Determining place-holder bits:*

(a) Choose a large time T (see Section 4.3.5 for the size of T) and initialize $\boldsymbol{W}(0) = \mathbf{0}$. Run the QLA algorithm with parameter V, at every time slot t, update $\boldsymbol{W}(t)$ according to the QLA algorithm and obtain $\boldsymbol{W}(T)$.

(b) For each queue j, define:

$$\mathcal{W}_j = \max\left[W_j(T) - [\log(V)]^2, 0\right], \tag{4.59}$$

as the number of *place-holder bits*.

(II) *Place-holder-bit based action:* same as FQLA-Ideal.

The performance of FQLA-General is summarized as follows:

Theorem 10. *Assume that the conditions in Theorem 9 hold and the system is in steady state at time T. Then, under FQLA-General with a sufficiently large V, with probability $1 - O(\frac{1}{V^4})$: (a) $\overline{q} = O([\log(V)]^2)$, (b) $f_{av}^{FG} = f_{av}^* + O(1/V)$, and (c) $P_{drop} = O(1/V^{c_0 \log(V)})$, where $c_0 = \Theta(1)$ and f_{av}^{FG} is the expected time average cost of FQLA-General.*

Proof. We will show that with probability $1-O(\frac{1}{V^4})$, $\mathcal{W}_j$ is close to $\max[\gamma_{Vj}^* - [\log(V)]^2, 0]$. The rest can then be proven similarly as in the proof of Theorem 9.

For each queue j, define:

$$v_j^+ = \gamma_{Vj}^* + \frac{1}{2}[\log(V)]^2, \quad v_j^- = \max\left[\gamma_{Vj}^* - \frac{1}{2}[\log(V)]^2, 0\right].$$

Note that v_j^- is defined with a $\max[\cdot,\cdot]$ operator. This is due to the fact that γ_{Vj}^* can be zero. As in (4.58), we see that by Theorem 6, there exists $D_1 = \Theta(1), K_1 = \Theta(1)$ such that if V satisfies $\frac{1}{4}[\log(V)]^2 \geq D_1$ and $\log(V) \geq 16K_1$, then:

$$\Pr\left\{\exists\, j,\, W_j(T) \notin [v_j^-, v_j^+]\right\} \quad \leq \quad \mathcal{P}^{(r)}(D_1, \frac{1}{2}[\log(V)]^2 - D_1)$$

$$= \quad O(1/V^4).$$

Thus $\Pr\{W_j(T) \in [v_j^-, v_j^+] \, \forall j\} = 1 - O(1/V^4)$, implying:

$$\Pr\{\mathcal{W}_j \in [\hat{v}_j^-, \hat{v}_j^+] \quad \forall j\} = 1 - O(1/V^4),$$

where $\hat{v}_j^+ = \max\left[\gamma_{Vj}^* - \frac{1}{2}[\log(V)]^2, 0\right]$ and $\hat{v}_j^- = \max\left[\gamma_{Vj}^* - \frac{3}{2}[\log(V)]^2, 0\right]$. Hence for large V, with probability $1 - O(\frac{1}{V^4})$, if $\gamma_{Vj}^* > 0$, we have $\gamma_{Vj}^* - \frac{3}{2}[\log(V)]^2 \leq \mathcal{W}_j \leq \gamma_{Vj}^* - \frac{1}{2}[\log(V)]^2$; else if $\gamma_{Vj}^* = 0$, we have $\mathcal{W}_j = \gamma_{Vj}^*$. The rest of the proof is similar to the proof of Theorem 9. □

4.3.5 Practical issues

From Lemma 2 we see that the magnitude of $\boldsymbol{\gamma}_V^*$ can be $\Theta(V)$. This means that T in FQLA-General may need to be $\Omega(V)$, which is not very desirable when V is large. We can instead use the following heuristic method to accelerate the process of determining $\boldsymbol{\mathcal{W}}$: For every queue j, guess a very large $\mathcal{W}_j$. Then start with this $\boldsymbol{\mathcal{W}}$ and run the QLA algorithm for some T_1, say $\sqrt{V}$ slots. Observe the resulting backlog process. Modify the guess for each queue j using a bisection algorithm until a proper $\boldsymbol{\mathcal{W}}$ is found, i.e. when running QLA from $\boldsymbol{\mathcal{W}}$, we observe fluctuations of $W_j(t)$ around $\mathcal{W}_j$ instead of a nearly constant increase or decrease for all j. Then let $\mathcal{W}_j = \max[\mathcal{W}_j - [\log(V)]^2, 0]$. To further reduce the error probability, one can repeat Step-I (a) multiple times and use the average value as $\boldsymbol{W}(T)$.

4.3.6 Simulation

In this section we provide simulation results for the FQLA algorithms. For simplicity, we only consider the case where $g_0(\boldsymbol{\gamma})$ is locally polyhedral. We consider a five queue system

similar to the example in Section 3.1.4. In this case $r = 5$. The system is shown in Fig. 4.3. The goal is to perform power allocation at each node so as to support the arrivals with minimum energy expenditure.

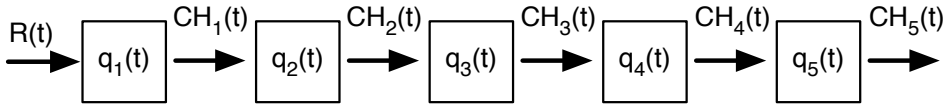

Figure 4.3: A five queue system

In this example, the random network state $S(t)$ is the vector $(R(t), CH_i(t), i = 1, .., 5)$. Similar to Section 3.1.4, we have: $\boldsymbol{A}(t) = (R(t), \mu_1(t), \mu_2(t), \mu_3(t), \mu_4(t))^T$ and $\boldsymbol{\mu}(t) = (\mu_1(t), \mu_2(t), \mu_3(t), \mu_4(t), \mu_5(t))^T$, i.e., $A_1(t) = R(t)$, $A_i(t) = \mu_{i-1}(t)$ for $i \geq 2$, where $\mu_i(t)$ is the service rate obtained by queue i at time t. $R(t)$ is 0 or 2 with probabilities $\frac{3}{8}$ and $\frac{5}{8}$, respectively. $CH_i(t)$ can be "Good" or "Bad" with equal probabilities for $1 \leq i \leq 5$. When the channel is good, one unit of power can serve two packets; otherwise it can serve only one. We assume that the $CH_i(t)$ are all independent, and all channels can be activated at the same time without affecting others. It can be verified that $\boldsymbol{\gamma}_V^* = (5V, 4V, 3V, 2V, V)^T$ is unique. In this example, the backlog vector process evolves as a Markov chain with countably many states. Thus there exists a stationary distribution for the backlog vector under QLA.

We simulate FQLA-Ideal and FQLA-General with $V = 50, 100, 200, 500, 1000$ and 2000. We run each case for $5 \cdot 10^6$ slots. For FQLA-General, we use $T = 50V$ in Step-I and repeat Step-I 100 times and use their average as $\boldsymbol{W}(T)$. The top-left plot in Fig. 4.4 shows that the average queue sizes under both FQLAs are always close to the value $5[\log(V)]^2$ ($r = 5$). The top-right plot shows that the percentage of packets dropped decreases rapidly and gets below 10^{-4} when $V \geq 500$ under both FQLAs. These plots

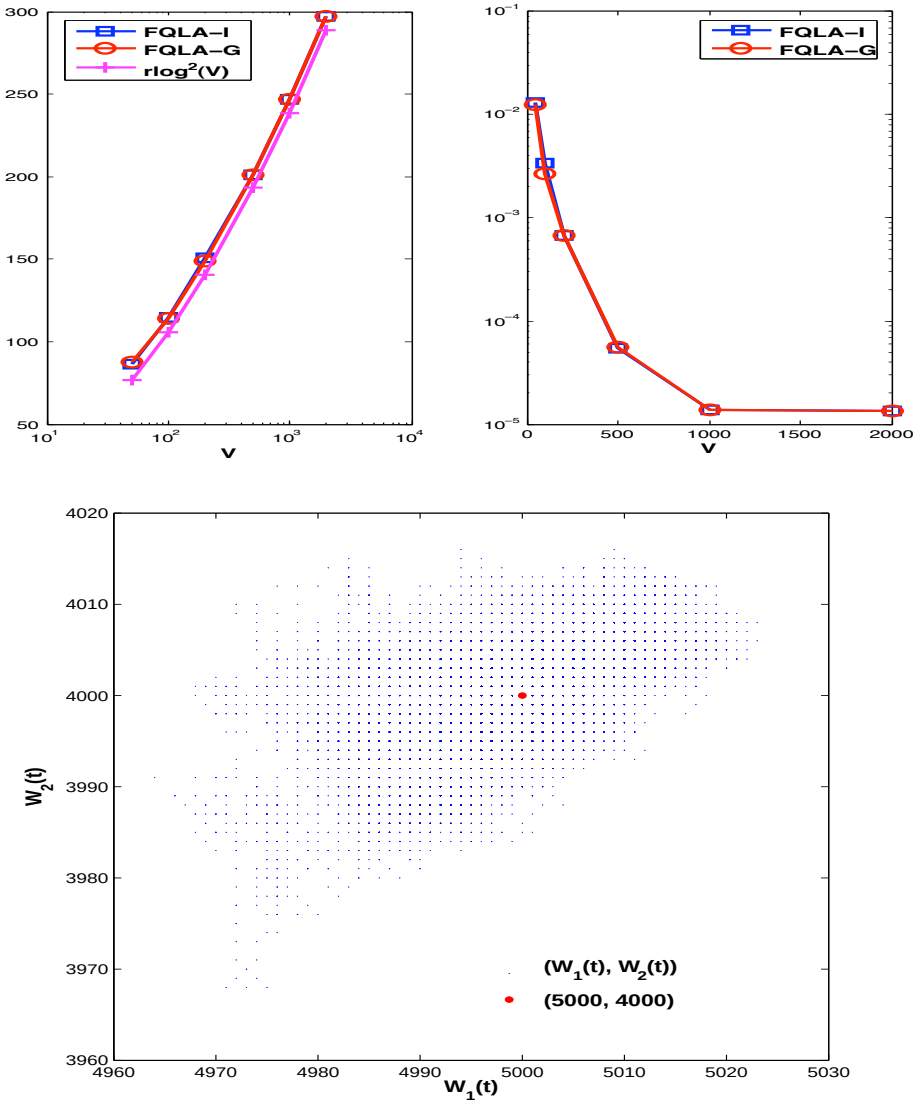

Figure 4.4: FQLA-Ideal performance: Up-Left - Average queue size; Up-Right - Percentage of packets dropped; Bottom - Sample $(W_1(t), W_2(t))$ process for $t \in [10000, 110000]$ and $V = 1000$ under FQLA-Ideal.

show that in practice, V may not have to be very large for Theorem 9 and 10 to hold. The bottom plot shows a sample $(W_1(t), W_2(t))$ process for a 10^5-slot interval under FQLA-Ideal with $V = 1000$, considering only the first two queues of Fig. 4.3. We see that $(W_1(t), W_2(t))$ always remains close to $(\gamma^*_{V1}, \gamma^*_{V2}) = (5V, 4V)$, and $W_1(t) \geq \mathcal{W}_1 = 4952$, $W_2(t) \geq \mathcal{W}_2 = 3952$. For all V values, the average power expenditure is very close to 3.75, which is the optimal energy expenditure, and the average of $\sum_j W_j(t)$ is very close to $15V$.

Interestingly, the "attraction phenomenon" in the bottom plot of Fig. 4.4 was also observed in the system implementation paper [MSKG10], which implemented the QLA algorithm in a 40-node wireless sensor network testbed. It has also been shown in [MSKG10] that by using QLA plus Last-In-First-Out (LIFO), one can reduce the delay experienced by all but a small fraction of the network traffic by more than 90%. While this fact can not be explained by any previous results on QLA, it can easily be explained using Theorems 5 and 8 as follows: Consider a node j. First suppose that First-In-First-Out (FIFO) is used. Then, a packet entering node j is placed at the end of the buffer. By Theorems 5 and 8, the backlog size q_j at node j always stays close to $\gamma^*_{Vj} = \Theta(V)$. Thus, a new packet has to wait for roughly γ^*_{Vj} packets before getting served, resulting in a delay linear in V. Now if LIFO is used, then packets entering node j are placed at the front of the buffer. We know that $q_j \in \mathcal{I} = [\gamma^*_{Vj} - [\log(V)]^2, \gamma^*_{Vj} + [\log(V)]^2]$ for most of the time. Thus most packets enter and leave node j when $q_j \in \mathcal{I}$. Hence for most packets, node j is a queue with on average no more than $2[\log(V)]^2$ packets. Therefore, on average, most packets need to wait for no more than $\Theta([\log(V)]^2)$ packets before getting served. This intuitive argument is made rigorous in the following section.

4.4 The LIFO-Backpressure algorithm

We see from the above discussion that the FQLA algorithms are able to achieve the near optimal utility-delay tradeoff for general network optimization problems. Below, we show that such a near optimal performance can also be achieved by only changing the queueing discipline of the network nodes from First-In-First-Out (FIFO) to Last-In-First-Out (LIFO). This LIFO version of the QLA algorithm was first proposed in a system implementation work [MSKG10], and was demonstrated to yield orders of magnitude delay improvement over the FIFO version, although they did not provide any theoretical performance guarantee. In this section, we provide a rigorous theoretical analysis for this method. The importance of this method, as we will see, is that it does not require any knowledge of $\boldsymbol{\gamma}_V^*$. This greatly simplifies the implementation of the algorithm, and avoids the potential utility lost due to the error that can occur during the leaning phase of FQLA.

We first state this LIFO version of the QLA algorithm below (QLA was stated in Section 4.1.2). We will also follow the convention and call it the LIFO-Backpressure algorithm.

LIFO-Backpressure: At every time slot t, observe the current network state $S(t)$ and the backlog $\boldsymbol{q}(t)$. If $S(t) = s_i$, choose $x^{(s_i)} \in \mathcal{X}^{(s_i)}$ that solves the following:

$$\begin{aligned} \max \quad & -Vf(s_i, x) + \sum_{j=1}^{r} q_j(t)\big[\mu_j(s_i, x) - A_j(s_i, x)\big] \\ s.t. \quad & x \in \mathcal{X}^{(s_i)}. \end{aligned} \tag{4.60}$$

Then serve the packets in each queue using the LIFO discipline.

We show in Theorem 13 that, under the conditions of Theorem 6, under LIFO-Backpressure, the time average delay for almost all packets entering the network is $O([\log(V)]^2)$ when the utility is pushed to within $O(1/V)$ of the optimal value. Note that the implementation complexity of LIFO-Backpressure is the same as the original Backpressure, and LIFO-Backpressure only requires the knowledge of the instantaneous network condition. This is a remarkable feature that distinguishes it from the previous algorithms achieving similar poly-logarithmic tradeoffs in the i.i.d. case, e.g., [Nee06b] [Nee07] [HN11a], which all require knowledge of some implicit network parameters other than the instant network state.

4.4.1 Performance of LIFO-Backpressure

In this section, we analyze the performance LIFO-Backpressure. Below, we first provide a simple example to demonstrate the need for careful treatment of the usage of LIFO in Backpressure algorithms, and then present a modified Little's theorem that will be used for our proof.

4.4.1.1 A simple example on the LIFO delay

Consider a slotted system where two packets arrive at time 0, and one packet periodically arrives every slot thereafter (at times $1, 2, 3, \ldots$). The system is initially empty and can serve exactly one packet per slot. The arrival rate λ is clearly 1 packet/slot (so that $\lambda = 1$). Further, under either FIFO or LIFO service, there are always 2 packets in the system, so $\overline{Q} = 2$.

Under FIFO service, the first packet has a delay of 1 and all packets thereafter have a delay of 2:

$$W_1^{FIFO} = 1\ ,\ W_i^{FIFO} = 2\ \forall i \in \{2, 3, 4, \ldots\},$$

where W_i^{FIFO} is the delay of the i^{th} packet under FIFO (W_i^{LIFO} is similarly defined for LIFO). We thus have:

$$\overline{W}^{FIFO} \triangleq \lim_{K\to\infty} \frac{1}{K} \sum_{i=1}^{K} W_i^{FIFO} = 2.$$

Thus, $\lambda\overline{W}^{FIFO} = 1 \cdot 2 = 2$, $\overline{Q} = 2$, and so $\lambda\overline{W}^{FIFO} = \overline{Q}$ indeed holds.

Now consider the same system under LIFO service. We still have $\lambda = 1$, $\overline{Q} = 2$. However, in this case the first packet never departs, while all other packets have a delay equal to 1 slot:

$$W_1^{LIFO} = \infty\ ,\ W_i^{LIFO} = 1\ \forall i \in \{2, 3, 4, \ldots\}.$$

Thus, for all integers $K > 0$:

$$\frac{1}{K} \sum_{i=1}^{K} W_i^{LIFO} = \infty.$$

and so $\overline{W}^{LIFO} = \infty$. Clearly $\lambda\overline{W}^{LIFO} \neq \overline{Q}$. On the other hand, if we ignore the one packet with infinite delay, we note that all other packets get a delay of 1 (exactly half the delay in the FIFO system). Thus, in this example, LIFO service significantly improves delay for all but the first packet.

For the above LIFO example, it is interesting to note that if we define $\tilde{Q}$ and $\tilde{W}$ as the average backlog and delay *associated only with those packets that eventually depart*, then we have $\tilde{Q} = 1$, $\tilde{W} = 1$, and the equation $\lambda\tilde{W} = \tilde{Q}$ indeed holds. This motivates the

theorem in the next subsection, which considers a time average only over those packets that eventually depart.

4.4.1.2 A Modified Little's Theorem for LIFO systems

We now present the modified Little's theorem. Let $\mathcal{B}$ represent a finite set of buffer locations for a LIFO queueing system. Let $N(t)$ be the number of arrivals that use a buffer location within set $\mathcal{B}$ up to time t. Let $D(t)$ be the number of departures from a buffer location within the set $\mathcal{B}$ up to time t. Let W_i be the delay of the ith job to depart from the set $\mathcal{B}$. [9] Define $\overline{W}$ as the lim sup average delay *considering only those jobs that depart*:

$$\overline{W} \triangleq \limsup_{t\to\infty} \frac{1}{D(t)} \sum_{i=1}^{D(t)} W_i.$$

We then have the following theorem:

Theorem 11. *Suppose that the LIFO queueing discipline is used, that there is a constant* $\lambda_{min} > 0$ *such that with probability 1:*

$$\liminf_{t\to\infty} \frac{N(t)}{t} \geq \lambda_{min},$$

Further suppose that $\lim_{t\to\infty} D(t) = \infty$ *with probability 1 (so the number of departures is infinite). Then the average delay* $\overline{W}$ *satisfies:*

$$\overline{W} \triangleq \limsup_{t\to\infty} \frac{1}{D(t)} \sum_{i=1}^{D(t)} W_i \leq |\mathcal{B}|/\lambda_{min},$$

where $|\mathcal{B}|$ *is the size of the finite set* $\mathcal{B}$.

Proof. See Section 4.10.4. □

[9] Note here that, since we consider a LIFO system, a packet departs from any location in the set $\mathcal{B}$ will leave the queue and not re-enter $\mathcal{B}$.

4.4.1.3 LIFO-Backpressure proof

We now provide the analysis of LIFO-Backpressure. To prove our result, we first have the following theorem, which is the first to show that Backpressure (with either FIFO or LIFO) achieves the exact $[O(1/V), O(V)]$ utility-delay tradeoff under a Markovian network state process. It generalizes the $[O(1/V), O(V)]$ performance result of Backpressure in the i.i.d. case in [GNT06].

Theorem 12. *Suppose that $S(t)$ is a finite state irreducible and aperiodic Markov chain*[10] *and condition (3.1) holds. Then, Backpressure (with either FIFO or LIFO) achieves the following:*

$$f_{av}^{BP} = f_{av}^{*} + O(1/V),\ \overline{q}^{BP} = O(V), \tag{4.61}$$

where f_{av}^{BP} and $\overline{q}^{BP}$ are the expected time average cost and backlog under Backpressure.

Proof. See Section 4.10.5. □

Theorem 12 thus shows that LIFO-Backpressure guarantees an average backlog of $O(V)$ when pushing the utility to within $O(1/V)$ of the optimal value. We now consider the delay performance of LIFO-Backpressure.

Below, the notion "average arrival rate" is defined as follows: Let $A_j(t)$ be the number of packets entering queue j at time t. Then the time average arrival rate of these packets is defined (assuming it exists): $\lambda_j = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} A_j(\tau)$. For the theorem, we assume that time averages under Backpressure exist with probability 1. This is a reasonable assumption, and holds whenever the resulting discrete time Markov chain for the queue

[10] In Section 4.10.5, we prove the theorem under more general Markovian $S(t)$ processes that include the $S(t)$ process assumed here.

vector $\boldsymbol{q}(t)$ under Backpressure is countably infinite and irreducible. Note that the state space is indeed countably infinite if we assume that packets arrive in integer units. If the system is also irreducible then the finite average backlog result of Theorem 12 implies that all states are positive recurrent.

Let D_1, K_1, c_1^* be constants as defined in Theorem 6, and recall that these are all $\Theta(1)$ (independent of V). Assume that $V \geq 1$, and define $Q_{j,\text{High}}$ and $Q_{j,\text{Low}}$ as:

$$Q_{j,\text{High}} \triangleq \gamma_{Vj}^* + D_1 + K_1[\log(V)]^2, \tag{4.62}$$

$$Q_{j,\text{Low}} \triangleq \max[\gamma_{Vj}^* - D_1 - K_1[\log(V)]^2, 0]. \tag{4.63}$$

Define the interval $\mathcal{B}_j \triangleq [Q_{j,\text{Low}}, Q_{j,\text{High}}]$. The following theorem considers the rate and delay of packets that enter when $q_j(t) \in \mathcal{B}_j$ and that eventually depart.

Theorem 13. *Suppose that $V \geq 1$, that $\boldsymbol{\gamma}_V^*$ is unique, that the slackness assumption (3.1) holds, and that the dual function $g(\boldsymbol{\gamma})$ satisfies:*

$$g(\boldsymbol{\gamma}_V^*) \geq g(\boldsymbol{\gamma}) + L||\boldsymbol{\gamma}_V^* - \boldsymbol{\gamma}|| \quad \forall\, \boldsymbol{\gamma} \succeq \mathbf{0}, \tag{4.64}$$

for some constant $L > 0$ independent of V. Define D_1, K_1, c_1^* as in Theorem 6, and define $\mathcal{B}_j$ as above. Then for any queue j with a time average input rate $\lambda_j > 0$, we have under LIFO-Backpressure that:

(a) The rate $\tilde{\lambda}_j$ of packets that both arrive to queue j when $q_j(t) \in \mathcal{B}_j$ and that eventually depart the queue satisfies:

$$\lambda_j \geq \tilde{\lambda}_j \geq \left[\lambda_j - \frac{\delta_{max} c^*}{V^{\log(V)}}\right]^+. \tag{4.65}$$

(b) The average delay of these packets is at most W_{bound}, where:

$$W_{bound} \triangleq [2D_1 + 2K_1[\log(V)]^2 + \delta_{max}]/\tilde{\lambda}_j.$$

This theorem says that the delay of packets that enter when $q_j(t) \in \mathcal{B}_j$ and that eventually depart is at most $O([\log(V)]^2)$. Further, by (4.65), when V is large, these packets represent the overwhelming majority, in that the rate of packets not in this set is at most $O(1/V^{\log(V)})$.

Proof. (Theorem 13) Theorem 12 shows that average queue backlog is finite. Thus, there can be at most a finite number of packets that enter the queue and never depart, so the rate of packets arriving that never depart must be 0. It follows that $\tilde{\lambda}_j$ is equal to the rate at which packets arrive when $q_j(t) \in \mathcal{B}_j$. Define the indicator function $1_j(t)$ to be 1 if $q_j(t) \notin \mathcal{B}_j$, and 0 else. Define $\tilde{\lambda}_j^c \triangleq \lambda_j - \tilde{\lambda}_j$. Then with probability 1 we get: [11]

$$\tilde{\lambda}_j^c = \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} A_j(\tau)1_j(\tau) = \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\big\{A_j(\tau)1_j(\tau)\big\}.$$

Then using the fact that $A_j(t) \leq \delta_{max}$ for all j, t:

$$\begin{aligned}\mathbb{E}\big\{A_j(t)1_j(t)\big\} &= \mathbb{E}\big\{A_j(t)|q_j(t) \notin \mathcal{B}_j\big\}\Pr\{q_j(t) \notin \mathcal{B}_j) \\ &\leq \delta_{max}\Pr(q_j(t) \notin [Q_{j,Low}, Q_{j,High}]).\end{aligned}$$

Therefore:

$$\begin{aligned}\tilde{\lambda}_j^c &\leq \delta_{max} \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \Pr(q_j(\tau) \notin [Q_{j,Low}, Q_{j,High}]) \\ &\leq \delta_{max} \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \Pr(|q_j(\tau) - \gamma_{V,j}^*| > D_1 + K_1 m),\end{aligned}$$

where we define $m \triangleq [\log(V)]^2$, and note that $m \geq 0$ because $V \geq 1$. From Theorem 6 we thus have:

$$0 \leq \tilde{\lambda}_j^c \leq \delta_{max} c_1^* e^{-m} = \frac{\delta_{max} c_1^*}{V^{\log(V)}}. \tag{4.66}$$

[11] The time average expectation is the same as the pure time average by the Lebesgue Dominated Convergence Theorem [Fol99], because we assume that the pure time average exists with probability 1, and that $0 \leq A_j(t) \leq \delta_{max}$ $\forall t$.

This completes the proof of part (a). Now define $\tilde{\mathcal{B}}_j = \triangleq[Q_{j,\text{Low}}, Q_{j,\text{High}} + \delta_{max}]$. Since $\mathcal{B}_j \subset \tilde{\mathcal{B}}_j$, we see that the rate of the packets that enter $\tilde{\mathcal{B}}_j$ is at least $\tilde{\lambda}_j$. Part (b) then follows from Theorem 11 and the facts that queue j is stable and that $|\tilde{\mathcal{B}}_j| \leq 2D + 2K[\log(V)]^2 + \delta_{max}$. □

Note that if $\lambda_j = \Theta(1)$, we see from Theorem 13 that, under LIFO-Backpressure, the time average delay for almost all packets going through queue j is only $O([\log(V)]^2)$. Applying this argument to all network queues with $\Theta(1)$ input rates, we see that all but a tiny fraction of the traffic entering the network only experiences a delay of $O([\log(V)]^2)$. This contrasts with the delay performance result of the usual Backpressure with FIFO, which states that the time average delay will be $\Theta(V)$ for all packets [HN11a]. The downside is that, under LIFO-Backpressure, some packets may stay in the queue for a very long time. This problem can be compensated by introducing certain coding techniques, e.g., fountain codes [Mit04], into the LIFO-Backpressure algorithm.

4.4.2 Simulation

In this section, we provide simulation results of the LIFO-Backpressure algorithm. We consider the network shown in Fig. 4.5, where we try to support a flow sourced by Node 1 destined for Node 7 with minimum energy consumption.

We assume that $A(t)$ evolves according to the 2-state Markov chain in Fig. 4.6. When the state is HIGH, $A(t) = 3$, else $A(t) = 0$. We assume that the condition of each link can either be HIGH or LOW at a time. All the links except link $(2, 4)$ and link $(6, 7)$ are assumed to be i.i.d. every time slot, whereas the conditions of link $(2, 4)$ and link $(6, 7)$ are assumed to be evolving according to independent 2-state Markov chains in Fig. 4.6.

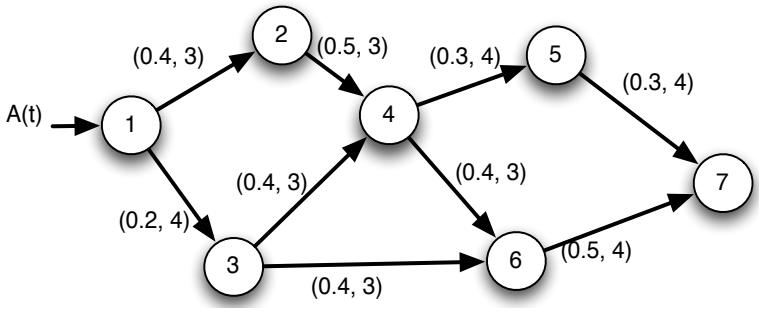

Figure 4.5: A multihop network. (a, b) represents the HIGH probability a and the rate b obtained with one unit of power when HIGH.

Each link's HIGH probability and unit power rate at the HIGH state is shown in Fig. 4.5. The unit power rates of the links at the LOW state are all assumed to be 1. We assume that the link states are all independent and there is no interference. However, each node can only spend one unit of power per slot to transmit over one outgoing link, although it can simultaneously receive from multiple incoming links. The goal is to minimize the time average power while maintaining network stability.

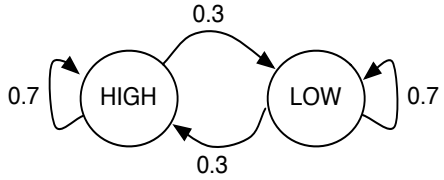

Figure 4.6: The two state Markov chain with the transition probabilities.

We simulate Backpressure with both LIFO and FIFO for 10^6 slots with $V \in \{20, 50, 100, 200, 500\}$. The backlog vector converges to a unique attractor as V increases in this case. The left two plots in Fig. 4.7 show the average power consumption and the average backlog under LIFO-Backpressure. The average power quickly converges to the optimal value and the average backlog grows linearly in V. The right plot of Fig. 4.7 shows the percentage of time when there exists a q_j whose value deviates from γ^*_{Vj} by more than

$2[\log(V)]^2$. As we can see, this percentage is always very small, i.e., between 0.002 and 0.013, showing a good match between the theory and the simulation results.

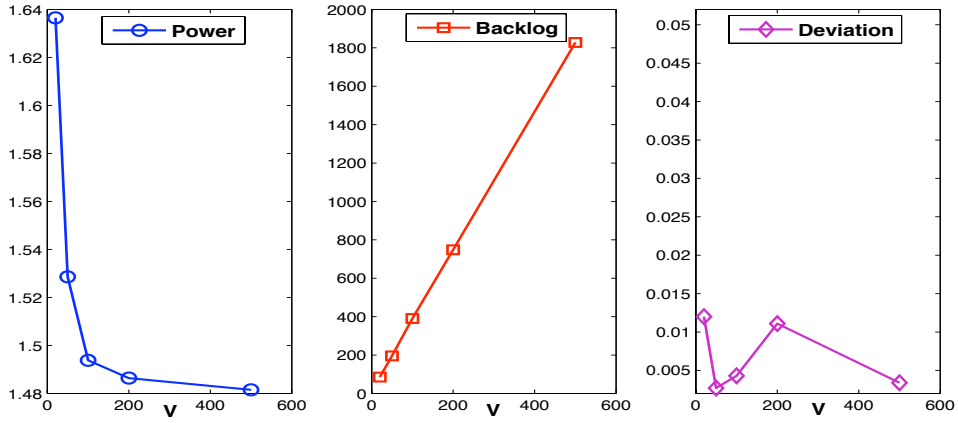

Figure 4.7: LEFT: average network power consumption. MIDDLE: average network backlog size. RIGHT: percentage of time when $\exists\ q_j$ such that $|q_j - \gamma^*_{Vj}| > 2[\log(V)]^2$.

Fig. 4.8 compares the delay statistics of LIFO and FIFO for more than 99.9% of the packets that leave the system before the simulation ends, under the cases $V = 100$ and $V = 500$. We see that LIFO not only dramatically reduces the average packet delay for these packets, but also greatly reduces the delay for most of these packets. For instance, when $V = 500$, under FIFO, almost all packets experience a delay that is equal to the average delay, which is around 1220 slots. Under LIFO, the average packet delay is brought down to 78. Moreover, 52.9% of the packets only experience delay less than 20 slots, and 90.4% of the packets experience delay less than 100 slots. *Hence most packets' delay are reduced by a factor of* 12 *under LIFO as compared to that under FIFO!*

Fig. 4.9 also shows the delay for the first 20000 packets that enter the network in the case when $V = 500$. We see that under Backpressure plus LIFO, most of the packets experience very small delay; under Backpressure with FIFO, each packet experiences roughly the average delay.

V=100				
Case	Avg. DL	% $DL < 20$	% $DL < 50$	% $DL < 100$
LIFO	55.4	55.0	82.1	91.8
FIFO	260.6	0	0	0

V=500				
Case	Avg. DL	% $DL < 20$	% $DL < 50$	% $DL < 100$
LIFO	78.3	52.9	80.4	90.4
FIFO	1219.8	0	0	0

Figure 4.8: Delay statistics under Backpressure with LIFO and FIFO for packets that leave the system before simulation ends (more than 99.9%). $\%DL < a$ is the percentage of packets that enter the network and have delay less than a.

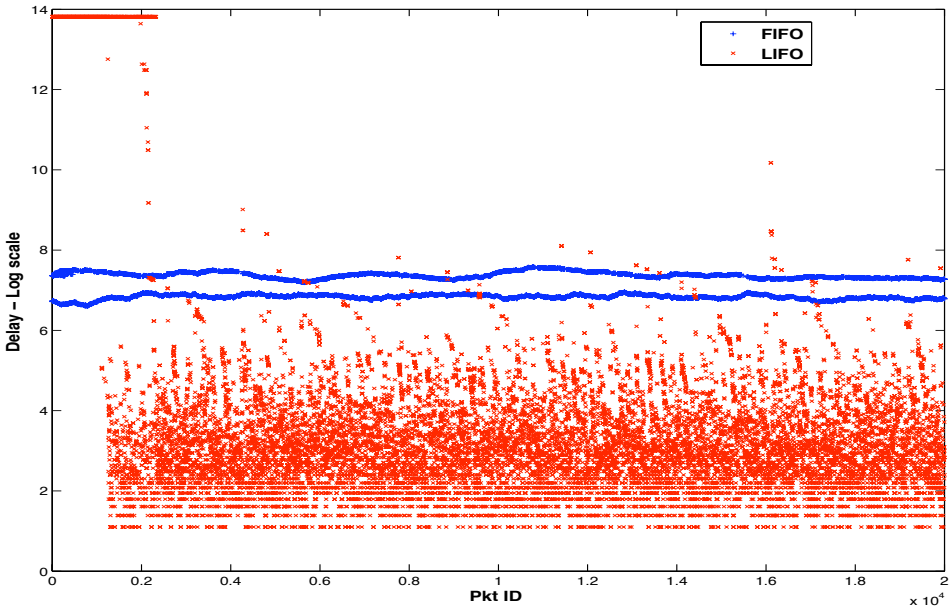

Figure 4.9: Packet delay under Backpressure with LIFO and FIFO

4.5 The LIFO^p-Backpressure algorithm

In this section, we generalized the LIFO-Backpressure technique to allow interleaving between both LIFO and FIFO. Specifically, at every time slot, each queue will randomly decide to serve packets from either the back of the queue or the front of the queue. The motivation of this interleaving approach is that a few packets may get stuck in the queues for a very long time under LIFO-Backpressure. Thus we want to resolve this problem by

also allowing the FIFO discipline. We parameterize the algorithm by a single parameter $p \in [0, 1]$, [12] which represents the probability that a queue serves the packets from the back of the queue at a given time. We call this approach the LIFO^p-Backpressure algorithm.

4.5.1 The algorithm

The idea of LIFO^p-Backpressure is shown in Fig. 4.10: We first pre-specify a probability p. [13] Then at every time slot, after choosing all the network action decisions according to Backpressure, the queue serves packets from its end with probability p; and from the front otherwise. Note that this back/front decision is independent of the action taken by Backpressure, and is independent for each queue. As we vary p from 0 to 1, the algorithm basically goes from the usual (FIFO) Backpressure algorithm to the LIFO-Backpressure algorithm.

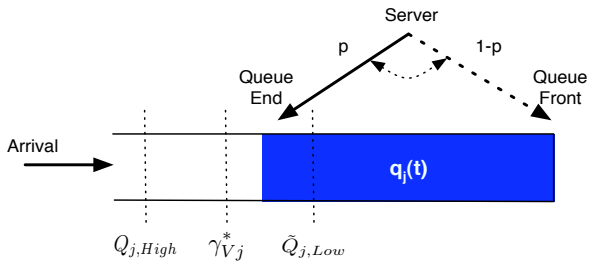

Figure 4.10: The LIFO-FIFO interleaving technique. A packet is either served when the queue is serving the end of the queue, or it gradually moves to the right towards the front of the queue and is served when the queue is serving the packets from the front of the queue.

[12] All the results in this section hold even if we choose different probabilities for different queues.

[13] This p value is pre-determined and not be changed. It is possible to use a p value that is a function of time in implementation. However, the analysis will be very challenging.

4.5.2 Performance analysis

We analyze the performance of the LIFO^p-Backpressure algorithm. First note that since all the sample path queue processes here are the same as those under the regular Backpressure, the $O(1/V)$ close-to-optimal utility performance bounds still apply. Our following theorem shows that LIFO^p-Backpressure ensures that roughly a fraction p of the packets experience only $O([\log(V)]^2)$ queueing delay. This result generalizes Theorem 13. In the theorem, we use $Q_{j,\text{High}}$ and $Q_{j,\text{Low}}$ defined in (4.62) and (4.63), i.e.,

$$\begin{aligned} Q_{j,\text{High}} &\triangleq \gamma^*_{Vj} + D_1 + K_1[\log(V)]^2, \\ Q_{j,\text{Low}} &\triangleq \max[\gamma^*_{Vj} - D_1 - K_1[\log(V)]^2, 0], \end{aligned}$$

as well as:

$$\tilde{Q}_{j,\text{Low}} \triangleq \max[\gamma^*_{Vj} - D_1 - K_1[\log(V)]^2 - \delta_{max}, 0].$$

Note that $\tilde{Q}_{j,\text{Low}}$ is a bit different from the $Q_{j,\text{Low}}$ and is chosen to make the analysis easier. Then we similarly define

$$\tilde{\mathcal{B}}_j \triangleq [\tilde{Q}_{j,\text{Low}}, Q_{j,\text{High}}], \tag{4.67}$$

and assume throughout the theorem that all the corresponding limits exist.

Theorem 14. *Suppose that $V \geq 1$, that $\boldsymbol{\gamma}^*_V$ is unique, that the slackness assumption (3.1) holds, and that the dual function $g(\boldsymbol{\gamma})$ satisfies:*

$$g(\boldsymbol{\gamma}^*_V) \geq g(\boldsymbol{\gamma}) + L||\boldsymbol{\gamma}^*_V - \boldsymbol{\gamma}|| \quad \forall\, \boldsymbol{\gamma} \succeq \mathbf{0}, \tag{4.68}$$

for some constant $L > 0$ independent of V. Define D_1, K_1, c_1^* as in Theorem 6, and define $\tilde{\mathcal{B}}_j$ as in (4.67) above. Then for any queue j with a time average input rate $\lambda_j > 0$, we have under LIFO^p-Backpressure that:

(a) All the packets eventually leave the queue if $0 \leq p < 1$.

(b) There exists a set of packets $\mathbb{P}_{j0}$ that arrive to queue j when $q_j(t) \in \tilde{\mathcal{B}}_j$, that depart before they move to the right of $\tilde{Q}_{j,Low}$, and that are served when the queue is serving the back of the queue. And $\mathbb{P}_{j0}$ has an average rate $\lambda_{\mathbb{P}_{j0}}$ that satisfies:

$$p\lambda_j \geq \lambda_{\mathbb{P}_{j0}} \geq \left[p\lambda_j - O(\frac{\delta_{max}c_1^*}{V^{\log(V)}})\right]^+. \tag{4.69}$$

(c) If $\lambda_{\mathbb{P}_{j0}} > 0$, the average delay of these packets is at most $\hat{W}_{bound}$, where:

$$\hat{W}_{bound} \triangleq 2(D_1 + K_1[\log(V)]^2 + \delta_{max})/\lambda_{\mathbb{P}_{j0}}.$$

Theorem 14 says that by allocating a portion of the time serving the packets from the front of the queue, the problem of packets being stuck in the queue can be resolved. If the p parameter is chosen to be very close to one, then LIFO^p-Backpressure achieves almost the same performance guarantee as LIFO-Backpressure, while ensuring that all the packets are delivered.

The formal proof of Theorem 14 is given in Section 4.10.6. Here we sketch the proof idea: under the LIFO^p-Backpressure policy, for any queue j with an input rate $\lambda_j > 0$, the fraction of the packets that are served when the queue is serving the back of the queue is $p\lambda_j$. Now we look at these packets that are served from the back. First, we see that they will almost all arrive to the queue when $q_j \in \tilde{\mathcal{B}}_j$ by Theorem 6. Second, they will also almost all leave the queue before they move to the right of $\tilde{Q}_{j,\text{Low}}$. The reason for this is that if a packet moves to the right of $\tilde{Q}_{j,\text{Low}}$, i.e., it moves into a buffer spot in $[0, \tilde{Q}_{j,\text{Low}})$, then since $q_j(t)$ rarely gets below $Q_{j,\text{Low}} \approx \tilde{Q}_{j,\text{Low}} + \mu_{max}$, it is very unlikely that this packet will be served from the back of the queue. And it can only gradually move to the front of the queue and be served there. Therefore, almost all the packets that are served from the back will enter and leave the queue when they are in

the interval $\tilde{\mathcal{B}}_j$, which is of size $O([\log(V)]^2)$, and they have an average rate of roughly $p\lambda_j$. Using Theorem 11, we see that they experience an average delay of no more than $O([\log(V)]^2)/p\lambda_j$.

4.6 Discussion of assumption 1: the uniqueness of the optimal Lagrange multiplier

Without Assumption 1, all the exponential attraction results developed in Section 4.2 will still hold, provided that we replace $\|\boldsymbol{\gamma}-\boldsymbol{\gamma}_V^*\|$ with $\text{dist}(\boldsymbol{\gamma}, \boldsymbol{\Gamma}_V^*)$, where $\boldsymbol{\Gamma}_V^*$ is the set of optimal Lagrange multipliers to the dual problem under a given V value, and $\text{dist}(\boldsymbol{a}, \boldsymbol{X})$ is the Euclidean distance between the set $\boldsymbol{X}$ and vector $\boldsymbol{a}$. In other words, the backlog vector is exponentially attracted to a set.

However, the assumption is needed for the development of all the delay-efficient algorithms, i.e., FQLA, LIFO-Backpressure and LIFO^p-Backpressure. This is not because the algorithms are not "clever" enough. Indeed, if the assumption is violated, $\boldsymbol{\Gamma}_V^*$ will be a set of size $\Theta(V)$ by Lemma 2. In this case, the backlog vector can take any value within this set, and still make optimal control actions. This situation makes it impossible to design delay-efficient algorithms using only quadratic Lyapunov functions. It also raises the interesting open problem on how to modify the deterministic problem so as to make $\boldsymbol{\Gamma}_V^*$ a singleton in a stochastic setting to enable the development of delay-efficient algorithms, for networks where Assumption 1 does not hold.

4.7 Comparing QLA and the greedy primal-dual approach

Another general online algorithm design technique is the greedy primal-dual approach (GPD) developed in [BN07]. Although GPD looks similar to QLA considered in this chapter, they differ from each other in the following:

- QLA applies to problems with stochastic inputs and concerns about asymptotic system performance. The performance guarantee is usually given in terms of the utility loss. GPD is able to handle general sample path inputs, and focuses more on finite horizon system utility. The performance measure of GPD algorithms is the ratio between the achieved performance and the optimal value.

- QLA applies to problems with general constraints and objectives; GPD applies to problems that can be formulated as a packing-covering program.

- QLA is closely related to the dual subgradient method, and relies on using time averages to ensure that the constraints are met. The queue vector under QLA naturally represents the Lagrange multiplier, which makes QLA an ideal tool for controlled queueing network problems. GPD instead relies on a general method for updating the primal and dual variables, and LP rounding techniques [Vaz03]. Under GPD, it is possible that the constraints are violated. The technique is very useful for general combinatorial optimization problems.

4.8 Lagrange Multiplier: "shadow price" and "network gravity"

It is well known that Lagrange Multipliers can play the role of "shadow prices" to regulate flows in many flow-based problems with different objectives, e.g., [Kel97b]. This important feature has enabled the development of many distributed algorithms in resource allocation problems, e.g., [CNT05]. However, a problem of this type typically requires data transmissions to be represented as flows. Thus in a network that is discrete in nature, e.g., time slotted or packetized transmission, a rate allocation solution obtained by solving such a flow-based problem does not immediately specify a scheduling policy.

Recently, several Lyapunov algorithms have been proposed to solve utility optimization problems under discrete network settings. In these algorithms, backlog vectors act as the "gravity" of the network and allow optimal scheduling to be built upon them. It is also revealed in [NMR05] that QLA is closely related to the dual subgradient method and backlogs play the same role as Lagrange multipliers in a time invariant network. Now we see by Theorem 5 and 8 that the backlogs indeed play the same role as Lagrange multipliers even under a more general stochastic network.

In fact, the backlog process under QLA can be closely related to a sequence of updated Lagrange multipliers under a subgradient method. Consider the following important variant of OSM, called the randomized incremental subgradient method (RISM) [BNO03], which makes use of the separable nature of (4.13) and solves the dual problem (4.11) as follows:

RISM: Initialize $\boldsymbol{\gamma}(0)$; at iteration t, observe $\boldsymbol{\gamma}(t)$, choose a random state $S(t) \in \mathcal{S}$ according to some probability law. (1) If $S(t) = s_i$, find $x^{(s_i)}_{\boldsymbol{\gamma}(t)} \in \mathcal{X}^{(s_i)}$ that solves the following:

$$\begin{aligned} \min: \quad & Vf(s_i, x) + \sum_j \gamma_j(t)\big[A_j(s_i, x) - \mu_j(s_i, x)\big] \\ \text{s.t.} \quad & x \in \mathcal{X}^{(s_i)}. \end{aligned} \tag{4.70}$$

(2) Choose a step size α^t. Using the $x^{(s_i)}_{\boldsymbol{\gamma}(t)}$ found, update $\boldsymbol{\gamma}(t)$ according to: [14]

$$\gamma_j(t+1) = \max\left[\gamma_j(t) - \alpha^t \mu_j(s_i, x^{(s_i)}_{\boldsymbol{\gamma}(t)}), 0\right] + \alpha^t A_j(s_i, x^{(s_i)}_{\boldsymbol{\gamma}(t)}).$$

As an example, $S(t)$ can be chosen by independently choosing $S(t) = s_i$ with probability π_{s_i} every time slot. In this case $S(t)$ is i.i.d. Note that in the stochastic problem, a network state s_i is chosen randomly by nature as the physical system state at time t, while here a state is chosen artificially by RISM according to some probability law. Now we see from (4.8) and (4.70) that given $\boldsymbol{q}(t) = \boldsymbol{\gamma}(t)$ and s_i, *QLA and RISM choose an action in the same way.* If also $\alpha^t = 1$ for all t, and $S(t)$ under RISM evolves according to the same probability law as $S(t)$ of the physical system, we see that *applying QLA to the network is indeed equivalent to applying RISM to the dual problem of (4.10), with the network state being chosen by nature, and the network backlog being the Lagrange multiplier.* Therefore, Lagrange Multipliers under such stochastic discrete network settings act as the "network gravity," thus allowing scheduling to be done optimally and adaptively based on them. This "network gravity" functionality of Lagrange Multipliers in discrete network problems can thus be viewed as the counterpart of their "shadow price" functionality in the flow-based problems. Further more, the "network gravity"

[14] Note that this update rule is different from RISM's usual rule, i.e., $\gamma_j(t+1) = \max\big[\gamma_j(t) - \alpha^t \mu_j(s_i, x) + \alpha^t A_j(s_i, x), 0\big]$, but it almost does not affect the performance of RISM.

property of Lagrange Multipliers enables the use of place holder bits to reduce network delay in network utility optimization problems. This is a unique feature not possessed by its "price" counterpart. We can also see from this connection that, the convergence time of QLA can be characterized by studying the convergence time of the equivalent RISM algorithm, which has been previously studied in [BNO03].

4.9 Chapter summary

In this chapter, we study the backlog behavior under the QLA algorithm for a class of stochastic network optimization problems. We show that for every such problem, under some mild conditions, the network backlog is "exponentially attracted" to an attractor, which is the dual optimal solution of a corresponding *deterministic* optimization problem. Based on this finding, we develop three algorithms, i.e., FQLA, LIFO-Backpressure, LIFO^p-Backpressure, to achieve an $[O(1/V), O([\log(V)]^2)]$ utility-delay tradeoff for problems with a discrete set of action options, and a square-root tradeoff for continuous problems. The results in this chapter demonstrate how the Lyapunov networking technique is connected to the classic subgradient methods, and provide new insights into the Lyapunov technique.

4.10 Proofs of the chapter

4.10.1 Proof of Lemma 3

Here we prove Lemma 3. First we prove the following:

Lemma 5. *Under queueing dynamic (3.5), we have:*

$$\|\boldsymbol{q}(t+1)-\boldsymbol{\gamma}_V^*\|^2 \;\leq\; \|\boldsymbol{q}(t)-\boldsymbol{\gamma}_V^*\|^2+2B^2-2\big(\boldsymbol{\gamma}_V^*-\boldsymbol{q}(t)\big)^T(\boldsymbol{A}(t)-\boldsymbol{\mu}(t)).$$

Proof. (Lemma 5) From (3.5), we see that $\boldsymbol{q}(t+1)$ is obtained by first projecting $\boldsymbol{q}(t)-\boldsymbol{\mu}(t)$ onto $\mathbb{R}_+^r$ and then adding $\boldsymbol{A}(t)$. Thus we have (we use $[\boldsymbol{a}]^+$ to denote the projection of $\boldsymbol{a}$ onto $\mathbb{R}_+^r$):

$$\begin{aligned}
\|\boldsymbol{q}(t+1)-\boldsymbol{\gamma}_V^*\|^2 &= \|[\boldsymbol{q}(t)-\boldsymbol{\mu}(t)]^+ + \boldsymbol{A}(t)-\boldsymbol{\gamma}_V^*\|^2\\
&= \big([\boldsymbol{q}(t)-\boldsymbol{\mu}(t)]^+ + \boldsymbol{A}(t)-\boldsymbol{\gamma}_V^*\big)^T\big([\boldsymbol{q}(t)-\boldsymbol{\mu}(t)]^+ + \boldsymbol{A}(t)-\boldsymbol{\gamma}_V^*\big)\\
&= \big([\boldsymbol{q}(t)-\boldsymbol{\mu}(t)]^+ - \boldsymbol{\gamma}_V^*\big)^T\big([\boldsymbol{q}(t)-\boldsymbol{\mu}(t)]^+ - \boldsymbol{\gamma}_V^*\big)\\
&\qquad +2\big([\boldsymbol{q}(t)-\boldsymbol{\mu}(t)]^+ - \boldsymbol{\gamma}_V^*\big)^T\boldsymbol{A}(t)+\|\boldsymbol{A}(t)\|^2. \qquad (4.71)
\end{aligned}$$

By the non expansive property of projection [BNO03], we have:

$$\begin{aligned}
&\big([\boldsymbol{q}(t)-\boldsymbol{\mu}(t)]^+ - \boldsymbol{\gamma}_V^*\big)^T\big([\boldsymbol{q}(t)-\boldsymbol{\mu}(t)]^+ - \boldsymbol{\gamma}_V^*\big)\\
&\qquad\leq \big(\boldsymbol{q}(t)-\boldsymbol{\mu}(t)-\boldsymbol{\gamma}_V^*\big)^T\big(\boldsymbol{q}(t)-\boldsymbol{\mu}(t)-\boldsymbol{\gamma}_V^*\big)\\
&\qquad= \|\boldsymbol{q}(t)-\boldsymbol{\gamma}_V^*\|^2+\|\boldsymbol{\mu}(t)\|^2-2(\boldsymbol{q}(t)-\boldsymbol{\gamma}_V^*)^T\boldsymbol{\mu}(t).
\end{aligned}$$

Plug this into (4.71), we have:

$$\begin{aligned}
\|\boldsymbol{q}(t+1)-\boldsymbol{\gamma}_V^*\|^2 \;\leq\; &\|\boldsymbol{q}(t)-\boldsymbol{\gamma}_V^*\|^2+\|\boldsymbol{\mu}(t)\|^2-2(\boldsymbol{q}(t)-\boldsymbol{\gamma}_V^*)^T\boldsymbol{\mu}(t) \qquad (4.72)\\
&+\|\boldsymbol{A}(t)\|^2+2\big([\boldsymbol{q}(t)-\boldsymbol{\mu}(t)]^+ - \boldsymbol{\gamma}_V^*\big)^T\boldsymbol{A}(t).
\end{aligned}$$

Now since $\boldsymbol{q}(t),\boldsymbol{\mu}(t),\boldsymbol{A}(t)\succeq\boldsymbol{0}$, it is easy to see that:

$$\big([\boldsymbol{q}(t)-\boldsymbol{\mu}(t)]^+\big)^T\boldsymbol{A}(t)\leq\boldsymbol{q}(t)^T\boldsymbol{A}(t). \qquad (4.73)$$

By (4.72) and (4.73) we have:

$$\begin{aligned}
\|\boldsymbol{q}(t+1)-\boldsymbol{\gamma}_V^*\|^2 \;\leq\; &\|\boldsymbol{q}(t)-\boldsymbol{\gamma}_V^*\|^2+\|\boldsymbol{\mu}(t)\|^2-2(\boldsymbol{q}(t)-\boldsymbol{\gamma}_V^*)^T\boldsymbol{\mu}(t)\\
&+\|\boldsymbol{A}(t)\|^2+2\big(\boldsymbol{q}(t)-\boldsymbol{\gamma}_V^*\big)^T\boldsymbol{A}(t)
\end{aligned}$$

$$\leq \quad \|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\|^2 + 2B^2 - 2\big(\boldsymbol{\gamma}_V^* - \boldsymbol{q}(t)\big)^T(\boldsymbol{A}(t) - \boldsymbol{\mu}(t)),$$

where the last inequality follows since $\|\boldsymbol{A}(t)\|^2 \leq B^2$ and $\|\boldsymbol{\mu}(t)\|^2 \leq B^2$. □

We now prove Lemma 3.

Proof. (Lemma 3) By Lemma 5 we see that when $S(t) = s_i$, we have the following for any network state s_i with a given $\boldsymbol{q}(t)$ (here we add superscripts to $\boldsymbol{q}(t+1)$, $\boldsymbol{A}(t)$ and $\boldsymbol{\mu}(t)$ to indicate their dependence on s_i):

$$\|\boldsymbol{q}^{(s_i)}(t+1) - \boldsymbol{\gamma}_V^*\|^2 \leq \|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\|^2 + 2B^2 \\ -2(\boldsymbol{\gamma}_V^* - \boldsymbol{q}(t))^T(\boldsymbol{A}^{(s_i)}(t) - \boldsymbol{\mu}^{(s_i)}(t)). \tag{4.74}$$

By definition, $A_j^{(s_i)}(t) = A_j(s_i, x_{\boldsymbol{q}(t)}^{(s_i)})$, and $\mu_j^{(s_i)}(t) = \mu_j(s_i, x_{\boldsymbol{q}(t)}^{(s_i)})$, with $x_{\boldsymbol{q}(t)}^{(s_i)}$ being the solution of (4.8) for the given $\boldsymbol{q}(t)$. Now consider the deterministic problem (4.10) with only a single network state s_i, then the corresponding dual function (4.12) becomes:

$$g_{s_i}(\boldsymbol{\gamma}) = \inf_{x^{(s_i)} \in \mathcal{X}^{(s_i)}} \left\{ V f(s_i, x^{(s_i)}) + \sum_j \gamma_j \big[A_j(s_i, x^{(s_i)}) - \mu_j(s_i, x^{(s_i)})\big] \right\}. \tag{4.75}$$

Therefore, by (4.15) we see that $(\boldsymbol{A}^{(s_i)}(t) - \boldsymbol{\mu}^{(s_i)}(t))$ is a subgradient of $g_{s_i}(\boldsymbol{\gamma})$ at $\boldsymbol{q}(t)$. Thus, by (4.16) we have:

$$(\boldsymbol{\gamma}_V^* - \boldsymbol{q}(t))^T(\boldsymbol{A}^{(s_i)}(t) - \boldsymbol{\mu}^{(s_i)}(t)) \geq g_{s_i}(\boldsymbol{\gamma}_V^*) - g_{s_i}(\boldsymbol{q}(t)). \tag{4.76}$$

Plugging (4.76) into (4.74), we get:

$$\|\boldsymbol{q}^{(s_i)}(t+1) - \boldsymbol{\gamma}_V^*\|^2 \quad \leq \quad \|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\|^2 + 2B^2 - 2\big(g_{s_i}(\boldsymbol{\gamma}_V^*) - g_{s_i}(\boldsymbol{q}(t))\big). \tag{4.77}$$

More generally, we have:

$$\|\boldsymbol{q}(t+1) - \boldsymbol{\gamma}_V^*\|^2 \leq \|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\|^2 + 2B^2 - 2\big(g_{S(t)}(\boldsymbol{\gamma}_V^*) - g_{S(t)}(\boldsymbol{q}(t))\big). \tag{4.78}$$

Now fixing $\nu > 0$ and summing up (4.78) from time t to $t + T_\nu - 1$, we obtain:

$$\|\boldsymbol{q}(t+T_\nu) - \boldsymbol{\gamma}_V^*\|^2 \leq \|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\|^2 + 2T_\nu B^2 \tag{4.79}$$

$$-2\sum_{\tau=0}^{T_\nu-1}\big[g_{S(t+\tau)}(\boldsymbol{\gamma}_V^*)-g_{S(t+\tau)}(\boldsymbol{q}(t+\tau))\big].$$

Adding and subtracting the term $2\sum_{\tau=0}^{T_\nu-1} g_{S(t+\tau)}(\boldsymbol{q}(t))$ from the RHS, we obtain:

$$\|\boldsymbol{q}(t+T_\nu)-\boldsymbol{\gamma}_V^*\|^2 \le \|\boldsymbol{q}(t)-\boldsymbol{\gamma}_V^*\|^2 + 2T_\nu B^2 \tag{4.80}$$
$$-2\sum_{\tau=0}^{T_\nu-1}\big[g_{S(t+\tau)}(\boldsymbol{\gamma}_V^*)-g_{S(t+\tau)}(\boldsymbol{q}(t))\big]+2\sum_{\tau=0}^{T_\nu-1}\big[g_{S(t+\tau)}(\boldsymbol{q}(t+\tau))-g_{S(t+\tau)}(\boldsymbol{q}(t))\big].$$

Since $\|\boldsymbol{q}(t)-\boldsymbol{q}(t+\tau)\|\le\tau B$ and $\|\boldsymbol{A}^{(s_i)}(t)-\boldsymbol{\mu}^{(s_i)}(t)\|\le B$, using (4.76) and the fact that for any two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, $\boldsymbol{a}^T\boldsymbol{b}\le\|\boldsymbol{a}\|\|\boldsymbol{b}\|$, we have:

$$g_{S(t+\tau)}(\boldsymbol{q}(t+\tau))-g_{S(t+\tau)}(\boldsymbol{q}(t))\le\tau B^2. \tag{4.81}$$

Hence:

$$\sum_{\tau=0}^{T_\nu-1}\big[g_{S(t+\tau)}(\boldsymbol{q}(t+\tau))-g_{S(t+\tau)}(\boldsymbol{q}(t))\big]\le\sum_{\tau=0}^{T_\nu-1}\big(\tau B^2\big)=\frac{1}{2}(T_\nu^2B^2-T_\nu B^2).$$

Plugging this into (4.80), we have:

$$\|\boldsymbol{q}(t+T_\nu)-\boldsymbol{\gamma}_V^*\|^2\le\|\boldsymbol{q}(t)-\boldsymbol{\gamma}_V^*\|^2+(T_\nu^2+T_\nu)B^2 \tag{4.82}$$
$$-2\sum_{\tau=0}^{T_\nu-1}\big[g_{S(t+\tau)}(\boldsymbol{\gamma}_V^*)-g_{S(t+\tau)}(\boldsymbol{q}(t))\big].$$

Now denote $\mathcal{Z}(t)=(\mathcal{H}(t),\boldsymbol{q}(t))$, i.e., the pair of the history up to time t, $\mathcal{H}(t)=\{S(\tau)\}_{\tau=0}^{t-1}$ and the current backlog. Taking expectations on both sides of (4.82), conditioning on $\mathcal{Z}(t)$, we have:

$$\begin{aligned}\mathbb{E}\big\{\|\boldsymbol{q}(t+T_\nu)-\boldsymbol{\gamma}_V^*\|^2\mid\mathcal{Z}(t)\big\} &\le \mathbb{E}\big\{\|\boldsymbol{q}(t)-\boldsymbol{\gamma}_V^*\|^2\mid\mathcal{Z}(t)\big\}+(T_\nu^2+T_\nu)B^2\\ &\quad -2\mathbb{E}\big\{\sum_{\tau=0}^{T_\nu-1}\big[g_{S(t+\tau)}(\boldsymbol{\gamma}_V^*)-g_{S(t+\tau)}(\boldsymbol{q}(t))\big]\mid\mathcal{Z}(t)\big\}.\end{aligned}$$

Since the number of times $g_{s_i}(\boldsymbol{\gamma})$ appears in the interval $[t,t+T_\nu-1]$ is $\|\mathcal{T}_{s_i}(t,T_\nu)\|$, we can rewrite the above as:

$$\begin{aligned}\mathbb{E}\big\{\|\boldsymbol{q}(t+T_\nu)-\boldsymbol{\gamma}_V^*\|^2\mid\mathcal{Z}(t)\big\} &\le \mathbb{E}\big\{\|\boldsymbol{q}(t)-\boldsymbol{\gamma}_V^*\|^2\mid\mathcal{Z}(t)\big\}+(T_\nu^2+T_\nu)B^2\\ &\quad -2T_\nu\mathbb{E}\big\{\sum_{i=1}^{M}\frac{\|\mathcal{T}_{s_i}(t,T_\nu)\|}{T_\nu}\big[g_{s_i}(\boldsymbol{\gamma}_V^*)-g_{s_i}(\boldsymbol{q}(t))\big]\mid\mathcal{Z}(t)\big\}.\end{aligned}$$

Adding and subtracting $2T_\nu \sum_{i=1}^M \pi_{s_i}\big[g_{s_i}(\boldsymbol{\gamma}_V^*) - g_{s_i}(\boldsymbol{q}(t))\big]$ from the RHS, we have:

$$\begin{aligned}\mathbb{E}\big\{\|\boldsymbol{q}(t+T_\nu) - \boldsymbol{\gamma}_V^*\|^2 \mid \mathcal{Z}(t)\big\} \;\leq\; & \mathbb{E}\big\{\|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\|^2 \mid \mathcal{Z}(t)\big\} + (T_\nu^2 + T_\nu)B^2 \\ & -2T_\nu \sum_{i=1}^M \pi_{s_i}\big[g_{s_i}(\boldsymbol{\gamma}_V^*) - g_{s_i}(\boldsymbol{q}(t))\big] \\ & -2T_\nu \mathbb{E}\Big\{\sum_{i=1}^M \left[\frac{\|\mathcal{T}_{s_i}(t,T_\nu)\|}{T_\nu} - \pi_{s_i}\right] \times \\ & \qquad \big[g_{s_i}(\boldsymbol{\gamma}_V^*) - g_{s_i}(\boldsymbol{q}(t))\big] \mid \mathcal{Z}(t)\Big\}. \end{aligned} \tag{4.83}$$

Denote the term inside the last expectation of (4.83) as $\mathcal{Q}$, i.e.,

$$\mathcal{Q} = \sum_{i=1}^M \left[\frac{\|\mathcal{T}_{s_i}(t,T_\nu)\|}{T_\nu} - \pi_{s_i}\right]\big[g_{s_i}(\boldsymbol{\gamma}_V^*) - g_{s_i}(\boldsymbol{q}(t))\big]. \tag{4.84}$$

Using the fact that $g_{s_i}(\boldsymbol{\gamma}_V^*) - g_{s_i}(\boldsymbol{q}(t))$ is a constant given $\mathcal{Z}(t)$, we have:

$$\begin{aligned}\mathbb{E}\big\{\mathcal{Q} \mid \mathcal{Z}(t)\big\} \;&=\; \sum_{i=1}^M \left[\frac{\mathbb{E}\big\{\|\mathcal{T}_{s_i}(t,T_\nu)\| \mid \mathcal{Z}(t)\big\}}{T_\nu} - \pi_{s_i}\right] \times \big[g_{s_i}(\boldsymbol{\gamma}_V^*) - g_{s_i}(\boldsymbol{q}(t))\big] \\ &\leq\; \sum_{i=1}^M \left|\frac{\mathbb{E}\big\{\|\mathcal{T}_{s_i}(t,T_\nu)\| \mid \mathcal{Z}(t)\big\}}{T_\nu} - \pi_{s_i}\right| \times \big|g_{s_i}(\boldsymbol{\gamma}_V^*) - g_{s_i}(\boldsymbol{q}(t))\big|. \end{aligned}$$

By (4.76), $g_{s_i}(\boldsymbol{\gamma}_V^*) - g_{s_i}(\boldsymbol{q}(t)) \leq B\|\boldsymbol{\gamma}_V^* - \boldsymbol{q}(t)\|$, thus we have:

$$\begin{aligned}\mathbb{E}\big\{\mathcal{Q} \mid \mathcal{Z}(t)\big\} \;&\leq\; B\|\boldsymbol{\gamma}_V^* - \boldsymbol{q}(t)\| \times \sum_{i=1}^M \left|\frac{\mathbb{E}\big\{\|\mathcal{T}_{s_i}(t,T_\nu)\| \mid \mathcal{Z}(t)\big\}}{T_\nu} - \pi_{s_i}\right| \\ &\leq\; \nu B\|\boldsymbol{\gamma}_V^* - \boldsymbol{q}(t)\|, \end{aligned} \tag{4.85}$$

where the last step follows from the definition of T_ν. Now by (4.13) and (4.75):

$$\sum_{i=1}^M \pi_{s_i}\big[g_{s_i}(\boldsymbol{\gamma}_V^*) - g_{s_i}(\boldsymbol{q}(t))\big] = g(\boldsymbol{\gamma}_V^*) - g(\boldsymbol{q}(t)).$$

Plug this and (4.85) into (4.83), we have:

$$\begin{aligned}\mathbb{E}\big\{\|\boldsymbol{q}(t+T_\nu) - \boldsymbol{\gamma}_V^*\|^2 \mid \mathcal{Z}(t)\big\} \;\leq\; & \mathbb{E}\big\{\|\boldsymbol{q}(t) - \boldsymbol{\gamma}_V^*\|^2 \mid \mathcal{Z}(t)\big\} + (T_\nu^2 + T_\nu)B^2 \\ & -2T_\nu\big(g(\boldsymbol{\gamma}_V^*) - g(\boldsymbol{q}(t))\big) + 2T_\nu\nu B\|\boldsymbol{\gamma}_V^* - \boldsymbol{q}(t)\|. \end{aligned}$$

Recall that $\mathcal{Z}(t) = (\mathcal{H}(t), \boldsymbol{q}(t))$. Taking expectation over $\mathcal{H}(t)$ on both sides proves the lemma. □

4.10.2 Proof of (4.31)

Here we prove that for $\tilde{Y}(t)$ defined in the proof of part (b) of Theorem 5, we have for all $\tilde{Y}(t) \geq 0$ that:

$$\Delta_{T_\nu}(\tilde{Y}(t)) \quad \leq \quad e^{2wT_\nu B} - \frac{w\eta}{2} e^{w\tilde{Y}(t)}.$$

Proof. If $\tilde{Y}(t) > T_\nu B$, denote $\delta(t) = \tilde{Y}(t+T_\nu) - \tilde{Y}(t)$. It is easy to see that $|\delta(t)| \leq T_\nu B$. Rewrite (4.30) as:

$$\Delta_{T_\nu}(\tilde{Y}(t)) \quad = \quad e^{w\tilde{Y}(t)} \mathbb{E}\big\{ \big(e^{w\delta(t)} - 1\big) \mid \boldsymbol{q}(t) \big\}. \tag{4.86}$$

By a Taylor expansion, we have that:

$$e^{w\delta(t)} \quad = \quad 1 + w\delta(t) + \frac{w^2\delta^2(t)}{2} l(w\delta(t)), \tag{4.87}$$

where $l(y) = 2\sum_{k=2}^{\infty} \frac{y^{k-2}}{k!} = \frac{2(e^y - 1 - y)}{y^2}$ [CL] has the following properties:

1. $l(0) = 1$; $l(y) \leq 1$ for $y < 0$; $l(y)$ is monotone increasing for $y \geq 0$;

2. For $y < 3$,

$$l(y) = 2\sum_{k=2}^{\infty} \frac{y^{k-2}}{k!} \leq \sum_{k=2}^{\infty} \frac{y^{k-2}}{3^{k-2}} = \frac{1}{1 - y/3}.$$

Thus by (4.87) we have:

$$e^{w\delta(t)} \quad \leq \quad 1 + w\delta(t) + \frac{w^2 T_\nu^2 B^2}{2} l(wT_\nu B). \tag{4.88}$$

Plugging this into (4.86), and noting that $\tilde{Y}(t) > T_\nu B$, so by (4.29) we have $\mathbb{E}\big\{\delta(t) \mid \boldsymbol{q}(t)\big\} \leq -\eta$. Hence:

$$\Delta_{T_\nu}(\tilde{Y}(t)) \quad \leq \quad e^{w\tilde{Y}(t)} \big(- w\eta + \frac{w^2 T_\nu^2 B^2}{2} l(wT_\nu B) \big). \tag{4.89}$$

Choosing $w = \frac{\eta}{T_\nu^2 B^2 + T_\nu B\eta/3}$, we see that $wT_\nu B < 3$, thus:

$$\frac{w^2 T_\nu^2 B^2}{2} l(wT_\nu B) \leq \frac{w^2 T_\nu^2 B^2}{2} \frac{1}{1 - wT_\nu B/3} = \frac{w\eta}{2},$$

where the last equality follows since:

$$\begin{aligned} w = \frac{\eta}{T_\nu^2 B^2 + T_\nu B\eta/3} &\Rightarrow w(T_\nu^2 B^2 + T_\nu B\eta/3) = \eta \\ &\Rightarrow wT_\nu^2 B^2 = \eta - wT_\nu B\eta/3 \\ &\Rightarrow wT_\nu^2 B^2 \frac{1}{1 - wT_\nu B/3} = \eta. \end{aligned}$$

Therefore (4.89) becomes:

$$\Delta_{T_\nu}(\tilde{Y}(t)) \leq -\frac{w\eta}{2} e^{w\tilde{Y}(t)} \leq e^{2wT_\nu B} - \frac{w\eta}{2} e^{w\tilde{Y}(t)}. \tag{4.90}$$

Now if $\tilde{Y}(t) \leq T_\nu B$, it is easy to see that $\Delta_{T_\nu}(\tilde{Y}(t)) \leq e^{2wT_\nu B} - e^{w\tilde{Y}(t)} \leq e^{2wT_\nu B} - \frac{w\eta}{2} e^{w\tilde{Y}(t)}$, since $\tilde{Y}(t+T_\nu) \leq T_\nu B + \tilde{Y}(t) \leq 2T_\nu B$ and $\frac{w\eta}{2} \leq 1$, as $\eta < T_\nu B$. Therefore for all $\tilde{Y}(t) \geq 0$, we see that (4.31) holds. □

4.10.3 Proof of Lemma 4

Here we prove Lemma 4. To save space, we sometimes use $[a]^+$ to denote $\max[a, 0]$.

Proof. It suffices to show that (4.53) holds for a single queue j. Also, when $\mathcal{W}_j = 0$, (4.53) trivially holds, thus we only consider $\mathcal{W}_j > 0$.

Part (A): We first prove $q_j(t) \leq \max[W_j(t) - \mathcal{W}_j, 0] + \delta_{max}$. First we see that it holds at $t = 0$, since $W_j(0) = \mathcal{W}_j$ and $q_j(t) = 0$. It also holds for $t = 1$. Since $q_j(0) = 0$ and $W_j(0) = \mathcal{W}_j$, we have $q_j(1) = A_j(0) \leq \delta_{max}$. Thus we have $q_j(1) \leq \max[W_j(1) - \mathcal{W}_j, 0] + \delta_{max}$.

Now assume that $q_j(t) \leq \max[W_j(t) - \mathcal{W}_j, 0] + \delta_{max}$ holds for $t = 0, 1, 2, ..., k$, we want to show that it also holds for $t = k + 1$. We first note that if $q_j(k) \leq \mu_j(k)$, the the result holds since then $q_j(k+1) = [q_j(k) - \mu_j(k)]^+ + A_j(k) = A_j(k) \leq \delta_{max}$. Thus we only consider $q_j(k) \geq \mu_j(k)$ in the following:

(A-I) Suppose that $W_j(k) \geq \mathcal{W}_j$. Note that in this case we have:

$$q_j(k) \leq W_j(k) - \mathcal{W}_j + \delta_{max}. \tag{4.91}$$

Also, $q_j(t+1) = \max[q_j(t) - \mu_j(t), 0] + A_j(t)$. Since $q_j(k) \geq \mu_j(k)$, we have:

$$\begin{aligned} q_j(k+1) &= q_j(k) - \mu_j(k) + A_j(k) \\ &\leq W_j(k) - \mathcal{W}_j + \delta_{max} - \mu_j(k) + A_j(k) \\ &\leq [W_j(k) - \mu_j(k) + A_j(k) - \mathcal{W}_j]^+ + \delta_{max} \\ &\leq \left[[W_j(k) - \mu_j(k)]^+ + A_j(k) - \mathcal{W}_j\right]^+ + \delta_{max} \\ &= \max[W_j(k+1) - \mathcal{W}_j, 0] + \delta_{max}, \end{aligned}$$

where the first inequality is due to (4.91), the second and third inequalities are due to the $[a]^+$ operator, and the last equality follows from the definition of $W_j(k+1)$.

(A-II) Now suppose that $W_j(k) < \mathcal{W}_j$. In this case we have $q_j(k) \leq \delta_{max}$, $\tilde{A}_j(k) = [A_j(k) - \mathcal{W}_j + W_j(k)]^+$ and:

$$q_j(k+1) = [q_j(k) - \mu_j(k)]^+ + \tilde{A}_j(k).$$

First consider the case when $W_j(k) < \mathcal{W}_j - A_j(k)$. In this case $\tilde{A}_j(k) = 0$, so we have:

$$q_j(k+1) = q_j(k) - \mu_j(k) \leq \delta_{max} - \mu_j(k) \leq \delta_{max},$$

which implies $q_j(k+1) \leq \max[W_j(k+1) - \mathcal{W}_j, 0] + \delta_{max}$. Else if $\mathcal{W}_j - A_j(k) \leq W_j(k) < \mathcal{W}_j$, we have:

$$\begin{aligned} q_j(k+1) &= q_j(k) - \mu_j(k) + A_j(k) - \mathcal{W}_j + W_j(k) \\ &\leq W_j(k) - \mathcal{W}_j + \delta_{max} - \mu_j(k) + A_j(k) \\ &\leq \max[W_j(k+1) - \mathcal{W}_j, 0] + \delta_{max}, \end{aligned}$$

where the first inequality uses $q_j(k) \leq \delta_{max}$ and the second inequality follows as in (A-I).

Part (B): We now show that $q_j(t) \geq \max[W_j(t) - \mathcal{W}_j, 0]$. First we see that it holds for $t = 0$ since $W_j(0) = \mathcal{W}_j$. We also have for $t = 1$ that:

$$\begin{aligned}
[W_j(1) - \mathcal{W}_j]^+ &= \left[[W_j(0) - \mu_j(0)]^+ + A_j(0) - \mathcal{W}_j\right]^+ \\
&\leq \left[[W_j(0) - \mu_j(0) - \mathcal{W}_j]^+ + A_j(0)\right]^+ \\
&= A_j(0)
\end{aligned}$$

Thus $q_j(1) \geq \max[W_j(1) - \mathcal{W}_j, 0]$ since $q_j(1) = A_j(0)$. Now suppose that $q_j(t) \geq \max[W_j(t) - \mathcal{W}_j, 0]$ holds for $t = 0, 1, ..., k$, we will show that it holds for $t = k + 1$. We note that if $W_j(k+1) < \mathcal{W}_j$, then $\max[W_j(k+1) - \mathcal{W}_j, 0] = 0$ and we are done. So we consider $W_j(k+1) \geq \mathcal{W}_j$.

(B-I) First if $W_j(k) \geq \mathcal{W}_j$, we have $\tilde{A}_j(k) = A_j(k)$. Hence:

$$\begin{aligned}
[W_j(k+1) - \mathcal{W}_j]^+ &= [W_j(k) - \mu_j(k)]^+ + A_j(k) - \mathcal{W}_j \\
&\leq [W_j(k) - \mu_j(k) - \mathcal{W}_j]^+ + A_j(k) \\
&\leq [[W_j(k) - \mathcal{W}_j]^+ - \mu_j(k)]^+ + A_j(k) \\
&\leq [q_j(k) - \mu_j(k)]^+ + A_j(k),
\end{aligned}$$

where the first two inequalities are due to the $[a]^+$ operator and the last inequality is due to $q_j(k) \geq [W_j(k) - \mathcal{W}_j]^+$. This implies $[W_j(k+1) - \mathcal{W}_j]^+ \leq q_j(k+1)$.

(B-II) Suppose that $W_j(k) < \mathcal{W}_j$. Since $W_j(k+1) \geq \mathcal{W}_j$, we have $\mathcal{W}_j - A_j(k) \leq W_j(k) < \mathcal{W}_j$, for otherwise $W_j(k) < \mathcal{W}_j - A_j(k)$ and $W_j(k+1) = [W_j(k) - \mu_j(t)]^+ + A_j(t) < \mathcal{W}_j$. Hence in this case $\tilde{A}_j(k) = A_j(k) - \mathcal{W}_j + W_j(k) \geq 0$.

$$\begin{aligned}
[W_j(k+1) - \mathcal{W}_j]^+ &= [W_j(k) - \mu_j(k)]^+ + A_j(k) - \mathcal{W}_j \\
&\leq [W_j(k) + q_j(k) - \mu_j(k)]^+ + A_j(k) - \mathcal{W}_j \\
&\leq [q_j(k) - \mu_j(k)]^+ + A_j(k) - \mathcal{W}_j + W_j(k)
\end{aligned}$$

$$= \quad q_j(k+1),$$

where the two inequalities are due to the fact that $q_j(k) \geq 0$ and $W_j(k) \geq 0$. □

4.10.4 Proof of Theorem 11

Here we provide the proof of Theorem 11.

Proof. Consider a sample path for which the lim inf arrival rate is at least λ_{min} and for which we have an infinite number of departures (this happens with probability 1 by assumption). There must be a non-empty subset of $\mathcal{B}$ consisting of buffer locations that experience an infinite number of departures. Call this subset $\tilde{\mathcal{B}}$. Now for a buffer slot $b \in \tilde{\mathcal{B}}$, let $W_i^{(b)}$ be the delay of the i^{th} departure from b, and let $D^{(b)}(t)$ denote the number of departures from b up to time t, and use $Q^{(b)}(t)$ to denote the occupancy of the buffer slot b at time t. Note that $Q^{(b)}(t)$ is either 0 or 1. For all $t \geq 0$, it can be shown that:

$$\sum_{i=1}^{D^{(b)}(t)} W_i^{(b)} \leq \int_0^t Q^{(b)}(\tau) d\tau. \tag{4.92}$$

This can be seen from Fig. 4.11 below.

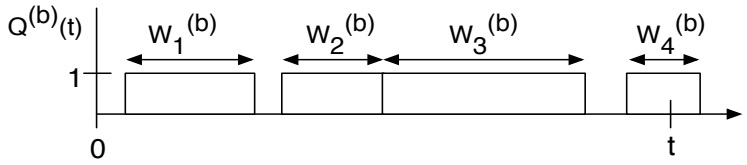

Figure 4.11: An illustration of inequality (4.92) for a particular buffer location b. At time t in the figure, we have $D^{(b)}(t) = 3$.

Therefore, we have:

$$\begin{aligned} \sum_{b \in \tilde{\mathcal{B}}} \sum_{i=1}^{D^{(b)}(t)} W_i^{(b)} &\leq \int_0^t \sum_{i \in \tilde{\mathcal{B}}} Q^{(b)}(\tau) d\tau \\ &\leq \int_0^t |\tilde{\mathcal{B}}| d\tau \end{aligned}$$

$$\leq \quad |\mathcal{B}|t. \tag{4.93}$$

The left-hand-side of the above inequality is equal to the sum of all delays of jobs that depart from locations in $\tilde{\mathcal{B}}$ up to time t. All other buffer locations (in $\mathcal{B}$ but not in $\tilde{\mathcal{B}}$) experience only a finite number of departures. Let $\mathcal{J}$ represent an index set that indexes all of the (finite number) of jobs that depart from these other locations. Note that the delay W_j for each job $j \in \mathcal{J}$ is finite (because, by definition, job j eventually departs). We thus have:

$$\sum_{i=1}^{D(t)} W_i \leq \sum_{b\in\tilde{\mathcal{B}}} \sum_{i=1}^{D^{(b)}(t)} W_i^{(b)} + \sum_{j\in\mathcal{J}} W_j,$$

where the inequality is because the second term on the right-hand-side sums over jobs in $\mathcal{J}$, and these jobs may not have departed by time t. Combining the above and (4.93) yields for all $t \geq 0$:

$$\sum_{i=1}^{D(t)} W_i \leq |\mathcal{B}|t + \sum_{j\in\mathcal{J}} W_j.$$

Dividing by $D(t)$ yields:

$$\frac{1}{D(t)} \sum_{i=1}^{D(t)} W_i \leq \frac{|\mathcal{B}|t}{D(t)} + \frac{1}{D(t)} \sum_{j\in\mathcal{J}} W_j.$$

Taking a lim sup as $t \to \infty$ yields:

$$\limsup_{t\to\infty} \frac{1}{D(t)} \sum_{i=1}^{D(t)} W_i \leq \limsup_{t\to\infty} \frac{|\mathcal{B}|t}{D(t)}, \tag{4.94}$$

where we have used the fact that $\sum_{j\in\mathcal{J}} W_j$ is a finite number, and $D(t) \to \infty$ as $t \to \infty$, so that:

$$\limsup_{t\to\infty} \frac{1}{D(t)} \sum_{j\in\mathcal{J}} W_j = 0.$$

Now note that, because each buffer location in $\mathcal{B}$ can hold at most one job, the number of departures $D(t)$ is at least $N(t) - |\mathcal{B}|$, which is a positive number for sufficiently large t. Thus:

$$\begin{aligned}\limsup_{t\to\infty} \frac{|\mathcal{B}|t}{D(t)} &\leq \limsup_{t\to\infty} \left[\frac{|\mathcal{B}|t}{N(t)-|\mathcal{B}|}\right] \\ &= \limsup_{t\to\infty} \left[\frac{|\mathcal{B}|}{N(t)/t - |\mathcal{B}|/t}\right] \\ &\leq |\mathcal{B}|/\lambda_{min}.\end{aligned}$$

Using this in (4.94) proves the result. □

4.10.5 $[O(1/V), O(V)]$ performance of QLA under Markovian dynamics

In this section, we prove that under the Markovian network state dynamics, QLA achieves an exact $[O(\frac{1}{V}), O(V)]$ utility-delay tradeoff for the stochastic problem. This is the first formal proof of this result. It generalizes the $[O(\frac{1}{V}), O(V)]$ performance result of QLA in the i.i.d. case in [GNT06]. To prove the result, we use a variable multi-slot Lyapunov drift argument. Different from previous multi-slot drift arguments, e.g., [NMR05] and [Nee10b], where the drift is usually computed over a fixed number of slots, the slot number here is a random variable corresponding to the return time of the network states. As we will see, this variable multi-slot drift analysis allows us to obtain the exact $[O(\frac{1}{V}), O(V)]$ utility-delay tradeoff for QLA. Moreover, it also allows us to state QLA's performance in terms of explicit parameters of the Markovian $S(t)$ process.

In the following, we define $T_i(t_0)$ to be the first return time of $S(t)$ to state s_i given that $S(t_0) = s_i$, i.e.,

$$T_i(t_0) = \inf\{T > 0, \, s.t. \, S(t_0 + T) = s_i \,|\, S(t_0) = s_i\}.$$

We see that $T_i(t_0)$ has the same distribution for all t_0. Thus, we use $\overline{T_i}$ to denote the expected value of $T_i(t)$ for any ts.t.$S(t) = s_i$ and use $\overline{T_i^2}$ to denote its second moment. By Theorem 3 in Chapter 5 of [Gal96], we have for all states s_i that:

$$\overline{T_i} = \frac{1}{\pi_{s_i}} < \infty, \tag{4.95}$$

i.e., the expected return time of any state s_i is finite. In the following, we also use $T_{ji}(t_0)$ to denote the first hitting time for $S(t)$ to enter the state s_i given that $S(t_0) = s_j$. It is again easy to see that $T_{ji}(t_0)$ has the same distribution at all t_0. Hence we similarly use $\overline{T_{ji}}$ and $\overline{T_{ji}^2}$ to denote its first and second moments. Throughout the analysis, we make the following assumption:

Assumption 2. *There exists a state s_1 such that:*

$$\overline{T_{j1}^2} < \infty, \quad \forall\ j.$$

That is, starting from any state s_j (including s_1), the random time needed to get into state s_1 has a finite second moment. This condition is not very restrictive and can be satisfied in many cases, e.g., when $\mathcal{S}$ is finite.

We now have the following theorem summarizing QLA's performance under the Markovian network state dynamics:

Theorem 15. *Suppose (3.1) holds. Then under the Markovian network state process $S(t)$, the QLA algorithm achieves the following:*

$$f_{av}^{QLA} \leq f_{av}^* + \frac{CB^2}{V\overline{T_1}}, \tag{4.96}$$

$$\overline{\sum_{j=1}^{r} q_j} \leq \frac{CB^2 + \overline{T_1}V\delta_{max}}{\eta} + \frac{DB^2}{2}, \tag{4.97}$$

where $\eta > 0$ is the slack parameter defined in (3.1) in Section 3.1.2, and C, D are defined as:

$$C = \overline{T_1^2} + \overline{T_1},\ D = \overline{T_1^2} - \overline{T_1}, \tag{4.98}$$

i.e., C and D are the sum and difference of the first and second moments of the return time associated with s_1.

Note that $\eta, C, D = \Theta(1)$ in (4.97), i.e., independent of V. Hence Theorem 15 shows that QLA indeed achieves an exact $[O(1/V), O(V)]$ utility-delay tradeoff for general stochastic network optimization problems with Markovian network dynamics. Although our bounds may be loose when the number of states is large, we note that Theorem 15 also applies to the case when $S(t)$ evolves according to a Markov modulated i.i.d. process, in which case there is a Markov chain of only a few states, but in each Markov state, there can be many i.i.d. randomness. For example, suppose $S(t)$ is i.i.d. with 10^4 states. Then we can view $S(t)$ as having one Markov state, but within the Markov state, it has 10^4 i.i.d. random choices. In this case, Theorem 15 will apply with $C = 2$ and $D = 0$. These Markov modulated processes can easily be incorporated into our analysis by taking expectation over the i.i.d. randomness of the current Markov state in Equation (4.99). These Markov modulated processes are important in stochastic modeling and include the ON/OFF processes for modeling time-correlated arrivals processes, e.g., [Nee09a].

Proof. (Theorem 15) To prove the theorem, we first define the Lyapunov function $L(t) = \frac{1}{2}\sum_{j=1}^{r} q_j^2(t)$. By using the queueing dynamic equation (3.5), it is easy to obtain that:

$$\frac{1}{2}q_j^2(t+1) - \frac{1}{2}q_j^2(t) \leq \delta_{max}^2 + q_j(t)[A_j(t) - \mu_j(t)].$$

Summing over all $j = 1, ..., r$ and adding to both sides the term $Vf(t)$, we obtain:

$$L(t+1) - L(t) + Vf(t) \leq B^2 + \left\{ Vf(t) + \sum_{j=1}^{r} q_j(t)[A_j(t) - \mu_j(t)] \right\}. \quad (4.99)$$

We see from (4.8) then given the network state $S(t)$, QLA chooses an action to *minimize the right-hand side (RHS) at time t*. Now compare the term in {} in the RHS of (4.99) with (4.75), we see that we indeed have:

$$L(t+1) - L(t) + Vf^Q(t) \quad \leq \quad B^2 + g_{S(t)}(\boldsymbol{q}(t)), \quad (4.100)$$

where we use $f^Q(t) = f(x^{QLA}(t))$ to denote the utility incurred by QLA's action at time t, and $g_{S(t)}(\cdot)$ is the function (4.75) with the network state being $S(t)$.

(Part A: Proof of Utility) We first prove the utility performance. Consider $t = 0$ and first assume that $S(0) = s_1$. Summing up the inequality (4.100) from time $t = 0$ to time $t = T_1(0) - 1$, we have:

$$L(T_1(0)) - L(0) + \sum_{t=0}^{T_1(0)-1} Vf^Q(t) \leq T_1(0)B^2 + \sum_{t=0}^{T_1(0)-1} g_{S(t)}(\boldsymbol{q}(t)).$$

This can be rewritten as:

$$\begin{aligned} L(T_1(0)) - L(0) + \sum_{t=0}^{T_1(0)-1} Vf^Q(t) &\leq T_1(0)B^2 \quad (4.101) \\ + \sum_{t=0}^{T_1(0)-1} g_{S(t)}(\boldsymbol{q}(0)) &+ \sum_{t=0}^{T_1(0)-1} \big[g_{S(t)}(\boldsymbol{q}(t)) - g_{S(t)}(\boldsymbol{q}(0)) \big]. \end{aligned}$$

Using (4.75) and the fact that $||\boldsymbol{q}(t+\tau) - \boldsymbol{q}(t)|| \leq \tau B$, we see that the final term can be bounded by:

$$\left| \sum_{t=0}^{T_1(0)-1} \big[g_{S(t)}(\boldsymbol{q}(t)) - g_{S(t)}(\boldsymbol{q}(0)) \big] \right| \leq \sum_{t=0}^{T_1(0)-1} tB^2 = \big[\frac{1}{2}(T_1(0))^2 - \frac{1}{2}T_1(0) \big] B^2.$$

Plugging this into (4.101), and letting $\hat{C} = \frac{1}{2}(T_1(0))^2 + \frac{1}{2}T_1(0)$, we obtain:

$$\begin{aligned} L(T_1(0)) - L(0) + \sum_{t=0}^{T_1(0)-1} Vf^Q(t) &\leq \hat{C}B^2 + \sum_{t=0}^{T_1(0)-1} g_{S(t)}(\boldsymbol{q}(0)) \\ &= \hat{C}B^2 + \sum_{s_i} n_{s_i}^{T_1(0)}(0) g_{s_i}(\boldsymbol{q}(0)). \end{aligned}$$

Here $n_{s_i}^{T_1(0)}(t_0)$ denotes the number of times the network state s_i appears in the period $[t_0, t_0 + T_1(0) - 1]$. Now we take expectations over $T_1(0)$ on both sides conditioning on $S(0) = s_1$ and $\boldsymbol{q}(0)$, we have:

$$\mathbb{E}\big\{L(T_1(0)) - L(0) \mid S(0), \boldsymbol{q}(0)\big\} + \mathbb{E}\big\{\sum_{t=0}^{T_1(0)-1} Vf^Q(t) \mid S(0), \boldsymbol{q}(0)\big\} \tag{4.102}$$
$$\leq CB^2 + \sum_{s_i} \mathbb{E}\big\{n_{s_i}^{T_1(0)}(0) \mid S(0), \boldsymbol{q}(0)\big\} g_{s_i}(\boldsymbol{q}(0)).$$

Here $C = \mathbb{E}\big\{\hat{C} \mid S(0), \boldsymbol{q}(0)\big\} = \frac{1}{2}[\overline{T_1^2} + \overline{T_1}]$. The above equation uses the fact that $g_{s_i}(\boldsymbol{q}(0))$ is a constant given $\boldsymbol{q}(0)$. Now by Theorem 2 in Page 154 of [Gal96] we have that:

$$\mathbb{E}\big\{n_{s_i}^{T_i(0)}(0) \mid S(0), \boldsymbol{q}(0)\big\} = \frac{\pi_{s_i}}{\pi_{s_1}}. \tag{4.103}$$

Plug this into (4.102), we have:

$$\mathbb{E}\big\{L(T_1(0)) - L(0) \mid S(0), \boldsymbol{q}(0)\big\} + \mathbb{E}\big\{\sum_{t=0}^{T_1(0)-1} Vf^Q(t) \mid S(0), \boldsymbol{q}(0)\big\} \tag{4.104}$$
$$\leq CB^2 + \frac{1}{\pi_{s_1}} \sum_{s_i} \pi_{s_i} g_{s_i}(\boldsymbol{q}(0)).$$

Now using (4.13) and (4.95), i.e., $\overline{T_1} = 1/\pi_{s_1}$ and $g(\boldsymbol{\gamma}) = \sum_{s_i} \pi_{s_i} g_{s_i}(\boldsymbol{\gamma})$, we obtain:

$$\mathbb{E}\big\{L(T_1(0)) - L(0) \mid S(0), \boldsymbol{q}(0)\big\} + \mathbb{E}\big\{\sum_{t=0}^{T_1(0)-1} Vf^Q(t) \mid S(0), \boldsymbol{q}(0)\big\} \tag{4.105}$$
$$\leq CB^2 + \overline{T_1} g(\boldsymbol{q}(0)).$$

It is shown in [HN10a] that $g(\boldsymbol{q}(0)) \leq g(\boldsymbol{\gamma}^*) \leq Vf_{av}^*$. Thus we conclude that:

$$\mathbb{E}\big\{L(T_1(0)) - L(0) \mid S(0), \boldsymbol{q}(0)\big\} + \mathbb{E}\big\{\sum_{t=0}^{T_1(0)-1} Vf^Q(t) \mid S(0), \boldsymbol{q}(0)\big\} \tag{4.106}$$
$$\leq CB^2 + \overline{T_1} Vf_{av}^*.$$

More generally, if $t_k = t_{k-1} + T_i(t_{k-1})$ with $t_0 = 0$ is the k^{th} time after time 0 when $S(t) = s_1$, we have:

$$\mathbb{E}\big\{L(t_{k+1}) - L(t_k) \mid S(t_k), \boldsymbol{q}(t_k)\big\} + \mathbb{E}\big\{\sum_{t=t_k}^{t_{k+1}-1} Vf^Q(t) \mid S(t_k), \boldsymbol{q}(t_k)\big\} \tag{4.107}$$
$$\leq CB^2 + \overline{T_1} Vf_{av}^*,$$

Now taking expectations over $\boldsymbol{q}(t_k)$ on both sides, we have:

$$\mathbb{E}\big\{L(t_{k+1}) - L(t_k) \mid S(t_k)\big\} + \mathbb{E}\big\{\sum_{t=t_k}^{t_{k+1}-1} Vf^Q(t) \mid S(t_k)\big\} \leq CB^2 + \overline{T_1}Vf^*_{av}.$$

Note that given $S(0) = s_1$, we have the complete information of $S(t_k)$ for all k. Hence the above is the same as:

$$\mathbb{E}\big\{L(t_{k+1}) - L(t_k) \mid S(0)\big\} + \mathbb{E}\big\{\sum_{t=t_k}^{t_{k+1}-1} Vf^Q(t) \mid S(0)\big\} \leq CB^2 + \overline{T_1}Vf^*_{av}. \quad (4.108)$$

Summing the above from $k = 0$ to $K - 1$, we get:

$$\mathbb{E}\big\{L(t_K) - L(0) \mid S(0) = s_1\big\} + \mathbb{E}\big\{\sum_{t=0}^{t_K-1} Vf^Q(t) \mid S(0) = s_1\big\} \quad (4.109)$$
$$\leq KCB^2 + K\overline{T_1}Vf^*_{av}.$$

Using the facts that $|f(t)| \leq \delta_{max}$, $\lceil K\overline{T_1}\rceil \leq K\overline{T_1} + 1$, $L(0) = 0$ and $L(t) \geq 0$ for all t, we have:

$$\mathbb{E}\big\{\sum_{t=0}^{\lceil K\overline{T_1}\rceil-1} Vf^Q(t) \mid S(0) = s_1\big\} \leq KCB^2 + K\overline{T_1}Vf^*_{av} + V\delta_{max}$$
$$+V\delta_{max}\mathbb{E}\big\{|K\overline{T_1} - t_K| \mid S(0) = s_1\big\}.$$

Dividing both sides by $V\lceil K\overline{T_1}\rceil$, we get:

$$\frac{1}{\lceil K\overline{T_1}\rceil}\mathbb{E}\big\{\sum_{t=0}^{\lceil K\overline{T_1}\rceil-1} f^Q(t) \mid S(0) = s_1\big\} \leq \frac{CB^2K}{V\lceil K\overline{T_1}\rceil} + f^*_{av}\frac{K\overline{T_1}V}{\lceil K\overline{T_1}\rceil} + \frac{V\delta_{max}}{\lceil K\overline{T_1}\rceil} \quad (4.110)$$
$$+\mathbb{E}\big\{\big|\frac{t_K - K\overline{T_1}}{K}\big| \mid S(0) = s_1\big\}\frac{K\delta_{max}}{\lceil K\overline{T_1}\rceil}.$$

Since $t_K = \sum_{k=0}^{K-1} T_i(t_k)$ with $t_0 = 0$, and each $T_1(t_k)$ is i.i.d. distributed with mean $\overline{T_1}$ and second moment $\overline{T_1^2} < \infty$, we have:

$$\Big[\mathbb{E}\big\{\big|\frac{t_K - K\overline{T_1}}{K}\big| \mid S(0) = s_1\big\}\Big]^2 \leq \mathbb{E}\big\{\big|\frac{t_K - K\overline{T_1}}{K}\big|^2 \mid S(0) = s_1\big\} \leq \frac{\overline{T_1^2}}{K}. \quad (4.111)$$

This implies that the term $\mathbb{E}\{\left|\frac{t_K - K\overline{T_1}}{K}\right| \mid S(0) = s_1\} \to 0$ as $K \to \infty$. It is also easy to see that $\lceil K\overline{T_1} \rceil \to \infty$ and $\frac{K}{\lceil K\overline{T_1} \rceil} \to \frac{1}{\overline{T_1}}$ as $K \to \infty$. Thus using (4.111) and taking a limsup as $K \to \infty$ in (4.110), we have:

$$\limsup_{K\to\infty} \frac{1}{\lceil K\overline{T_1} \rceil} \mathbb{E}\{ \sum_{t=0}^{\lceil K\overline{T_1} \rceil - 1} f^Q(t) \mid S(0) = s_1\} \leq \frac{CB^2}{V\overline{T_1}} + f^*_{av}.$$

Now consider the case when the starting state is $s_j \neq s_1$. In this case, let $T_{j1}(0)$ be the first time the system enters state s_1. Then we see that the above argument can be repeated for the system starting at time $T_{j1}(0)$. The only difference is that now the "initial" backlog in this case is given by $\boldsymbol{q}(T_{j1}(0))$. Specifically, we have from (4.109) that:

$$\mathbb{E}\{L(\hat{t_K}) - L(T_{j1}(0)) \mid T_{j1}(0), S(0) = s_j\} + \mathbb{E}\{ \sum_{t=T_{j1}(0)}^{\hat{t_K}-1} V f^Q(t) \mid T_{j1}(0), S(0) = s_j\} \leq KCB^2 + K\overline{T_1} V f^*_{av}.$$

Here $\hat{t_K}$ is the K^{th} return time of $S(t)$ to s_1 after time $T_{j1}(0)$. We thus obtain:

$$V\mathbb{E}\{ \sum_{t=T_{j1}(0)}^{\hat{t_K}-1} f^Q(t) \mid T_{j1}(0), S(0) = s_j\} \leq KCB^2 + K\overline{T_1} V f^*_{av} + \mathbb{E}\{L(T_{j1}(0)) \mid T_{j1}(0), S(0) = s_j\}.$$

However, since the increment of each queue is no more than δ_{max} every time slot, we see that $L(T_{j1}(0)) \leq [T_{j1}(0)]^2 B^2/2$. Also using the fact that $|f(t)| \leq \delta_{max}$ for all $0 \leq t \leq T_{j1}(0)$, we have:

$$V\mathbb{E}\{ \sum_{t=0}^{\hat{t_K}-1} f^Q(t) \mid T_{j1}(0), S(0) = s_j\} \leq KCB^2 + K\overline{T_1} V f^*_{av} + [T_{j1}(0)]^2 B^2/2 + T_{j1}(0) V \delta_{max}.$$

Now taking expectations over $T_{j1}(0)$ on both sides, and using a similar argument as (4.110), we get that for any starting state s_j, we have:

$$\limsup_{K\to\infty}\frac{1}{\lceil\overline{T_{j1}}+K\overline{T_1}\rceil}\mathbb{E}\Big\{\sum_{t=0}^{\lceil\overline{T_{j1}}+K\overline{T_1}\rceil-1}f^Q(t)\mid S(0)=s_j\Big\}\leq\frac{CB^2}{V\overline{T_1}}+f^*_{av}.$$

This proves the utility part (4.96).

(Part B: Proof of Backlog) Now we look at the backlog performance of QLA. We similarly first assume that $S(0)=s_1$. Recall that equation (4.105) says:

$$\mathbb{E}\big\{L(T_1(0))-L(0)\mid S(0),\boldsymbol{q}(0)\big\}+\mathbb{E}\Big\{\sum_{t=0}^{T_1(0)-1}Vf^Q(t)\mid S(0),\boldsymbol{q}(0)\Big\}\leq CB^2+\overline{T_1}g(\boldsymbol{q}(0)). \tag{4.112}$$

Now we define the following "convexified" dual function $g_c(\boldsymbol{\gamma})$:

$$g_c(\boldsymbol{\gamma})=\inf_{x_k^{(s_i)}\in\mathcal{X}^{(s_i)},a_k^{(s_i)}}\sum_{s_i}\pi_{s_i}\Bigg\{V\sum_{k=1}^{r+2}a_k^{(s_i)}f(s_i,x_k^{(s_i)})+\sum_j\gamma_j\Big[\sum_{k=1}^{r+2}a_k^{(s_i)}A_j(s_i,x_k^{(s_i)})-\sum_{k=1}^{r+2}a_k^{(s_i)}\mu_j(s_i,x_k^{(s_i)})\Big]\Bigg\}. \tag{4.113}$$

Here the $a_k^{(s_i)}$ variables are chosen to from the set: $\{a_k^{(s_i)}\geq 0,\sum_k a_k^{(s_i)}=1,\forall\, s_i\}$. Now by comparing (4.113) and (4.13), we see that $g_c(\boldsymbol{\gamma})=g(\boldsymbol{\gamma})$ for all $\boldsymbol{\gamma}\succeq\mathbf{0}$. This is so because at any $\boldsymbol{\gamma}\succeq\mathbf{0}$, we first have $g_c(\boldsymbol{\gamma})\leq g(\boldsymbol{\gamma})$, due to the use of the $a_k^{(s_i)}$ variables. And if $\{x^{(s_i)}\}_{i=1}^{\infty}$ are the minimizers of $g(\boldsymbol{\gamma})$, then $\{x_k^{(s_i)}\}_{i=1,2,\ldots}^{k=1,\ldots,r+2}$, with $x_k^{(s_i)}=x^{(s_i)}$, $a_1^{(s_i)}=1$ and $a_k^{(s_i)}=0$ if $k\neq 1$, will also be the minimizers of $g_c(\boldsymbol{\gamma})$.

Using the definition of $g_c(\boldsymbol{\gamma})$ defined in (4.113), and plugging the set of variables $\{\vartheta_k^{(s_i)}\}_{i=1,2,\ldots}^{k=1,2,\ldots,r+2}$ and the set of actions $\{x_k^{(s_i)}\}_{i=1,2,\ldots}^{k=1,2,\ldots,r+2}$ in the slackness assumption (3.1), and using the facts that $g(\boldsymbol{\gamma})=g_c(\boldsymbol{\gamma})$ and $0\leq f(t)\leq\delta_{max}$, it can be shown that $g(\boldsymbol{\gamma})$ satisfies:

$$g(\boldsymbol{\gamma})\leq V\delta_{max}-\eta\sum_{j=1}^{r}\gamma_j. \tag{4.114}$$

Using this in (4.112), we have:

$$\mathbb{E}\big\{L(T_1(0)) - L(0) \mid S(0), \boldsymbol{q}(0)\big\} + \mathbb{E}\Big\{\sum_{t=0}^{T_1(0)-1} Vf^Q(t) \mid S(0), \boldsymbol{q}(0)\Big\} \leq CB^2 + \overline{T_1}V\delta_{max} - \overline{T_1}\eta\sum_{j=1}^{r} q_j(0). \tag{4.115}$$

More generally, we have:

$$\mathbb{E}\big\{L(t_{k+1}) - L(t_k) \mid S(t_k), \boldsymbol{q}(t_k)\big\} \leq CB^2 + \overline{T_1}V\delta_{max} - \overline{T_1}\eta\sum_{j=1}^{r} q_j(t_k). \tag{4.116}$$

Here t_k is the k^{th} return time of $S(t)$ to state s_1 after time 0. Taking expectations on both sides over $\boldsymbol{q}(t_k)$ and rearranging the terms, we get:

$$\mathbb{E}\big\{L(t_{k+1}) - L(t_k) \mid S(t_k)\big\} + \overline{T_1}\eta\sum_{j=1}^{r}\mathbb{E}\big\{q_j(t_k) \mid S(t_k)\big\} \leq CB^2 + \overline{T_1}V\delta_{max}. \tag{4.117}$$

Now using the fact that conditioning on $S(t_k)$ is the same as conditioning on $S(0)$, we have:

$$\mathbb{E}\big\{L(t_{k+1}) - L(t_k) \mid S(0)\big\} + \overline{T_1}\eta\sum_{j=1}^{r}\mathbb{E}\big\{q_j(t_k) \mid S(0)\big\} \leq CB^2 + \overline{T_1}V\delta_{max}. \tag{4.118}$$

Summing over $k = 0, ..., K-1$, rearranging the terms, and using the facts that $L(0) = 0$ and $L(t) \geq 0$ for all t:

$$\sum_{k=0}^{K-1}\overline{T_1}\eta\sum_{j=1}^{r}\mathbb{E}\big\{q_j(t_k) \mid S(0)\big\} \leq KCB^2 + K\overline{T_1}V\delta_{max}. \tag{4.119}$$

Dividing both sides by $K\overline{T_1}\eta$, we get:

$$\frac{1}{K}\sum_{k=0}^{K-1}\sum_{j=1}^{r}\mathbb{E}\big\{q_j(t_k) \mid S(0)\big\} \leq \frac{CB^2 + \overline{T_1}V\delta_{max}}{\overline{T_1}\eta}. \tag{4.120}$$

Now using the fact that $|q_j(t+\tau) - q_j(t)| \leq \tau\delta_{max}$, we have:

$$\sum_{\tau=t_k}^{t_{k+1}-1}\sum_{j=1}^{r} q_j(\tau) \leq T_1(t_k)\sum_{j=1}^{r} q_j(t_k) + [\frac{1}{2}(T_1(t_k))^2 - \frac{1}{2}T_1(t_k)]B^2.$$

Taking expectations on both sides conditioning on $S(0)$ (which is the same as conditioning on $S(t_k)$), we get:

$$\mathbb{E}\Big\{\sum_{\tau=t_k}^{t_{k+1}-1}\sum_{j=1}^{r} q_j(\tau) \mid S(0)\Big\} \leq \mathbb{E}\Big\{T_1(t_k)\sum_{j=1}^{r} q_j(t_k) \mid S(0)\Big\} + [\frac{1}{2}\overline{T_1^2} - \frac{1}{2}\overline{T_1}]B^2$$
$$= \overline{T_1}\mathbb{E}\Big\{\sum_{j=1}^{r} q_j(t_k) \mid S(0)\Big\} + [\frac{1}{2}\overline{T_1^2} - \frac{1}{2}\overline{T_1}]B^2.$$

In the last step, we have used the fact that $T_1(t_k)$ is independent of $\boldsymbol{q}(t_k)$. Summing the above equation over $k = 0, 1, ..., K-1$, we have:

$$\mathbb{E}\Big\{\sum_{t=0}^{t_K-1}\sum_{j=1}^{r} q_j(t) \mid S(0)\Big\} \leq \sum_{k=0}^{K-1}\overline{T_1}\mathbb{E}\Big\{\sum_{j=1}^{r} q_j(t_k) \mid S(0)\Big\} + \frac{K[\overline{T_1^2} - \overline{T_1}]B^2}{2}.$$

Dividing both sides by K and using (4.120), we have:

$$\frac{1}{K}\mathbb{E}\Big\{\sum_{t=0}^{t_K-1}\sum_{j=1}^{r} q_j(t) \mid S(0)\Big\} \leq \frac{\overline{T_1}}{K}\sum_{k=0}^{K-1}\mathbb{E}\Big\{\sum_{j=1}^{r} q_j(t_k) \mid S(0)\Big\} + \frac{[\overline{T_1^2} - \overline{T_1}]B^2}{2}$$
$$\leq \frac{CB^2 + \overline{T_1}V\delta_{max}}{\eta} + \frac{[\overline{T_1^2} - \overline{T_1}]B^2}{2}.$$

Now notice that we always have $t_K \geq K$. Hence:

$$\frac{1}{K}\sum_{t=0}^{K-1}\mathbb{E}\Big\{\sum_{j=1}^{r} q_j(t) \mid S(0)\Big\} \leq \frac{1}{K}\mathbb{E}\Big\{\sum_{t=0}^{t_K-1}\sum_{j=1}^{r} q_j(t) \mid S(0)\Big\}$$
$$\leq \frac{CB^2 + \overline{T_1}V\delta_{max}}{\eta} + \frac{[\overline{T_1^2} - \overline{T_1}]B^2}{2}.$$

This proves (4.97) for the case when $S(0) = s_1$. The case when $S(0) = s_j \neq s_1$ can be treated in a similar way as in Part A. It can be shown that the above backlog bound still holds, as the effect of the backlog values before the first hitting time $T_{j1}(0)$ will vanish as time increases. This proves the backlog bound (4.97). Theorem 15 thus follows by combining the two proofs. □

4.10.6 Proof of Theorem 14

Here we prove Theorem 14.

Proof. First we see that under the conditions of Theorem 14, the network is stable. Thus the average rate out of any queue j is equal to λ_j, which is the total input rate. That is, letting $\mu_j(t)$ be the number of packets that depart from queue j at time t, we have:

$$\mu_j \triangleq \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mu_j(\tau) = \lambda_j.$$

Now we use $e_j(t)$ to denote the event that queue j decides to serve packets from the end of the queue, and let μ_j^E be the time average arrival rate of the packets that are served *during the times when queue j serves the end of the queue.* It is easy to see that:

$$\begin{aligned}
\mu_j^E &\triangleq \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mu_j(\tau)1_{[e_j(\tau)]} \\
&\overset{(*)}{=} \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\{\mu(\tau)1_{[e_j(\tau)]}\} \\
&= \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\{\mu_j(\tau)\}p = p\lambda_j.
\end{aligned}$$

Here $1_{[\cdot]}$ is the indicator function, and $(*)$ follows from the fact that the limit exists, that $0 \leq \mu(\tau)1_{[e(\tau)]} \leq \delta_{max}$, and the Lebesgue Dominated Convergence Theorem [Fol99]. As a result, the average rate of the packets that are served when the queue serves the front of the queue, denoted by μ_j^F, is $(1-p)\lambda_j$. Thus, if $0 \leq p < 1$, we see that $\mu_j^E, \mu_j^F > 0$. This implies that every packet will eventually leave the queue. To see this, suppose this is not the case. Then there exists at least one packet P^* that never leaves the queue. In this case, for any finite number $K > 0$, there must be more than K packets in the queue when P^* arrives, since the packets are drained out from the front with an average rate $\mu_j^F > 0$. However, this implies that the queue size must be infinite when P^* arrives, which contradicts the fact that the queue is stable. This proves Part (a).

Now define $\mathbb{P}_{j0}$ to be the set of packets that *(i) arrive to the queue when $q_j(t) \in \tilde{\mathcal{B}}_j$, and (ii) are served before they move to the right of $\tilde{Q}_{j,Low}$ and (iii) are served when the*

queue is serving the end of the queue. Let $\lambda_{\mathbb{P}_{j0}}$ be the average rate of the packets in $\mathbb{P}_{j0}$. It is easy to see that $\lambda_{\mathbb{P}_{j0}} \leq p\lambda_j$. We want to show that:

$$\lambda_{\mathbb{P}_{j0}} \geq \left[p\lambda_j - O(\frac{\delta_{max}c_1^*}{V\log(V)})\right]^+. \tag{4.121}$$

Then using the fact that the packets from $\mathbb{P}_{j0}$ occupy an interval of size at most $Q_{j,\text{High}} + A_{max} - \tilde{Q}_{j,\text{Low}} \leq 2(D_1 + K_1[\log(V)]^2 + \delta_{max}) = \Theta([\log(V)]^2)$, we can use Theorem 11 and conclude from (4.121) that the average delay for the packets in $\mathbb{P}_{j0}$ is $O([\log(V)]^2)/\lambda_{\mathbb{P}_{j0}}$ if $\lambda_{\mathbb{P}_{j0}} > 0$.

To prove (4.121), we first note that the packets that are served *when the queue is serving the end of the queue* consist of the following packet sets:

1. $\mathbb{P}_{j0}$
2. $\mathbb{P}_{j1}$, the set of packets that arrive when $q_j(t) \in \tilde{\mathcal{B}}_j$, and move to the right of $\tilde{Q}_{j,\text{Low}}$ but are still served from the end of the queue
3. $\mathbb{P}_{j2}$, the set of packets that are served from the end of the queue but arrive to the queue when $q_j(t) > Q_{j,\text{High}}$
4. $\mathbb{P}_{j3}$, the set of packets that are served from the end of the queue but arrive to the queue when $q_j(t) < \tilde{Q}_{j,\text{Low}}$

Now define $\lambda_{\mathbb{P}_{j1}}$, $\lambda_{\mathbb{P}_{j2}}$ and $\lambda_{\mathbb{P}_{j3}}$ to be the average rate of the packets in $\mathbb{P}_{j1}$, $\mathbb{P}_{j2}$ and $\mathbb{P}_{j3}$, respectively. We immediately have:

$$\mu_j^E = p\lambda_j = \lambda_{\mathbb{P}_{j0}} + \lambda_{\mathbb{P}_{j1}} + \lambda_{\mathbb{P}_{j2}} + \lambda_{\mathbb{P}_{j3}}.$$

This immediately implies that:

$$\lambda_{\mathbb{P}_{j0}} = p\lambda_j - (\lambda_{\mathbb{P}_{j1}} + \lambda_{\mathbb{P}_{j2}} + \lambda_{\mathbb{P}_{j3}}). \tag{4.122}$$

Therefore, to prove (4.121), it suffices to show that $\lambda_{\mathbb{P}_{ji}} = O(\frac{\delta_{max}c_1^*}{V^{\log(V)}})$ for $j = 1, 2, 3$. To do this, we first note that $\lambda_{\mathbb{P}_{j2}}$ and $\lambda_{\mathbb{P}_{j3}}$ are upper bounded by the total arrival rates of the packets that enter the queue when $q_j(t) > Q_{j,\text{High}}$ and $q_j(t) < \tilde{Q}_{j,\text{Low}}$, respectively. Using the definition of $Q_{j,\text{High}}$ and $\tilde{Q}_{j,\text{Low}}$ and Theorem 6, we have $\lambda_{\mathbb{P}_{j2}} = O(\frac{\delta_{max}c_1^*}{V^{\log(V)}}), \lambda_{\mathbb{P}_{j3}} = O(\frac{\delta_{max}c_1^*}{V^{\log(V)}})$.

We now compute an upper bound on $\lambda_{\mathbb{P}_{j1}}$. Note that when the queue decides to serve packets from the end of the queue, in order for it to serve a packet in $\mathbb{P}_{j1}$, i.e., a packet that arrives to the queue when $q_j(t) \in \tilde{\mathcal{B}}_j$ but moves to the right of $\tilde{Q}_{j,\text{Low}}$, we must have $q_j(t) < Q_{j,\text{Low}}$. Therefore, if we denote $\mu_{j1}(t)$ the number of packets in $\mathbb{P}_{j1}$ that are served at time t, we easily see that $\mu_{j1}(t) \leq \mu_{max} 1_{[q_j(t)<Q_{j,\text{Low}}]} 1_{[e_j(t)]}$ for all t. Therefore,

$$\begin{aligned}
\lambda_{\mathbb{P}_{j1}} &= \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\big\{\mu_{j1}(\tau)\big\} \\
&\leq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\big\{\mu_{max} 1_{[q_j(\tau)<Q_{j,\text{Low}}]} 1_{[e_j(\tau)]}\big\} \\
&\overset{(*)}{=} p\mu_{max} \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \Pr(q_j(\tau) < Q_{j,\text{Low}}) \\
&\overset{(**)}{\leq} p\mu_{max} \mathcal{P}(D_1, K_1[\log(V)]^2) \\
&= \frac{\delta_{max}c_1^*}{V^{\log(V)}}.
\end{aligned}$$

Here $(*)$ uses the fact that $e_j(t)$ is independent of the queue process, and $(**)$ follows from the definition of $\mathcal{P}(D_1, K_1[\log(V)]^2)$. Now using the fact that $\lambda_{\mathbb{P}_{ji}} = O(\frac{\delta_{max}c_1^*}{V^{\log(V)}})$ for $i = 1, 2, 3$ in (4.122), we conclude that:

$$\lambda_{\mathbb{P}_{j0}} \geq \big[p\lambda_j - O(\frac{\delta_{max}c_1^*}{V^{\log(V)}})\big]^+. \tag{4.123}$$

This proves Part (b).

Now by the definition of $\mathbb{P}_{j0}$, we see that the packets in $\mathbb{P}_{j0}$ only occupy an interval of size no more than $Q_{j,\text{High}} - \tilde{Q}_{j,\text{Low}} + A_{max} \leq 2(D_1 + K_1[\log(V)]^2 + \delta_{max}) = \Theta([\log(V)]^2)$. Thus, using Theorem 11, the average delay for the packets in $\mathbb{P}_{j0}$ is $O([\log(V)]^2)/\lambda_{\mathbb{P}_{j0}}$ if $\lambda_{\mathbb{P}_{j0}} > 0$. This proves Part (c) and completes the proof of the theorem. $\square$

Chapter 5

On order-optimal scheduling: the redundant constraint approach

In this chapter, we consider the problem of delay-efficient routing in stochastic networks. This corresponds to having a common constant cost function for all actions in the general model described in Chapter 3. In this case, we try to explore the possible delay reduction opportunities with respect to the *network size* under the Lyapunov technique. The algorithms developed in this chapter can be combined with those in Chapter 4 to further reduce the network delay. For ease of understanding, we state the specialized problem settings below rather than working on the abstract model presented in Chapter 3. Our solution is motivated by the connection between the Lyapunov technique and the RISM algorithm established in Section 4.8.

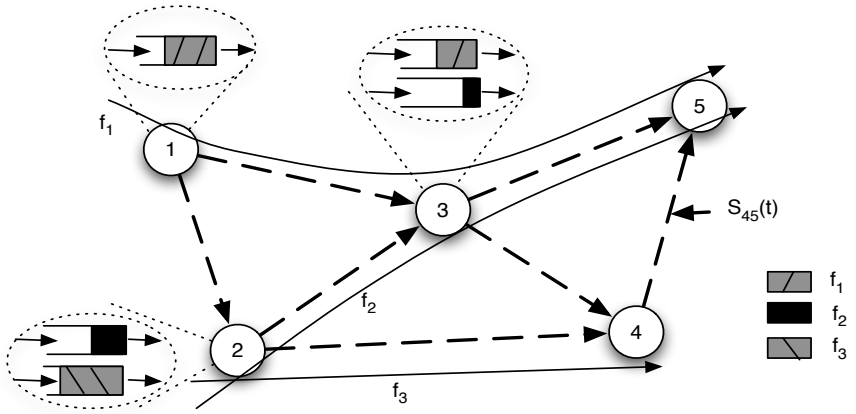

Figure 5.1: An example of flows going through a multihop network, where the arrows indicate the routes of the flows.

5.1 The redundant constraint approach: the intuition

In this section, we consider improving the delay performance of the Max-Weight type (QLA type) scheduling algorithm using appropriate "redundant constraints." To provide a good motivation of this approach, we first look at a simple example.

Consider the example in Fig. 5.1, where we try to support three commodity flows f_1, f_2, f_3 with arrival rates $\lambda_1, \lambda_2, \lambda_3$. We assume that the routes are all fixed. Suppose all channels are static, i.e., no channel variation. At each time, we need to allocate power and rates to the links for data transmission. Assume that due to physical constraints, e.g., interference, the obtained transmission rates are restricted to be in some rate allocation set $\mathcal{X}$. Now let $\mu_{[m,n]}^{f_i}$ be the rate allocated to flow f_i over link $[m,n]$. Using the results in Chapter 4, we see that the Max-Weight algorithm Dynamic Routing and Power Control algorithm (DRPC) in [NMR05] applied to this problem is closely related to solving the following optimization problem by a dual subgradient method:

$$(\mathbf{P1}) \ \min: \quad 1, \tag{5.1}$$

$$\text{s.t.} \quad \lambda_1 \leq \mu^{f_1}_{[1,3]},\, \mu^{f_1}_{[1,3]} \leq \mu^{f_1}_{[3,5]},$$

$$\lambda_2 \leq \mu^{f_2}_{[2,3]},\, \mu^{f_2}_{[2,3]} \leq \mu^{f_2}_{[3,5]},$$

$$\lambda_3 \leq \mu^{f_3}_{[2,4]},$$

$$(\mu^{f_i}_{[m,n]},\, i = 1, 2, 3,\, \forall [m, n] \in \mathcal{L})^T \in \mathcal{X}.$$

Here $\mathcal{L}$ denote the set of communication links. Now consider modifying (**P1**) by adding two *redundant* constraints as follows:

$$(\mathbf{P2}) \ \min: \quad 1, \tag{5.2}$$

$$\text{s.t.} \quad \lambda_1 \leq \mu^{f_1}_{[1,3]},\, \mu^{f_1}_{[1,3]} \leq \mu^{f_1}_{[3,5]},\, \lambda_2 \leq \mu^{f_2}_{[2,3]},\, \mu^{f_2}_{[2,3]} \leq \mu^{f_2}_{[3,5]}, \lambda_3 \leq \mu^{f_3}_{[2,4]}, \tag{5.3}$$

$$\theta\lambda_1 + (1-\theta)\mu^{f_1}_{[1,3]} \leq \mu^{f_1}_{[3,5]},\, \theta\lambda_2 + (1-\theta)\mu^{f_2}_{[2,3]} \leq \mu^{f_2}_{[3,5]}, \tag{5.4}$$

$$(\mu^{f_i}_{[m,n]},\, i = 1, 2, 3,\, \forall [m, n] \in \mathcal{L})^T \in \mathcal{X}.$$

The optimal solutions of (**P1**) and (**P2**) are always the same. Also, it has been observed in the optimization context, e.g., [Ber03], that adding the redundant constraints in (5.4) usually leads to faster convergence of the variables $\mu^{f_i}_{(m,n)}$ to their target values under gradient type methods, due to the fact that these additional constraints effectively "reduce" the search space of the optimal solutions for the optimization methods. Thus, in the network scheduling problem, we add to the original network queues a set of *accelerating queues*, which mimic the functionality of the constraints in (5.4). As we will see, by doing so one can indeed improve the convergence time of the allocated rate to each flow, which will likely lead to a better delay guarantee for the flows.

Interestingly, the recent work in [Nee09b], which proves that the Max-Weight scheduling algorithm achieves order-optimal (i.e., delay only scales as a function of the flow path

lengths, but not the number of flows) delay for downlink systems, can be viewed as considering all possible redundant constraints in the corresponding deterministic problem in the delay analysis. Thus this redundant constraint approach may be a potential tool for designing delay-order-optimal (delay increases only linearly in the flow path length) scheduling algorithms for general multihop network problems.

5.2 Network model

We consider a network operator that operates a general multihop network as shown in Fig. 5.1. The network is modeled as a graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N} = \{1, 2, ..., N\}$ denotes the set of N nodes and $\mathcal{L}$ denotes the set of L directed links in the network. Here a link $[m, n]$ where $m, n \in \mathcal{N}$ is in $\mathcal{L}$ if there is a communication link from node m to node n. The network is assumed to operate in slotted time, i.e., $t \in \{0, 1, 2, ...\}$. The goal of the operator is to support a set of flows going through the network and to achieve good delay performance.

5.2.1 The flow and routing model

We assume that there are a total of M flows going through the network. We denote the set of flows as $\mathcal{F} = \{1, 2, ..., M\}$. Each flow $f \in \mathcal{F}$ enters the network from its source node s_f and needs to be routed to its destination node d_f. Let $A_f(t)$ denote the number of packets of flow f arriving at its source node at time t. We assume that $A_f(t)$ is i.i.d. every slot and let $\lambda_f = \mathbb{E}\{A_f(t)\}$. The operator can observe the value of $A_f(t)$ at every slot. However, the statistics of $A_f(t)$ may be unknown. We assume that there exists some constant $A_{max} < \infty$ such that $A_f(t) \leq A_{max}$ for all f and t.

For each pair of nodes s, d, we define an acyclic path P between them to be a sequence of nodes $P = (n_1, n_2, ..., n_{K+1})$ such that $[n_i, n_{i+1}] \in \mathcal{L}$ for all $i = 1, ..., K$, $n_1 = s$, $n_{K+1} = d$ and $n_i \neq n_j$ if $i \neq j$. In order to highlight the scheduling component and to convey the idea of our algorithm, we first assume that every flow f is routed via a single given fixed path P_f to its destination. The case when every flow is routed over multiple paths to its destination is considered in Section 5.8. We define the *length* of the path $P_f = \{s_f = n_1, ..., n_{K_f+1} = d_f\}$ for flow f to be K_f. This denotes the number of links that every packet from flow f has to be transmitted over. We use $k_f(n)$ to denote node n's order in the path P_f for all $n \in P_f$, e.g., $k_f(s_f) = 1$ and $k_f(d_f) = K_f + 1$. For any node n with $k_f(n) \geq 2$, we use $u^f(n)$ to denote its *upstream* node in the path P_f; for any node n with $k_f(n) \leq K_f$, we use $l^f(n)$ to denote its *downstream* node in the path P_f.

Note that this routing model is different from the multihop network considered in [NMR05], [GJ07], where no route information is needed. However, this assumption is not very restrictive. In many cases, e.g., the internet, such route information can easily be obtained.

5.2.2 The transmission model

We assume that the channel states of the links are potentially time varying. We use $\boldsymbol{S}(t) = (S_{mn}(t), m, n \in \mathcal{N})$, where $S_{mn}(t)$ is the channel condition between nodes m and n, to denote the aggregate channel state vector of the network. Note that $\boldsymbol{S}(t)$ contains information of channels between all pairs of nodes in the network, even pairs that do not have a communication link in between. This is to capture the fact that in some cases, though one node can not communicate with another node, their transmissions can still

interfere with each other. We assume that $\boldsymbol{S}(t)$ takes values in a finite state space $\boldsymbol{\mathcal{S}}$. For simplicity, we also assume that $\boldsymbol{S}(t)$ is i.i.d. over slots, but the components of $\boldsymbol{S}(t)$ are allowed to be correlated. The network operator can observe the aggregate channel state information $\boldsymbol{S}(t)$ at every time slot, but the distribution of $\boldsymbol{S}(t)$ is not necessarily known.

At every time t, after observing the channel state vector $\boldsymbol{S}(t)$, the network operator allocates power to each link for data transmission. It does so by choosing a power allocation vector $\boldsymbol{P}(t) = (P_{[m,n]}(t), [m,n] \in \mathcal{L})$, where $P_{[m,n]}(t)$ denotes the power allocated to link $[m,n]$. We assume that if $\boldsymbol{S}(t) = \boldsymbol{S} \in \boldsymbol{\mathcal{S}}$, then $\boldsymbol{P}(t)$ is chosen from a feasible power allocation set associated with $\boldsymbol{S}$, denoted by $\mathcal{P}^{(\boldsymbol{S})}$. We assume that for any $\boldsymbol{S} \in \boldsymbol{\mathcal{S}}$, $\mathcal{P}^{(\boldsymbol{S})}$ is compact and time invariant. Given the channel state $\boldsymbol{S}(t)$ and the power vector $\boldsymbol{P}(t)$, the rate over link $[m,n]$ at time t is given by $\mu_{[m,n]}(t) = \Phi_{[m,n]}(\boldsymbol{S}(t), \boldsymbol{P}(t))$, for some general *rate-power* function $\Phi_{[m,n]}(\cdot,\cdot)$. Now let $\mu^f_{[m,n]}(t)$ be the rate allocated to flow f over link $[m,n]$ at time t, chosen subject to the following constraint: $\sum_f \mu^f_{[m,n]}(t) \leq \mu_{[m,n]}(t)$. It is evident that if $m \neq u^f(n)$, i.e., m is not the upstream node of n in the path P_f, then $\mu^f_{[m,n]}(t) = 0 \; \forall \, t$. In the following, we assume that there exists some $\mu_{max} < \infty$ such that $\mu_{[m,n]}(t) \leq \mu_{max}$ for all $[m,n] \in \mathcal{L}$, $\boldsymbol{S} \in \boldsymbol{\mathcal{S}}$ and $\boldsymbol{P} \in \mathcal{P}^{(\boldsymbol{S})}$.

5.2.3 Queueing dynamics and network capacity region

Let $\boldsymbol{Q}(t) = (Q^f_n(t), f \in \mathcal{F}, n \in P_f)$ and $t = 0, 1, 2, ...$ be the queue backlog vector of the network, in units of packets. [1] Note that the queues are associated with the nodes. We assume that the following queueing dynamics for all nodes $n \in P_f$ with $k_f(n) \leq K_f$:

$$Q^f_n(t+1) \leq \max[Q^f_n(t) - \mu^f_{[n,l^f(n)]}(t), 0] + \mu^f_{[u^f(n),n]}(t). \tag{5.5}$$

[1] Nodes that are not used by any flow are assumed to always have zero backlog for all flows.

In the above equation, we assume that when $k_f(n) = 1$, i.e., when node n is the source node of flow f, $\mu^f_{[u^f(n),n]}(t) = A_f(t)$ for all t. The inequality is due to the fact that the upstream node may not have enough packets to send. When $k_f(n) = K_f + 1$, i.e., $n = d_f$, we always assume that $Q^f_n(t) = \mu^f_{[n,l^f(n)]}(t) = 0$ for all t. Throughout this chapter, we assume the following notion of queue stability:

$$\overline{Q} \triangleq \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{f,n} \mathbb{E}\{Q^f_n(\tau)\} < \infty. \tag{5.6}$$

Define $\Lambda \subset \mathbb{R}^M$ to be the *network capacity region*, which is the closure of all arrival rate vectors $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_M)^T$ for which under the routing and transmission configurations of the network, e.g., fixed path routing, there exists a stabilizing control algorithm that ensures (5.6). The following theorem from [GNT06] gives a useful characterization of the capacity region in our setting and will be useful in the following analysis.

Theorem 16. *The capacity region* $\boldsymbol{\Lambda}$ *is the set of arrival vectors* $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_M)^T \in \mathbb{R}^M_+$ *(*$\mathbb{R}_+$ *is the set of nonnegative real numbers) such that there exists a stationary randomized power allocation and scheduling algorithm that allocates power and flow rates purely as a function of* $\boldsymbol{S}(t)$ *and achieves*

$$\mathbb{E}\{\mu^f_{[n,l^f(n)]}(t)\} = \lambda_f, \ \forall \ f \in \mathcal{F}, n \in P_f : k_f(n) \leq K_f, \tag{5.7}$$

where the expectation is taken with respect to the random channel states and the (possibly) random power allocation and scheduling actions. □

In the following, we use $\boldsymbol{S}$-only policies to refer to stationary randomized policies that allocate power and flow rates purely as functions of the aggregate channel state $\boldsymbol{S}(t)$.

5.2.4 The delay efficient scheduling problem

Our goal is to find a joint power allocation and scheduling algorithm that, at every time slot, chooses the right power allocation vector and transmits the right amount of packets for each flow, so as to maintain queue stability in the network and yield good delay performance. We refer to this problem as the *Delay Efficient Scheduling Problem* (DESP).

5.3 Related work

This framework has been studied in many previous articles, e.g., [TE92], [NMR05]. It is also well known that Max-Weight type algorithms in, e.g., [NMR05], [GNT06], can be used to stabilize the network whenever the arrival rate vector is in the network capacity region. However, the best known delay bounds of Max-Weight algorithms are not *order optimal*. Indeed, all the known delay-efficient results for Max-Weight algorithms are for single-hop networks, e.g., [TE93], [Nee09b]. For multihop networks, the Max-Weight algorithm usually incurs a time average network delay which grows at least quadratically in the network size [BSS09], and many algorithms, both Max-Weight type or non-Max-Weight type, have thus been proposed trying to improve its delay performance, e.g., [YST08], [YSR09], [BSS09], [GJ07], [NZJ09], [SdV09], [SSS04] and [VL07]. However, though the proposed schemes are intuitively delay efficient, it is usually difficult to obtain explicit delay bounds for the non-Max-Weight type algorithms, and the analytical bounds for the Max-Weight type schemes are usually not better than those of DRPC in [NMR05].

5.4 Towards better delay performance

In this section, we develop the Delay-Efficient SCheduling algorithm (DESC).

5.4.1 Accelerating queues and redundant constraints in optimization

We first define the notion of an *accelerating queue* (AQ). For each flow f traversing a path $P_f = (n_1^f, n_2^f, ..., n_{K_f+1}^f)$, we create K_f *accelerating queues* (AQ) $H_n^f(t)$, $n \in \{n_1^f, n_2^f, ..., n_{K_f}^f\}$ that evolve as follows:

$$H_n^f(t+1) = \max\left[H_n^f(t) - \mu^f_{[n,l^f(n)]}(t), 0\right] + \theta A_f(t) + (1-\theta)\mu^f_{[u^f(n),n]}(t), \tag{5.8}$$

where $\theta \in (0, 1]$ is a parameter that is chosen independent of the network size and routing configurations. We similarly define $\mu^f_{[u^f(n),n]}(t) = A_f(t)$ if $k^f(n) = 1$, i.e., $n = s_f$. If $k^f(n) = K_f + 1$, i.e., $n = d_f$, we also define $H_n^f(t) = \mu^f_{[n,l^f(n)]}(t) = 0, \forall\, t$. These AQs are used to propagate the instantaneous traffic arrival information to all downstream nodes. We emphasize that these AQs are *virtual* queues (or counters) and can be easily implemented in software. The actual queues in the system still obey the queueing dynamics in (5.5). We relate the AQs to redundant constraints in (5.4). Consider solving the problem (**P2**) in (5.2) with a dual subgradient method and assign to the redundant constraints in (5.4) Lagrange multipliers $H_3^{f_1}$ and $H_3^{f_2}$. Then $H_3^{f_1}$ and $H_3^{f_2}$ are updated according to:

$$H_3^{f_1}(t+1) = \left[H_3^{f_1}(t) - \mu^{f_1}_{[3,5]}(t)\right]^+ + \theta\lambda_1 + (1-\theta)\mu^{f_1}_{[1,3]}(t)$$

$$H_3^{f_2}(t+1) = \left[H_3^{f_2}(t) - \mu^{f_2}_{[3,5]}(t)\right]^+ + \theta\lambda_2 + (1-\theta)\mu^{f_2}_{[2,3]}(t).$$

Comparing these with the update rules (5.8), we see that *the AQs correspond exactly to the Lagrange multipliers of the redundant constraints.*

5.4.2 The DESC algorithm

In this section, we develop the Delay Efficient SCheduling algorithm (DESC) that will be applied to the DESP problem.

Delay Efficient SCheduling Algorithm (DESC): Choose a parameter $\theta \in (0, 1]$ independent of the network size and routing configurations. At every time slot t, observe all queue values $Q_n^f(t)$ and $H_n^f(t)$, and do:

(1) Link Weight Computing: For all $[m, n] \in \mathcal{L}$ such that there exists a flow f with $m = u^f(n)$, find the flow $f^*_{[m,n]}$ such that (ties broken arbitrarily)

$$f^*_{[m,n]} = \underset{f \in \mathcal{F}: m = u^f(n)}{\arg\max} \left\{ Q_m^f(t) - Q_n^f(t) + H_m^f(t) - (1-\theta) H_n^f(t) \right\}. \tag{5.9}$$

Then define the *weight* of the link $[m, n]$ to be:

$$W^*_{[m,n]}(t) = \max \Big[Q_m^{f^*_{[m,n]}}(t) - Q_n^{f^*_{[m,n]}}(t) + H_m^{f^*_{[m,n]}}(t) - (1-\theta) H_n^{f^*_{[m,n]}}(t), 0 \Big]. \tag{5.10}$$

If no such f exists, i.e., $[m, n]$ is not used by any flow, define $W^*_{[m,n]}(t) = 0 \; \forall \, t$.

(2) Power Allocation: Observe the aggregate channel state $\boldsymbol{S}(t)$, if $\boldsymbol{S}(t) = \boldsymbol{S}$, choose $\boldsymbol{P}(t) \in \mathcal{P}^{(\boldsymbol{S})}$ such that:

$$\boldsymbol{P}(t) = \underset{\boldsymbol{P} \in \mathcal{P}^{(\boldsymbol{S})}}{\arg\max} \sum_{[m,n] \in \mathcal{L}} \mu_{[m,n]}(t) W^*_{[m,n]}(t), \tag{5.11}$$

where $\mu_{[m,n]}(t) = \Phi_{[m,n]}(\boldsymbol{S}, \boldsymbol{P}(t))$.

(3) Scheduling: For each link $[m, n] \in \mathcal{L}$, allocate the transmission rate as follows:

$$\mu^f_{[m,n]}(t) = \begin{cases} \mu_{[m,n]}(t) & \text{if } f = f^*_{[m,n]} \text{ and } W^*_{[m,n]}(t) > 0 \\ 0 & \text{else.} \end{cases}$$

That is, the full transmission rate over each link $[m,n]$ at time t is allocated to the flow $f^*_{[m,n]}$ that achieves the maximum positive weight $W^*_{[m,n]}(t)$ of the link. If $\mu^{f^*_{[m,n]}}_{[m,n]}(t) > Q^{f^*_{[m,n]}}_m(t)$, *null bits* are transmitted if needed.

(4) Queue Update: Update $Q^f_n(t)$ and $H^f_n(t)$, $\forall\, f, n \in P_f$, according to (5.5) and (5.8), respectively.

We note that DESC inherits almost all properties of previous Max-Weight type algorithms: it does not require statistical knowledge of the arrival or channels, and it guarantees network stability whenever the arrival rate is inside the network capacity region. DESC is also not very difficult to implement. The link weights can easily be computed locally. The scheduling part can be done node-by-node. The AQs are virtual queues implemented in software and can easily be updated by message passing the information of $A_f(t)$ to all the nodes in P_f, similar to the assumptions of the internet flow models in [LPC08], [LL99]. Although DESC requires routing information, such information is usually not difficult to obtain in many contexts, e.g., the internet. The most complex part is the power allocation computation, which in general can be NP-hard. However, it can easily be solved in some special cases, e.g., when transmissions over different links do not interfere with each other, e.g., internet, or it can be easily approximated at the cost of some network capacity loss [GNT06].

We finally note that DESC is similar to the DRPC algorithm developed in [NMR05], in terms of power allocation and scheduling. Indeed, the major difference is the use of AQs in DESC. However, we will see later that this simple distinction can lead to a significant difference in algorithm performance.

5.5 DESC: stability and delay performance

Now we present the performance results of DESC and discuss their implications, using the DRPC algorithm in [NMR05] as the benchmark algorithm for comparison. The proofs are presented in Section 5.6. Our first result characterizes the performance of DESC in terms of queue stability. In particular, we show that whenever the arrival rate vector is within the capacity region, both the actual queues and the AQs are stable. This result shows that the DESC algorithm is *throughput optimal*. Our second result concerns the difference between the aggregate packet arrival rates and the aggregate service rates allocated to the flows. This result, as we will see, states that under DESC, the service rates allocated to each flow over its path converge quickly to their desired rate values. We now have our first performance result of DESC. In the rest of this chapter, the notation $\overline{x(t)}$ is defined:

$$\overline{x(t)} = \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\{x(t)\}, \tag{5.12}$$

which denotes the expected time average of a sequence $\{x(t), t = 0, 1, 2, ...\}$.

Theorem 17. *Suppose that there exists an $\epsilon > 0$ such that $\boldsymbol{\lambda} + \mathbf{1}\epsilon \in \boldsymbol{\Lambda}$. Then, under the DESC algorithm, we have:*

$$\overline{\sum_f \sum_{n\in P_f} \frac{Q_n^f(t)}{K_f}} \leq \frac{2\sum_f K_f B^2}{\epsilon} - \overline{\sum_f \sum_{n\in P_f} \frac{H_n^f(t)[\theta k_f(n) + (1-\theta)]}{K_f}}, \tag{5.13}$$

$$\overline{\sum_f \sum_{n\in P_f} H_n^f(t)} \leq \frac{2\sum_f K_f B^2}{\theta\epsilon} - \overline{\sum_f \frac{Q_{s_f}^f(t)}{\theta}}, \tag{5.14}$$

where $\mathbf{1}$ is the M-dimensional column vector with every entry equal to 1, $B = \max[A_{max}, \mu_{max}]$. Here the expectation is taken over the randomness of the arrivals and link states.

Note that here $\boldsymbol{\lambda} + \mathbf{1}\epsilon \in \boldsymbol{\Lambda}$ means that $(\lambda_1 + \epsilon, \lambda_2 + \epsilon, ..., \lambda_M + \epsilon)^T \in \boldsymbol{\Lambda}$. In other words, we can increase the arrival rates of *all* flows by the same amount ϵ and the rate vector

is still supportable. Thus ϵ can be viewed as measuring how far the current rate vector is from the boundary of the capacity region $\mathbf{\Lambda}$. Also note that the bound (5.13) contains the parameters K_f on the left-hand side, which are the path lengths of the flows. This is due to the following: When analyzing the average total backlog size, one has to compare the drift under DESC with the drift under an $\boldsymbol{S}$-only strategy. To obtain bounds on the total network backlog, the $\boldsymbol{S}$-only strategy has to *simultaneously* ensure that for each flow f, its average output rate at every node $n \in P_f (n \neq d_f)$ is larger than its average input rate into that node by some $\delta_n^f > 0$. To guarantee such a stationary randomized policy exists, we need $\sum_{n \in P_f} \delta_n^f \leq \epsilon$. In our case, all δ_n^f are chosen to be equal, hence $\delta_n^f = \epsilon / K_f$ and K_f appears on the left-hand side.

We note that the bound (5.13) has the potential to be strictly better than the bounds of the DRPC and EDRPC algorithms in [NMR05]. Indeed, the corresponding congestion bound under DRPC can be shown to be

$$\overline{\sum_f \sum_{n \in P_f} \frac{Q_n^f(t)}{K_f}} \quad \leq \quad \frac{\sum_f K_f B^2}{\epsilon}. \tag{5.15}$$

Hence if the second term in (5.13) is large, then (5.13) can be strictly better than (5.15). As we will see in the simulation section, the second term is indeed large and thus the bound in (5.13) can actually be better than (5.15) for DRPC.

We also note that the bound for the AQs in (5.14) is smaller than the bound (5.15) for the actual queues under DRPC roughly by a factor $\theta K_f / 2$. Since the AQ sizes measure the convergence speed of the allocated rates under DESC and the *actual queue* sizes measure the convergence speed of the allocated rates under DRPC, we see from (5.14) and (5.15) that the allocated rates under DESC converge faster to their target values

than under DRPC. To see this better, we define the following total *approximate network service lag* function $\text{Lag}(t)$ at time t as:

$$\text{Lag}(t) = \sum_f \left[K_f A_f[0, t-1] - \sum_{n \in P_f} \mu^f_{[n,l^f(n)]}[0, t-1] \right], \tag{5.16}$$

where $A_f[0, t-1] = \sum_{\tau=0}^{t-1} A_f(\tau)$ and $\mu^f_{[n,l^f(n)]}[0, t-1] = \sum_{\tau=0}^{t-1} \mu^f_{[n,l^f(n)]}(\tau)$ are the total number of arrivals of flow f and the total amount of rate allocated to flow f over link $[n, l^f(n)]$ in the time period $[0, t-1]$. Since every arrival from $A_f(t)$ needs to be transmitted K_f times before reaching the destination, the quantity $\text{Lag}(t)$ can be viewed as approximating the total amount of "effective unserved data" that is currently in the network. Hence any delay efficient algorithm is expected to have a small time average value of $\text{Lag}(t)$. The following theorem characterizes the performance of DESC with respect to the $\text{Lag}(t)$ metric.

Theorem 18. *Suppose that there exists an $\epsilon > 0$ such that $\boldsymbol{\lambda} + \mathbf{1}\epsilon \in \boldsymbol{\Lambda}$. Then, under the DESC algorithm, we have:*

$$\overline{Lag(t)} \quad \leq \quad \frac{2\sum_f K_f B^2}{\theta^2 \epsilon} - \sum_f \frac{\overline{Q^f_{s_f}(t)}}{\theta^2}, \tag{5.17}$$

where the notion $\overline{x(t)}$ is defined in (5.12).

Note that if $\sum_f K_f = O(N)$, e.g., $M = O(1)$, and $\epsilon = \Theta(1)$, then (5.17) implies that the average total service lag in the network grows only *linearly* in the network size.

5.5.1 Example

As a concrete demonstration of Theorems 17 and 18, we look at a simple example shown in Fig. 5.2, where a flow is going through an $N+1$ node line wireless network, with the source being node 1 and the destination being node $N+1$.

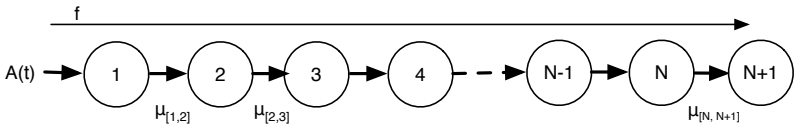

Figure 5.2: A flow traversing a tandem.

In this case we see that the path length $K_f = N$. Suppose that we choose $\theta = 1$ in the DESC algorithm. Then, Theorems 17 and 18 state that under DESC, we have:

$$\overline{\sum_{i=1}^{N} Q_n(t)} \leq \frac{2N^2B^2}{\epsilon} - \overline{\sum_{n=1}^{N} nH_n(t)}, \tag{5.18}$$

$$\overline{\sum_{n=1}^{N} H_n(t)} \leq \frac{2NB^2}{\epsilon} - \overline{Q_1(t)}, \tag{5.19}$$

$$\overline{\mathrm{Lag}(t)} = \overline{N\sum_{\tau=0}^{t-1} A(\tau) - \sum_{n=1}^{N}\sum_{\tau=0}^{t-1} \mu_{[n,n+1]}(\tau)} \leq \frac{2NB^2}{\epsilon}. \tag{5.20}$$

Suppose the network parameters are such that $\lambda = \Theta(1)$ and $\epsilon = \Theta(1)$. Then, we see that the term $\overline{\sum_{n=1}^{N} nH_n(t)} = \Omega(N^2)$. By subtracting out this term, the bound in (5.18) can likely be small. (5.19) shows that the time average value of the AQs is $O(N)$, which implies that the rates allocated to the links converge in $O(N)$ time to their target values. (5.20) shows that the average total network service lag at all the nodes is no more than some $\Theta(N)$ constant, whereas the average service lag under DRPC is $\Omega(N^2)$ in this case.

We will see in the simulation section that under the DESC algorithm, the time average actual queue backlog can indeed be $\Theta(N)$ when $\lambda = \Theta(1)$ and $\epsilon = \Theta(1)$.

5.5.2 Discussion of the choice of θ

We note that the results in Theorems 17 and 18 hold for any $\theta \in (0, 1]$. Hence the bounds may be optimized by choosing the best θ. Intuitively, using a larger θ, e.g., $\theta = 1$ will lead to a faster convergence of the allocated rates. However, using a $\theta \neq 1$ may also be

beneficial in some cases when we want to reduce the impact of the arrival information, for example, when the propagated traffic information may become noisy.

5.6 Performance analysis

In this section we analyze the DESC algorithm. To start, we first have the following lemma, which shows that if the current rate vector is strictly inside Λ, then we can find an $\boldsymbol{S}$-only policy that offers "perturbed" rates to all the nodes along the paths for all flows.

Lemma 6. *Suppose that there exists an $\epsilon > 0$ such that $\boldsymbol{\lambda} + \mathbf{1}\epsilon \in \boldsymbol{\Lambda}$. Then, there exists an $\boldsymbol{S}$-only policy under which:*

$$\mathbb{E}\{\mu^f_{[n,l^f(n)]}(t)\} = \lambda_f + \delta^f_n, \quad \forall\, n \in P_f : k_f(n) \leq K_f, \tag{5.21}$$

for any $-\lambda_f \leq \delta^f_n \leq \epsilon$ and for all $f \in \mathcal{F}$.

Proof. By Theorem 16, we see that if $\boldsymbol{\lambda} + \mathbf{1}\epsilon \in \Lambda$, then there exists an $\boldsymbol{S}$-only policy Π that achieves:

$$\mathbb{E}\{\mu^f_{[n,l^f(n)]}(t)\} = \lambda_f + \epsilon, \quad \forall\, f, n \in P_f : k_f(n) \leq K_f.$$

Now we create a new $\boldsymbol{S}$-only policy Π' by modifying Π as follows. In every time slot, allocate rates to nodes using the policy Π. However, for each $n \in P_f$, in every time slot, transmit packets for flow f with the corresponding rate with probability $(\lambda_f+\delta^f_n)/(\lambda_f+\epsilon)$ (this is a valid probability since $-\lambda_f \leq \delta^f_n \leq \epsilon$). We see that Π' is an $\boldsymbol{S}$-only policy, and that (5.21) is satisfied under Π'. □

We now analyze the performance of DESC. To start, we define the following Lyapunov function:

$$L(\boldsymbol{Q}(t), \boldsymbol{H}(t)) = \frac{1}{2}\sum_{f}\sum_{n\in P_f}\Big([Q_n^f(t)]^2 + [H_n^f(t)]^2\Big). \tag{5.22}$$

Denote $\boldsymbol{Z}(t) = (\boldsymbol{Q}(t), \boldsymbol{H}(t))$, and define the one-slot conditional *Lyapunov drift* to be

$$\Delta(t) \;\; = \;\; \mathbb{E}\big\{L(t+1) - L(t) \mid \boldsymbol{Z}(t)\big\}, \tag{5.23}$$

where we use $L(t)$ as a short-hand notation for $L(\boldsymbol{Q}(t), \boldsymbol{H}(t))$. We have the following lemma:

Lemma 7. *The drift $\Delta(t)$ defined in (5.23) satisfies:*

$$\begin{aligned}\Delta(t) \leq C &- \sum_{f}\sum_{n\in P_f} H_n^f(t)\mathbb{E}\big\{\mu^f_{[n,l^f(n)]}(t) - (1-\theta)\mu^f_{[u^f(n),n]}(t) - \theta A_f(t) \mid \boldsymbol{Z}(t)\big\} \\ &- \sum_{f}\sum_{n\in P_f} Q_n^f(t)\mathbb{E}\big\{\mu^f_{[n,l^f(n)]}(t) - \mu^f_{[u^f(n),n]}(t) \mid \boldsymbol{Z}(t)\big\},\end{aligned} \tag{5.24}$$

where $C = 2\sum_f K_f B^2$.

Proof. See Section 5.12.1. □

We are now ready to prove Theorem 17. We use the following theorem, which is Lemma 5.3 in Page 81 in [GNT06].

Theorem 19. *Let $\boldsymbol{Q}(t)$ be a vector process of queue backlogs that evolves according to some probability law, and let $L(\boldsymbol{Q}(t))$ be a non-negative function of $\boldsymbol{Q}(t)$. If there exists processes $f(t)$ and $g(t)$ and positive constants $a, b > 0$ such that at all times t, we have:*

$$\Delta(t) \leq ag(t) - bf(t),$$

then

$$b\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\big\{f(\tau)\big\} \leq a\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\big\{g(\tau)\big\}.\ \square$$

Proof. (Theorem 17) Rearranging the terms in (5.24), we obtain:

$$\begin{aligned}\Delta(t) \leq C &+ \sum_f \sum_{n\in P_f} \theta H_n^f(t)\mathbb{E}\big\{A_f(t) \mid \boldsymbol{Z}(t)\big\} \\ &+ \sum_f \big[Q_{s_f}^f(t) + (1-\theta)H_{s_f}^f(t)\big]\mathbb{E}\big\{A_f(t) \mid \boldsymbol{Z}(t)\big\} \\ &- \sum_f \sum_{n\in P_f: k_f(n)\leq K_f} \mathbb{E}\big\{\mu_{[n,l^f(n)]}^f(t)\big[Q_n^f(t) - Q_{l^f(n)}^f(t) \\ &\qquad + H_n^f(t) - (1-\theta)H_{l^f(n)}^f(t)\big] \mid \boldsymbol{Z}(t)\big\}.\end{aligned} \tag{5.25}$$

Comparing (5.25) and DESC, and recalling that $\mu_{[m,n]}(t) = \Phi_{[m,n]}(\boldsymbol{S}(t), \boldsymbol{P}(t))$, we see that at every time slot, *the DESC algorithm chooses the power allocation vector and allocates transmission rates to flows to minimize the right-hand side (RHS) of the drift expression (5.25).* Because the RHS of (5.24) and (5.25) are equivalent, the drift value satisfies:

$$\begin{aligned}\Delta(t) \leq C &- \sum_f \sum_{n\in P_f} H_n^f(t)\mathbb{E}\big\{\mu_{[n,l^f(n)]}^{*f}(t) - (1-\theta)\mu_{[u^f(n),n]}^{*f}(t) - \theta A_f(t) \mid \boldsymbol{Z}(t)\big\} \\ &- \sum_f \sum_{n\in P_f} Q_n^f(t)\mathbb{E}\big\{\mu_{[n,l^f(n)]}^{*f}(t) - \mu_{[u^f(n),n]}^{*f}(t) \mid \boldsymbol{Z}(t)\big\},\end{aligned} \tag{5.26}$$

where $\mu_{[n,l^f(n)]}^{*f}(t)$ corresponds to the rate allocated to Flow f on link $[n, l^f(n)]$ at time t by any other alternative algorithms.

Now since $\boldsymbol{\lambda} + \boldsymbol{\epsilon} \in \Lambda$, by Lemma 6, we see that there exists an $\boldsymbol{S}$-only policy that chooses the power allocation vector $\boldsymbol{P}(t)$ and allocates transmission rates $\mu_{[n,l^f(n)]}^f(t)$ purely as a function of the aggregate channel state $\boldsymbol{S}(t)$, and yields:

$$\mathbb{E}\big\{\mu_{[n,l^f(n)]}^{*f}(t) \mid \boldsymbol{Z}(t)\big\} = \lambda_f + \frac{k_f(n)\epsilon}{K_f}, \tag{5.27}$$

for all $f \in \mathcal{F}$ and $n \in P_f : k_f(n) \leq K_f$. Thus,

$$\mathbb{E}\big\{\mu_{[n,l^f(n)]}^{*f}(t)\big\} = \lambda_f + \frac{k_f(n)\epsilon}{K_f}, \tag{5.28}$$

$$\mathbb{E}\big\{\mu_{[u^f(n),n]}^{*f}(t)\big\} = \lambda_f + \frac{(k_f(n)-1)\epsilon}{K_f}. \tag{5.29}$$

Plugging this alternative algorithm into (5.26), we have:

$$\Delta(t) \;\leq\; C - \sum_f \sum_{n\in P_f} Q_n^f(t)\frac{\epsilon}{K_f} - \sum_f \sum_{n\in P_f} H_n^f(t)\frac{[\theta k_f(n)+(1-\theta)]\epsilon}{K_f}, \tag{5.30}$$

which by Theorem 19 implies

$$\overline{\sum_f \sum_{n\in P_f}\left[\frac{Q_n^f(t)}{K_f} + \frac{H_n^f(t)[\theta k_f(n)+(1-\theta)]}{K_f}\right]} \leq \frac{C}{\epsilon} = \frac{2\sum_f K_f B^2}{\epsilon}. \tag{5.31}$$

Rearranging terms, we have:

$$\overline{\sum_f \sum_{n\in P_f} \frac{Q_n^f(t)}{K_f}} \leq \frac{2\sum_f K_f B^2}{\epsilon} - \overline{\sum_f \sum_{n\in P_f} \frac{H_n^f(t)[\theta k_f(n)+(1-\theta)]}{K_f}}.$$

This proves (5.13). Now similar to the derivation of (5.30), but plug in (5.26) another alternative $\boldsymbol{S}$-only policy that yields for all $f \in \mathcal{F}$:

$$\mathbb{E}\big\{\mu^{*f}_{[n,l^f(n)]}(t) \mid \boldsymbol{Z}(t)\big\} = \lambda_f + \epsilon,\ n \in P_f : k_f(n) \leq K_f. \tag{5.32}$$

Such an algorithm exists by Lemma 6. We then obtain

$$\Delta(t) \leq C - \sum_f Q_{s_f}^f(t)\epsilon - \sum_f \sum_{n\in P_f} H_n^f(t)\theta\epsilon - \sum_f H_{s_f}^f(t)(1-\theta)\epsilon. \tag{5.33}$$

Using the fact that $H_{s_f}^f(t)(1-\theta)\epsilon \geq 0,\ \forall\, t$, (5.33) implies that

$$\overline{\sum_f \frac{Q_{s_f}^f(t)}{\theta}} + \overline{\sum_f \sum_{n\in P_f} H_n^f(t)} \leq \frac{2\sum_f K_f B^2}{\theta\epsilon}, \tag{5.34}$$

proving the theorem. □

Now we prove Theorem 18:

Proof. (Theorem 18) For a flow $f \in \mathcal{F}$, let its path be $P_f = \{n_1, n_2, ..., n_{K_f+1}\}$, where $n_1 = s_f$ and $n_{K_f+1} = d_f$. From (5.8), it can be shown as in [GNT06] that for all t, we have

$$H_{n_1}^f(t) \geq \sum_{\tau=0}^{t-1} A_f(\tau) - \sum_{\tau=0}^{t-1} \mu^f_{[n_1,n_2]}(\tau),$$

which implies

$$\sum_{\tau=0}^{t-1} \mu^f_{[n_1,n_2]}(\tau) \geq \sum_{\tau=0}^{t-1} A_f(\tau) - H_{n_1}^f(t). \tag{5.35}$$

Repeating the above for n_2, we have

$$\begin{aligned} H^f_{n_2}(t) &\geq \sum_{\tau=0}^{t-1}[\theta A_f(\tau) + (1-\theta)\mu^f_{[n_1,n_2]}(\tau)] - \sum_{\tau=0}^{t-1}\mu^f_{[n_2,n_3]}(\tau) \\ &\geq \sum_{\tau=0}^{t-1} A_f(\tau) - \sum_{\tau=0}^{t-1}\mu^f_{[n_2,n_3]}(\tau) - (1-\theta)H^f_{n_1}(t), \end{aligned}$$

where the second inequality follows from (5.35). Hence,

$$H^f_{n_2}(t) + (1-\theta)H^f_{n_1}(t) \geq \sum_{\tau=0}^{t-1} A_f(\tau) - \sum_{\tau=0}^{t-1}\mu^f_{[n_2,n_3]}(\tau).$$

More generally, we have for all $i = 1, ..., K_f$ that

$$\sum_{j=1}^{i}(1-\theta)^{i-j}H^f_{n_j}(t) \geq \sum_{\tau=0}^{t-1} A_f(\tau) - \sum_{\tau=0}^{t-1}\mu^f_{[n_i,n_{i+1}]}(\tau).$$

Summing up all $i = 1, 2, ..., K_f$, we have

$$\sum_{i=1}^{K_f}\sum_{j=1}^{i}(1-\theta)^{i-j}H^f_{n_j}(t) \geq \sum_{i=1}^{K_f}\left[\sum_{\tau=0}^{t-1} A_f(\tau) - \sum_{\tau=0}^{t-1}\mu^f_{[n_i,n_{i+1}]}(\tau)\right].$$

However, we see that

$$\sum_{i=1}^{K_f}\sum_{j=1}^{i}(1-\theta)^{i-j}H^f_{n_j}(t) = \sum_{i=1}^{K_f} H^f_{n_i}(t)\cdot\left[\sum_{j=0}^{K_f-i}(1-\theta)^j\right] \leq \frac{1}{\theta}\sum_{i=1}^{K_f} H^f_{n_i}(t),$$

which implies

$$\sum_{i=1}^{K_f}\left[\sum_{\tau=0}^{t-1} A_f(\tau) - \sum_{\tau=0}^{t-1}\mu^f_{[n_i,n_{i+1}]}(\tau)\right] \leq \frac{1}{\theta}\sum_{i=1}^{K_f} H^f_{n_i}(t).$$

Summing this over all $f \in \mathcal{F}$ and using (5.14) in Theorem 17 proves Theorem 18. □

5.7 DESC under delayed arrival information

Here we consider the case when the time required to propagate the arrival information $A_f(t)$ is nonzero. Such a case can happen, for instance, when there is no central controller, and thus, message passing is required to propagate the $A_f(t)$ values. Let $D^f_n(t)$ be the delay (in number of slots) to propagate the $A_f(t)$ information from s_f to node $n \in P_f$ at time t. We assume that there exists a constant $D < \infty$ such that $D^f_n(t) \leq D$

for all f, n, t. Note that in this case, we can no longer use (5.8) to update the AQs due to the message passing delay. Instead, we modify the DESC algorithm to use the "delayed" traffic information. Specifically, we create a set of AQs using the "delayed" traffic information $A_f(t-D)$ as follows: For all $0 \leq t < D$, let $H_n^f(t) = 0$, and for all $t \geq D$, we update the AQs according to:

$$H_n^f(t+1) = \max\left[H_n^f(t) - \mu^f_{[n,l^f(n)]}(t), 0\right] \\ +\theta A_f(t-D) + (1-\theta)\mu^f_{[u^f(n),n]}(t). \tag{5.36}$$

We then define the following Delayed-DESC algorithm to perform power allocation and scheduling.

Delayed-DESC: Choose a parameter $\theta \in (0, 1]$ independent of the network size and routing configurations. At every time slot, observe all queue values $Q_n^f(t-D)$ and $H_n^f(t)$, and do the followings

1. *Link Weight Computing:* For all $[m, n] \in \mathcal{L}$ such that there exists a flow f with $m = u^f(n)$, find the flow $f^*_{[m,n]}$ such that (ties broken arbitrarily):

   $$f^*_{[m,n]} = \underset{f\in\mathcal{F}: m=u^f(n)}{\arg\max} \left\{Q_m^f(t-D) - Q_n^f(t-D) + H_m^f(t) - (1-\theta)H_n^f(t)\right\}. \tag{5.37}$$

   Then define the *weight* of the link $[m, n]$ to be:

   $$W^*_{[m,n]}(t) = \max\Big[Q_m^{f^*_{[m,n]}}(t-D) - Q_n^{f^*_{[m,n]}}(t-D) \\ +H_m^{f^*_{[m,n]}}(t) - (1-\theta)H_n^{f^*_{[m,n]}}(t), 0\Big]. \tag{5.38}$$

   If no such f exists, i.e., $[m, n]$ is not used by any flow, define $W^*_{[m,n]}(t) = 0\ \forall\, t$.

2. *Power Allocation* and *Scheduling* are the same as DESC except that the weights are now given by (5.38).

3. *Queue Update:* Update $Q_n^f(t)$ and $H_n^f(t)$ for all n, f according to (5.5) and (5.36), respectively.

Specifically, Delayed-DESC is the same as DESC except that it uses $(\boldsymbol{Q}(t-D), \boldsymbol{H}(t))$ as the queue backlog vector to perform power and rate allocation, and the AQ values are updated according to (5.36) instead of (5.8). The performance of Delayed-DESC is summarized in the following theorem.

Theorem 20. *Suppose that there exists an $\epsilon > 0$ such that $\boldsymbol{\lambda} + \mathbf{1}\epsilon \in \boldsymbol{\Lambda}$. Then under the Delayed-DESC algorithm with parameter D, we have:*

$$\overline{\sum_f \sum_{n\in P_f} \frac{Q_n^f(t)}{K_f}} \leq \frac{2\sum_f K_f(1+D)B^2}{\epsilon} - \overline{\sum_f \sum_{n\in P_f} \frac{H_n^f(t)[\theta k_f(n) + (1-\theta)\epsilon]}{K_f}},$$

$$\overline{\sum_f \sum_{n\in P_f} H_n^f(t)} \leq \frac{2\sum_f K_f(1+D)B^2}{\theta\epsilon} - \overline{\sum_f \frac{Q_{s_f}^f(t)}{\theta}},$$

$$\overline{Lag(t)} \leq \frac{2\sum_f K_f(1+D)B^2}{\theta^2\epsilon} - \overline{\sum_f \frac{Q_{s_f}^f(t)}{\theta^2}},$$

where $\overline{x(t)}$ represents the expected time average of the sequence $\{x(t)\}_{t=0}^{\infty}$ and is defined in (5.12).

Note that since we typically only need a few bits to represent the $A_f(t)$ values, and we can pass this AQ information at a much faster time scale, i.e., not necessarily once per slot, the D value is typically very small. In this case, Theorem 20 states that Delayed-DESC performs nearly as well as DESC.

Proof. (Theorem 20) Let $\hat{\boldsymbol{Z}}(t) = (\boldsymbol{Q}(t-D), \boldsymbol{H}(t))$ be the delayed queue state and define the drift to be

$$\Delta(t) \triangleq \mathbb{E}\big\{L(t+1) - L(t) \mid \hat{\boldsymbol{Z}}(t)\big\},$$

where if $t - D < 0$ we define $Q_n^f(t-D) = 0$. Now using Lemma 7, we see that

$$\Delta(t) \leq C - \sum_f \sum_{n \in P_f} H_n^f(t) \mathbb{E}\big\{\mu^f_{[n,l^f(n)]}(t) \\ -(1-\theta)\mu^f_{[u^f(n),n]}(t) - \theta A_f(t-D) \mid \hat{\boldsymbol{Z}}(t)\big\} \\ - \sum_f \sum_{n \in P_f} \mathbb{E}\big\{Q_n^f(t)\big[\mu^f_{[n,l^f(n)]}(t) - \mu^f_{[u^f(n),n]}(t)\big] \mid \hat{\boldsymbol{Z}}(t)\big\}, \tag{5.39}$$

where $C = 2\sum_f K_f B^2$ and $B = \max[A_{max}, \mu_{max}]$. Denote the RHS of (5.39) as $\Delta_R(t)$. Using the fact that for any f and $n \in P_f$, we get

$$Q_n^f(t-D) - DB \leq Q_n^f(t) \leq Q_n^f(t-D) + DB,$$

we have:

$$\sum_f \sum_{n \in P_f} Q_n^f(t)\big[\mu^f_{[n,l^f(n)]}(t) - \mu^f_{[u^f(n),n]}(t)\big] \geq -\sum_f \sum_{n \in P_f} 2DB^2 \\ + \sum_f \sum_{n \in P_f} Q_n^f(t-D)\big[\mu^f_{[n,l^f(n)]}(t) - \mu^f_{[u^f(n),n]}(t)\big],$$

Plugging this into (5.39), we get

$$\Delta_R(t) \leq C + 2\sum_f K_f D B^2 \\ - \sum_f \sum_{n \in P_f} Q_n^f(t-D)\mathbb{E}\big\{\mu^f_{[n,l^f(n)]}(t) - \mu^f_{[u^f(n),n]}(t) \mid \hat{\boldsymbol{Z}}(t)\big\} \\ - \sum_f \sum_{n \in P_f} H_n^f(t)\mathbb{E}\big\{\mu^f_{[n,l^f(n)]}(t) - (1-\theta)\mu^f_{[u^f(n),n]}(t) - \theta A_f(t-D) \mid \hat{\boldsymbol{Z}}(t)\big\}. \tag{5.40}$$

We can similarly see that *the power and rate allocation of Delayed-DESC minimizes the RHS of (5.40).* Hence the above inequality holds if we plug in any alternative power and rate allocation policy. Thus under Delayed-DESC, we have:

$$\Delta(t) \leq C + 2\sum_f K_f D B^2 \\ - \sum_f \sum_{n \in P_f} Q_n^f(t-D)\mathbb{E}\big\{\mu^{*f}_{[n,l^f(n)]}(t) - \mu^{*f}_{[u^f(n),n]}(t) \mid \hat{\boldsymbol{Z}}(t)\big\} \\ - \sum_f \sum_{n \in P_f} H_n^f(t)\mathbb{E}\big\{\mu^{*f}_{[n,l^f(n)]}(t) - (1-\theta)\mu^{*f}_{[u^f(n),n]}(t) - \theta A_f(t-D) \mid \hat{\boldsymbol{Z}}(t)\big\}, \tag{5.41}$$

where $\mu^{*f}_{[n,l^f(n)]}(t)$ is the rate allocated to flow f over link $[n, l^f(n)]$ by any alternative policy. The rest of the proof follows by using a similar argument as in the proofs of Theorem 17 and 18. □

5.8 M-DESC for multi-path routing

In this section, we extend the DESC algorithm to the case where each flow can be routed via several acyclic paths to its destination. Specifically, we assume that each flow f uses a set of J_f paths, denoted by $\mathbb{P}_f = \{P_f^1, P_f^2, ..., P_f^{J_f}\}$.

We assume that each path is acyclic and that $P_f^j \cap P_f^k = \{s_f, d_f\}$ for all $P_f^j, P_f^k \in \mathbb{P}_f$ with $j \neq k$. That is, all paths are *node-disjoint* except for the source and destination nodes. [2] Such a setting allows more flexibility in routing packets to destinations. As in the single-path routing case, we denote by K_f^j the path length of the path P_f^j, and denote by $k_f^j(n)$ the order of node $n \in P_f^j$, for each node $n \in P_f^j$ (Recall that s_f is the 1^{st} node and d_f is the $(K_f^j + 1)^{th}$ node in path P_f^j.) In the following, it is also useful to denote by $\hat{s}_f^j$ the *second* node in the path P_f^j of the flow f (the first node in each path is always the source node). For any node $n \in P_f^j$ with $k_f^j(n) \geq 2$, we again use $u^f(n)$ to denote the set of upstream nodes of node n. Note that $u^f(n)$ contains only one node for any $n \neq d_f$, since all the paths are node-disjoint. For any $n \in P_f^j$ with $k_f^j(n) \leq K_f^j$, we use $l^f(n)$ to denote its set of downstream nodes. Similarly $l^f(n)$ contains only one node when $n \neq s_f$. In this case, an actual queue $Q_n^f(t)$ again evolves according to the queueing

[2] This assumption is made mainly for notation simplicity. The results can easily be generalized to the case where paths share nodes other than the source and the destination nodes.

dynamic (5.5) if n is not the source node for the flow f, i.e., $n \neq s_f$. When $n = s_f$, we see that $Q_n^f(t)$ evolves according to

$$Q_{s_f}^f(t+1) = \max[Q_{s_f}^f(t) - \sum_{P_f^j} \mu^f_{[s_f,\hat{s}_f^j]}(t), 0] + A_f(t). \tag{5.42}$$

Before stating the DESC algorithm for this multi-path routing case, we first state a theorem which can be proven similarly to Theorem 16, and which characterizes the capacity region in this multi-path routing case. In the theorem, we use the notion of a *feasible network rate splitting vector*. Specifically, we say that a vector $\boldsymbol{\gamma}_f = (\gamma_f^1, ..., \gamma_f^{J_f})^T$ is a feasible rate splitting vector for flow f if there exists an $\boldsymbol{S}$-only policy under which:

$$\mathbb{E}\{\mu^f_{[n,l^f(n)]}(t)\} \quad = \quad \gamma_f^j, \tag{5.43}$$

for all $n \in P_f^j$ with $k_f^j(n) \leq K_f^j$, and for all P_f^j. That is, γ_f^j represents the time average rate achieved over the path P_f^j for flow f. Here the expectation is taken with respect to the random channel states and the (possibly) random power allocation and routing actions. If there exists an $\boldsymbol{S}$-only policy that simultaneously achieves $\boldsymbol{\gamma}_f$ for each f, we call the vector $\overline{\boldsymbol{\gamma}} = (\boldsymbol{\gamma}_1, ..., \boldsymbol{\gamma}_M)^T$ a feasible network rate splitting vector.

Theorem 21. *The capacity region $\boldsymbol{\Lambda}$ in the multi-path routing case is the set of arrival rate vectors $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_M)^T \in \mathbb{R}_+^M$ such that there exists a feasible network rate spitting vector $\overline{\boldsymbol{\gamma}}$ with:*

$$\boldsymbol{\gamma}_f^T \cdot \mathbf{1} = \sum_j \mathbb{E}\{\mu^f_{[s_f,\hat{s}_f^j]}(t)\} = \lambda_f, \tag{5.44}$$

for all flows $f \in \mathcal{F}$. $\square$

We now describe the DESC algorithm for this multi-path case. Under this setting, we can no longer use the AQs as before. This is the due to the fact that the arrivals to the source will no longer be all routed via a single path. In this case, we again create a

set of AQs. However, for each AQ, we use the arrival to the second node $\hat{s}_f^j$ in each path as part of its input. Specifically, for each path P_f^j, we create $K_f^j - 1$ AQs as follows: for each $n \in P_f^j$ with $2 \leq k_f^j(n) \leq K_f^j$, we have:

$$H_n^f(t+1) = \max\left[H_n^f(t) - \mu^f_{[n,lf(n)]}(t), 0\right] + \theta\mu^f_{[s_f,\hat{s}_f^j]}(t) + (1-\theta)\mu^f_{[u^f(n),n]}(t). \quad (5.45)$$

For notation simplicity, we also define $H_{s_f}^f(t) = H_{d_f}^f(t) = 0$ for all t and all f. We now present the DESC algorithm for this multi-path routing case.

Multi-path DESC (M-DESC): Choose a parameter $\theta \in (0, 1]$ independent of the network size and routing configuration. At every time slot t, observe all queue values $Q_n^f(t)$ and $H_n^f(t)$, and do the following

(1) Link Weight Computing: For all $[m, n] \in \mathcal{L}$ such that there exists a flow f with $m \in u^f(n)$, compute a weight $W^f_{[m,n]}(t)$ for the flow f over $[m, n]$ as follows:

1. If $m = s_f$, $n \in P_f^j$, i.e., $n = \hat{s}_f^j$, then:

$$W^f_{[m,n]}(t) = \max\left[Q_m^f(t) - Q_n^f(t) - (1-\theta)H_n^f(t) - \theta\sum_{n' \in P_f^j} H_{n'}^f(t), 0\right]. \quad (5.46)$$

2. If $m \neq s_f$, $m, n \in P_f^j$, then:

$$W^f_{[m,n]}(t) = \max\left[Q_m^f(t) - Q_n^f(t) + H_m^f(t) - (1-\theta)H_n^f(t), 0\right]. \quad (5.47)$$

Find the flow $f^*_{[m,n]}$ such that (ties broken arbitrarily):

$$f^*_{[m,n]} = \underset{f \in \mathcal{F}: m \in u^f(n)}{\arg\max} W^f_{[m,n]}(t).$$

Then define the *weight* of the link $[m, n]$ to be:

$$W^*_{[m,n]}(t) = W^{f^*_{[m,n]}}_{[m,n]}(t). \quad (5.48)$$

If no such f exists, i.e., $[m, n]$ is not used by any flow, define $W^*_{[m,n]}(t) = 0\ \forall\, t$.

(2) Power Allocation: same as DESC, except that the weights are now given by (5.48).

(3) Routing: Define transmission rates as follows:

$$\mu^f_{[m,n]}(t) = \begin{cases} \mu_{[m,n]}(t) & \text{if } f = f^*_{[m,n]} \text{ and } W^*_{[m,n]}(t) > 0 \\ 0 & \text{else.} \end{cases}$$

For each link $[m,n]$, transmit flow $f^*_{[m,n]}$ data according to the rate $\mu_{[m,n]}(t)$. If node m does not have enough packets to send over all its outgoing links, null bits are delivered.

(4) Queue Update: Update $Q^f_n(t)$ and $H^f_n(t)$, $\forall\, f, P^j_f, n \in P^j_f, n \neq s_f$, according to (5.5) and (5.45), respectively. Update $Q^f_{s_f}(t)$ according to (5.42) for all f.

We note that M-DESC is very similar to the DESC algorithm. Indeed, the main difference here is that M-DESC has to decide which path to route the packets through. For this part, we see from (5.46) that the weights of the links from the source nodes to their second-hop nodes are calculated differently. We also note that (5.46) requires the AQ values over the entire routing path. Such information is easy to obtain if centralized control is available. In the case when obtaining such information incurs a nonzero delay, one can also use an approach similar to the one used by Delayed-DESC in Section 5.7. The following theorem states the performance results of M-DESC:

Theorem 22. *Suppose $\boldsymbol{\lambda}$ is such that there exists a feasible network rate splitting vector $\overline{\boldsymbol{\gamma}}$ that achieves $\boldsymbol{\gamma}_f + \epsilon\mathbf{1}$ for each flow f with $\sum_j \gamma^j_f = \lambda_f$ for some $\epsilon > 0$. Then under M-DESC, we have the following bounds:*

$$\overline{\sum_f \sum_{P^j_f} \sum_{n \in P^j_f} H^f_n(t)} \leq \frac{\tilde{C}}{\theta\epsilon} - \overline{\sum_f \sum_{P^j_f} Q^f_{\hat{s}^j_f}(t)}/\theta, \tag{5.49}$$

$$\overline{\sum_f Q^f_{s_f}(t) c_f} + \overline{\sum_f \sum_{P^j_f} \sum_{n \in P^j_f, n \neq s_f} \frac{Q^f_n(t)}{K^j_f}} \leq \frac{\tilde{C}}{\epsilon} - \overline{\sum_f \sum_{P^j_f} \sum_{n \in P^j_f} H^f_n(t) \frac{\theta(k^j_f(n) - 1) + (1-\theta)}{K^j_f}}. \tag{5.50}$$

Here $\tilde{C} = B^2 \sum_f \left[\frac{1}{2}(J_f^2+1) + 2\sum_{P_f^j}(K_f^j - 1)\right]$, $c_f = \sum_{P_f^j} \frac{1}{K_f^j}$, *and the notation* $\overline{x(t)}$ *is defined in (5.12).*

We note that the performance result for M-DESC has one additional requirement: we need each element of the rate splitting vector $\boldsymbol{\gamma}_f$ for each flow f to be feasible after an ϵ increment. This implies that the arrival rate of flow f is at ϵJ_f distance from the boundary of the capacity region (since flow f is served by J_f paths). This additional requirement is due to the fact that when multiple routes are available, M-DESC will explore all of them. Hence we must take into account the effective load on each path when analyzing the congestion in the entire network. We also note that similar results as Theorem 18 on the network service lag can be obtained in this case. We can similarly see from (5.49) that if $M = O(1)$, $J_f = O(1)$ and $\epsilon = O(1)$ for all f, then the average total network service lag will again only grow linearly in the network size.

Proof. (Theorem 22) In this case, we use the following Lyapunov function that is similar to the one defined in (5.22):

$$L(\boldsymbol{Q}(t), \boldsymbol{H}(t)) = \frac{1}{2}\sum_f \sum_{P_f^j} \sum_{n \in P_f^j} \Big([Q_n^f(t)]^2 + [H_n^f(t)]^2 \Big).$$

We similarly define $\boldsymbol{Z}(t) = (\boldsymbol{Q}(t), \boldsymbol{H}(t))$. Using the same approach as in Lemma 7, we can compute the drift as:

$$\begin{aligned}
\Delta(t) \leq \tilde{C} &- \sum_f \mathbb{E}\big\{ Q_{s_f}^f(t) \big[\sum_{P_f^j} \mu^f_{[s_f, \hat{s}_f^j]}(t) - A_f(t) \big] \mid \boldsymbol{Z}(t) \big\} \\
&- \sum_f \sum_{P_f^j} \sum_{n \in P_f^j : n \neq s_f} \mathbb{E}\big\{ Q_n^f(t) \big[\mu^f_{[n, l^f(n)]}(t) - \mu^f_{[u^f(n), n]}(t) \big] \mid \boldsymbol{Z}(t) \big\} \qquad (5.51) \\
&- \sum_f \sum_{P_f^j} \sum_{n \in P_f^j} \mathbb{E}\big\{ H_n^f(t) \big[\mu^f_{[n, l^f(n)]}(t) \\
&\qquad\qquad - \theta \mu^f_{[s_f, \hat{s}_f^j]}(t) - (1-\theta)\mu^f_{[u^f(n), n]}(t) \big] \mid \boldsymbol{Z}(t) \big\}.
\end{aligned}$$

where $\tilde{C} = B^2 \sum_f \left[(J_f^2+1) + 2\sum_{P_f^j}(K_f^j - 1)\right]$. Now by rearranging the terms, we have:

$$\begin{aligned}\Delta(t) \leq \tilde{C} &+ \sum_f \mathbb{E}\big\{Q^f_{s_f}(t)A_f(t) \mid \boldsymbol{Z}(t)\big\} \\ &- \sum_f \sum_{P_f^j} \mathbb{E}\big\{\mu^f_{[s_f,\hat{s}_f^j]}(t)\big[Q^f_{s_f}(t) - Q^f_{\hat{s}_f^j}(t) \\ &\qquad -(1-\theta)H^f_{\hat{s}_f^j}(t) - \theta\sum_{n\in P_f^j} H_n^f(t)\big] \mid \boldsymbol{Z}(t)\big\} \\ &- \sum_f \sum_{P_f^j} \sum_{n\in P_f^j : n\neq s_f} \mathbb{E}\big\{\mu^f_{[n,l^f(n)]}(t)\big[Q_n^f(t) - Q^f_{l^f(n)}(t) \\ &\qquad +H_n^f(t) - (1-\theta)H^f_{l^f(n)}(t)\big] \mid \boldsymbol{Z}(t)\big\}. \end{aligned} \tag{5.52}$$

Thus we see that the M-DESC algorithm *chooses a power allocation vector and allocates rates to the flows to minimize the RHS of (5.51) at every time slot.* Hence using the same argument as in the proof of Theorem 17, we can now plug into (5.52) an $\boldsymbol{S}$-only policy that achieves for each flow f,

$$\mathbb{E}\big\{\mu^f_{[s_f,\hat{s}_f^j]}(t) \mid \boldsymbol{Z}(t)\big\} = \gamma_f^j, \tag{5.53}$$

with $\sum_j \gamma_f^j = \lambda_f$, $\forall\, f$, and for each $n \in P_f^j$ with $2 \leq k_f^j(n) \leq K_f^j$ that:

$$\mathbb{E}\big\{\mu^f_{[n,l^f(n)]}(t) \mid \boldsymbol{Z}(t)\big\} = \gamma_f^j + \epsilon. \tag{5.54}$$

Using (5.51), we thus obtain:

$$\Delta(t) \leq \tilde{C} - \sum_f \sum_{P_f^j} Q^f_{\hat{s}_f^j}(t)\epsilon - \sum_f \sum_{P_f^j}\sum_{n\in P_f^j} \theta\epsilon H_n^f(t) - \sum_f \sum_{P_f^j} H^f_{\hat{s}_f^j}(t)(1-\theta)\epsilon.$$

Using Theorem 19 and the fact that $H^f_{\hat{s}_f^j}(t) \geq 0$, $\forall\, t$, we have:

$$\overline{\sum_f \sum_{P_f^j}\sum_{n\in P_f^j} H_n^f(t)} \leq \frac{\tilde{C}}{\theta\epsilon} - \overline{\sum_f \sum_{P_f^j} Q^f_{\hat{s}_f^j}(t)}/\theta. \tag{5.55}$$

This proves (5.49).

Now we plug in (5.52) another $\boldsymbol{S}$-only policy that achieves for each flow f:

$$\mathbb{E}\big\{\mu^f_{[n,l^f(n)]}(t) \mid \boldsymbol{Z}(t)\big\} = \gamma_f^j + \epsilon\frac{k_f^j(n)}{K_f^j}, \tag{5.56}$$

for each $n \in P_f^j$ with $1 \le k_f^j(n) \le K_f^j$, where $\sum_j \gamma_f^j = \lambda_f,\ \forall\ f$. Using (5.51) again, we see that:

$$\Delta(t) \le \tilde{C} - \sum_f Q_{s_f}^f(t)\epsilon\big(\sum_{P_f^j}\frac{1}{K_f^j}\big) - \sum_f\sum_{P_f^j}\sum_{n\in P_f^j, n\neq s_f} Q_n^f(t)\frac{\epsilon}{K_f^j}$$
$$-\sum_f\sum_{P_f^j}\sum_{n\in P_f^j} H_n^f(t)\epsilon\frac{\theta(k_f^j(n)-1)+(1-\theta)}{K_f^j}.$$

Now denoting $c_f = \sum_{P_f^j}\frac{1}{K_f^j}$, we have:

$$\overline{\sum_f Q_{s_f}^f(t)c_f} + \overline{\sum_f\sum_{P_f^j}\sum_{n\in P_f^j, n\neq s_f}\frac{Q_n^f(t)}{K_f^j}}$$
$$\le \frac{\tilde{C}}{\epsilon} - \overline{\sum_f\sum_{P_f^j}\sum_{n\in P_f^j} H_n^f(t)\frac{\theta(k_f^j(n)-1)+(1-\theta)}{K_f^j}}.$$

This proves (5.50) and completes the proof of Theorem 22. □

It is not difficult to show, as in [Nee03], that under this multi-path routing case, the congestion bound under DRPC is given by:

$$\overline{\sum_f Q_{s_f}^f(t)c_f} + \overline{\sum_f\sum_{P_f^j}\sum_{n\in P_f^j, n\neq s_f}\frac{Q_n^f(t)}{K_f^j}} \le \frac{\hat{C}}{\epsilon}, \tag{5.57}$$

where $\hat{C} = B^2\sum_f\big[\frac{1}{2}(J_f^2+1) + \sum_{P_f^j}(K_f^j-1)\big]$. Comparing this bound with (5.49) and (5.50), we see again that the backlog bound of M-DESC can potentially be better than that of DRPC, and the rate convergence speed is faster under M-DESC.

5.9 Simulation

Here we provide simulation results of our algorithms. For simplicity, we only simulate the DESC algorithm and the M-DESC algorithm, and compare them with the DRPC algorithm.

5.9.1 The single-path case

The network topology and flow configuration are shown in Fig. 5.3. We assume that the channel conditions are independent and each link $[m, n]$ is i.i.d., every slot being ON with probability 0.8 and OFF with probability 0.2. When the channel is "ON," we can allocate one unit of power and transmit two packets; otherwise we can send zero packets. We further assume that all links can be activated without affecting others. However, a node can only transmit over one link at a time, though it can simultaneously receive packets from multiple nodes. Each flow f_i is an independent Bernoulli process with $A_{f_i}(t) = 2$ with probability $\lambda_i/2$ and $A_{f_i}(t) = 0$ else. The rate vector is given by $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)^T = (0.8, 0.4, 0.2, 0.6)^T$. We simulate the system with $(h = \frac{\eta}{2}, \upsilon = \frac{\eta}{2})$, where η, h and υ are parameters in Fig. 5.3. Note that in this case, $N = \frac{3\eta}{2} + 7$. The η value is chosen to be $\{10, 50, 100, 200, 500\}$. We use $\theta = 0.5$.

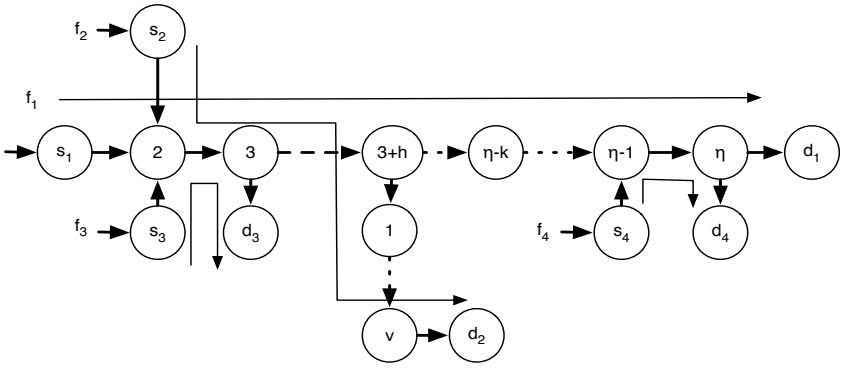

Figure 5.3: A Network with 4 Flows. η is f_1's path length, h measures the path overlap length of f_1 and f_2, and υ is the vertical path length of f_2.

Fig. 5.4 and Fig. 5.5 show the simulation results. Fig. 5.4 shows that under DESC, the average total backlogs of Flow 1 and 2 scale only *linearly* in N. This is in contrast to

the example provided in [BSS09], which shows that the average backlog grows quadratically with N under the usual Max-Weight scheduling policy. We also see that the average total backlog of Flows 3 and 4 remains roughly the same. This is intuitive, as their path lengths do not grow with N. Interestingly, we observe that $\overline{\sum_{n\in P_f} H_n^f(t)} \geq \overline{\sum_{n\in P_f} Q_n^f(t)}$, $\forall\, f$. By equation (5.14) of Theorem 17, this implies that :

$$\overline{\sum_f \sum_{n\in P_f} Q_n^f(t)} \leq \overline{\sum_f \sum_{n\in P_f} H_n^f(t)} \leq \frac{2\sum_f K_f B^2}{\theta\epsilon}.$$

Since we also have $\sum_f K_f = O(N)$ and $\epsilon = \Theta(1)$ in this example, we see that indeed $\overline{\sum_f \sum_{n\in P_f} Q_n^f(t)} = O(N)$ in this case. This suggests that DESC can potentially be a way to achieve *delay-order-optimal* scheduling in general multihop networks. Fig. 5.5 shows that the total average rates (running averages) allocated to Flows 1 and 2 over their paths, i.e., $\frac{1}{tK_{f_1}} \sum_{n_i\in P_{f_1}} \mu^{f_1}_{[n_i,l^{f_1}(n_i)]}[0,t-1]$ and $\frac{1}{tK_{f_2}} \sum_{n_j\in P_{f_2}} \mu^{f_2}_{[n_j,l^{f_2}(n_j)]}[0,t-1]$ with $\mu^{f_k}_{[n_i,l^{f_k}(n_i)]}[0,t-1] = \sum_{\tau=0}^{t-1} \mu^{f_k}_{[n_i,l^{f_k}(n_i)]}(\tau)$, converge quickly from above to the actual average arrival rates i.e., $\frac{1}{t}A_f[0,t-1]$; on the other hand the corresponding rates under DRPC converge very slowly from below to the actual average arrival rate. These plots suggest that the poor delay performance of many Max-Weight type algorithms in multihop networks can be due to slow convergence of the corresponding service rates.

5.9.2 The multi-path case

We now look at the multi-path routing case. We consider a simple network having two paths as shown in Fig. 5.6. Each flow uses both paths. The two paths are given by $P_1 = (n_1, n_2, ..., n_{K_1})$ and $P_2 = (m_1, m_2, ..., m_{K_2})$, where $K_1 = \eta, K_2 = \eta/2$, and η is chosen to be $\{10, 50, 100, 200, 500\}$. Similar as above, we assume that each link is i.i.d. ON with probability 0.6, and OFF with probability 0.4. When ON, we can allocate one

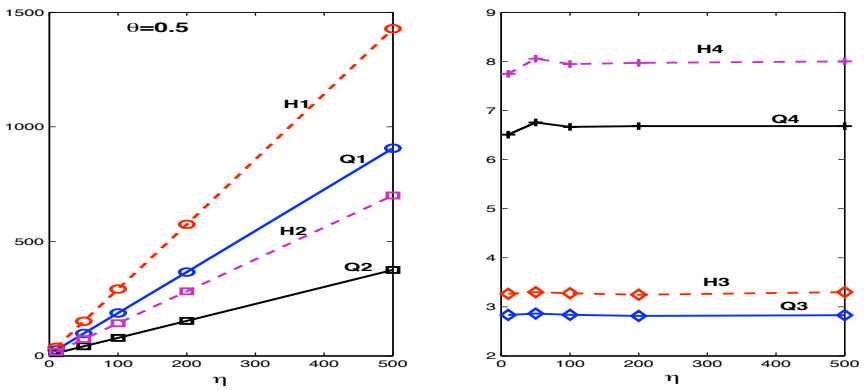

Figure 5.4: Q_i and H_i, $i = 1, 2, 3, 4$, are the average total *actual* and AQ backlog sizes of flow i, respectively.

unit of power to serve 2 packets; otherwise we cannot serve any. We assume that all the transmissions over the links are independent, and do not affect others. A node can simultaneously receive from multiple nodes, but can only send over one outgoing link. $A_1(t)$ is i.i.d., being 2 with probability $\lambda_1 = 0.5$, and being 0 else. $A_2(t)$ is i.i.d., being 2 with probability $\lambda_2 = 0.3$ and being 0 else. We use $\theta = 0.5$.

We see from Fig.5.7 that as in the single path routing case, the network backlog only increases linearly in the network size. In this case, though the average total actual backlog size is not always below the average total AQ backlog size, as in Fig. 5.4, we see that the total AQ size is always above the total actual queue size excluding the source nodes. As in the single-path case, this result can also be used together with (5.49) to show that in this example, M-DESC achieves a total network backlog that grows only linearly in the network size.

Similar to Fig. 5.5 in the single path routing case, Fig. 5.8 also shows the aggregate rates of the two flows over their paths versus their arrival rates. We again see that the rates converge faster under M-DESC than under the DRPC algorithm.

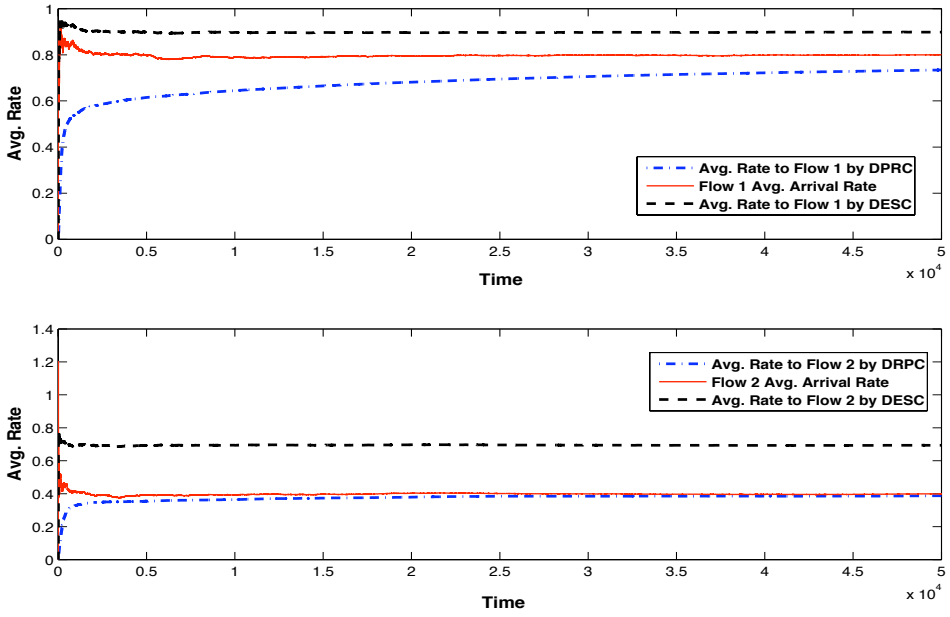

Figure 5.5: UP: the average rate allocated to Flow 1 ($\eta = 100$); DOWN: the average rate allocated to Flow 2 ($\eta = 100$).

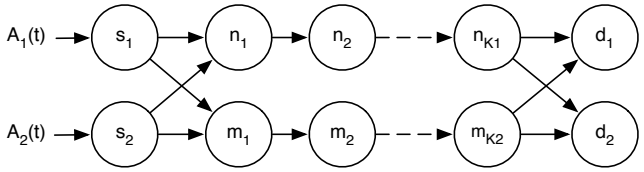

Figure 5.6: Two flows going through a network with two paths.

5.10 Further discussion

The redundant constraint approach can also be applied to network utility optimization problems. In this case, the method can also be combined with any of the delay-reduction techniques, i.e., FQLA, LIFO-Backpressure and LIFO^p-Backpressure, to further improve the delay performance of the algorithms.

As in Section 4.2.4, the assumption that there exists $\epsilon > 0$ such that $\boldsymbol{\lambda} + \epsilon \mathbf{1} \in \Gamma$ is important for deriving the queueing bounds. When this assumption is violated, the

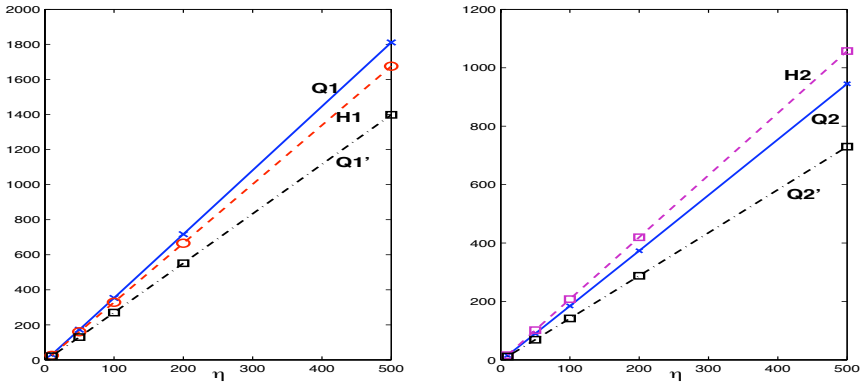

Figure 5.7: Here Q_i and H_i, $i = 1, 2$ denote the time average actual and AQ backlog sizes. Q'_i, $i = 1, 2$ denotes the average total actual backlog without the source nodes.

arrival rate vector is either on the boundary, or outside the capacity region. In both cases, the network congestion will go unbounded.

5.11 Chapter summary

In this chapter, we consider the problem of delay-efficient scheduling in general multihop networks. We develop a Max-Weight type Delay Efficient SCheduling Algorithm (DESC). We show that DESC is throughput optimal and derive a queueing bound which can potentially be better than previous congestion bounds on Max-Weight type algorithms. We also show that under DESC, the time required for the allocated rates to converge to their target values scales only linearly in the network size. This contrasts with the usual Max-Weight algorithms, which typically require a time that is at least quadratic in the network size.

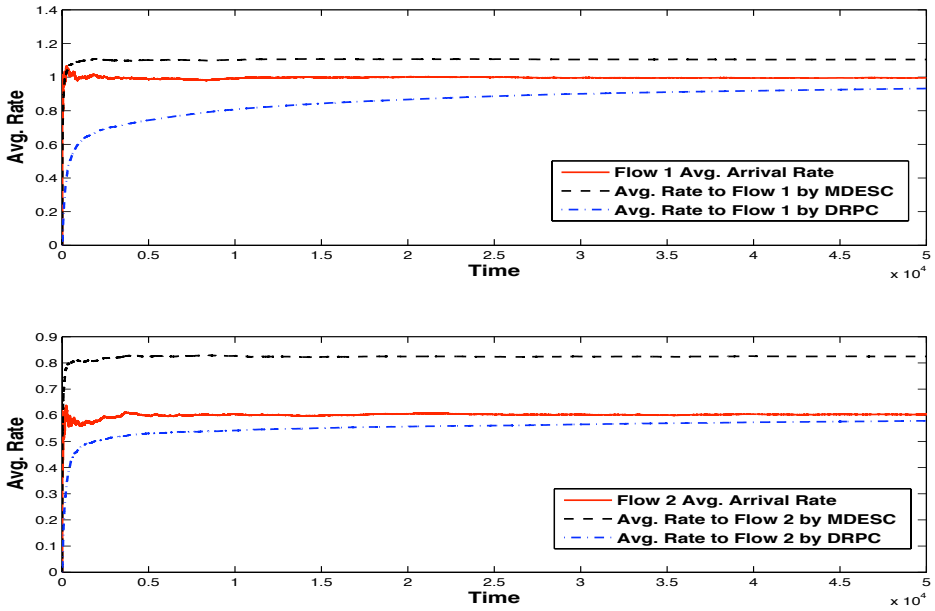

Figure 5.8: UP: the average rate allocated to Flow 1 ($\eta = 100$); DOWN: the average rate allocated to Flow 2 ($\eta = 100$).

5.12 Proofs of the chapter

5.12.1 Proof of Lemma 7

Proof. Define $B = \max[A_{max}, \mu_{max}]$. From the queueing dynamic equation (5.5) we have for all node $n \in P_f$ with $k_f(n) \leq K_f$ that:

$$\begin{aligned}
[Q_n^f(t+1)]^2 &= \left([Q_n^f(t) - \mu^f_{[n,l^f(n)]}(t)]^+ + \mu^f_{[u^f(n),n]}(t)\right)^2 \\
&= \left([Q_n^f(t) - \mu^f_{[n,l^f(n)]}(t)]^+\right)^2 + [\mu^f_{[u^f(n),n]}(t)]^2 \\
&\qquad +2([Q_n^f(t) - \mu^f_{[n,l^f(n)]}(t)]^+)\mu^f_{[u^f(n),n]}(t) \\
&\leq [Q_n^f(t) - \mu^f_{[n,l^f(n)]}(t)]^2 + B^2 + 2Q_n^f(t)\mu^f_{[u^f(n),n]}(t).
\end{aligned}$$

The inequality holds since $([Q_n^f(t) - \mu^f_{[n,l^f(n)]}(t)]^+)\mu^f_{[u^f(n),n]}(t) \leq Q_n^f(t)\mu^f_{[u^f(n),n]}(t)$, and for any $x \in \mathbb{R}$, we have $([x]^+)^2 \leq x^2$. We also use in the above equation that if $k_f(n) = 1$,

i.e., node n is the source node of flow f, then $\mu^f_{[u^f(n),n]}(t) = A_f(t)$ for all t. Thus by expanding the term $[Q^f_n(t) - \mu^f_{[n,l^f(n)]}(t)]^2$, we have for all $n \in P_f$ with $k_f(n) \leq K_f$ that:

$$[Q^f_n(t+1)]^2 \leq [Q^f_n(t)]^2 + 2B^2 - 2Q^f_n(t)\big[\mu^f_{[n,l^f(n)]}(t) - \mu^f_{[u^f(n),n]}(t)\big].$$

Note that if $k_f(n) = K_f + 1$, i.e., $n = d_f$, we have $Q^f_n(t) = 0$ for all t, and so $[Q^f_n(t+1)]^2 - [Q^f_n(t)]^2 = 0$, $\forall t$. Summing the above over all n, f, and multiply by $\frac{1}{2}$, we have:

$$\begin{aligned} &\frac{1}{2}\sum_f \sum_{n\in P_f} [Q^f_n(t+1)]^2 - \frac{1}{2}\sum_f \sum_{n\in P_f} [Q^f_n(t)]^2 \\ &\qquad \leq Y_1 B^2 - \sum_f \sum_{n\in P_f} Q^f_n(t)\big[\mu^f_{[n,l^f(n)]}(t) - \mu^f_{[u^f(n),n]}(t)\big], \end{aligned} \tag{5.58}$$

where $Y_1 = \sum_f K_f$. Repeat the above argument on the term $\sum_f \sum_{n\in P_f}[H^f_n(t+1)]^2 - \sum_f \sum_{n\in P_f}[H^f_n(t)]^2$, we obtain:

$$\begin{aligned} &\frac{1}{2}\sum_f \sum_{n\in P_f} [H^f_n(t+1)]^2 - \frac{1}{2}\sum_f \sum_{n\in P_f} [H^f_n(t)]^2 \\ &\qquad \leq Y_1 B^2 - \sum_f \sum_{n\in P_f} H^f_n(t)[\mu^f_{[n,l^f(n)]}(t) - (1-\theta)\mu^f_{[u^f(n),n]}(t) - \theta A_f(t)]. \end{aligned} \tag{5.59}$$

Now adding (5.58) to (5.59), taking expectations conditioning on $\boldsymbol{Z}(t) = (\boldsymbol{Q}(t), \boldsymbol{H}(t))$, and letting $C = 2Y_1B^2 = 2\sum_f K_f B^2$ proves the lemma. □

Part III

Utility Optimal Scheduling for Complex Networks

Chapter 6

Resolving underflows in complex network scheduling problems

In this chapter, we consider the problem of optimal scheduling in general complex networks that involve the "no-underflow" constraint. This network class contains the important class of processing networks, which are generalizations of traditional communication networks. Such scheduling problems are usually hard to solve, and the dynamic programming technique is often used. We instead develop the novel *perturbed* Max-Weight approach for algorithm design for these networks. This approach has low complexity and avoids the "curse of dimensionality" of dynamic programming. The results developed in this chapter can be applied to areas such as manufacturing networks, information processing, network coding and energy harvesting networks.

6.1 A data processing example

In this section, we study a data processing example and develop the Perturbed Max-Weight algorithm (PMW) in this case. This example demonstrates the main idea of this chapter. We later present the general model in Section 6.2.

6.1.1 Network settings

We consider a network shown in Fig. 6.1, where the network performs 2-stage data processing for the arriving data. In this network, there are two random data streams

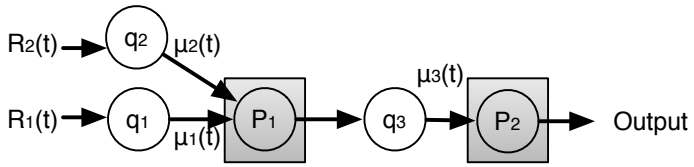

Figure 6.1: An example network consisting of three queues q_1, q_2, q_3 and two processors P_1, P_2.

$R_1(t), R_2(t)$, which represent, e.g., sensed data that arrives, or video and voice data that need to be mixed. We assume that $R_i(t) = 1$ or 0, equally likely, for $i = 1, 2$. At every time slot, the network controller first decides whether or not to admit the new arrivals, given that accepting any one new arrival unit incurs a cost of 1. The controller then has to decide how to activate the two processors P_1, P_2 for data processing. We assume that both processors can be activated simultaneously. When activated, P_1 consumes one unit of data from both q_1 and q_2, and generates one unit of fused data into q_3. This data needs further processing that is done by P_2. When P_2 is activated, it consumes one unit of data from q_3, and generates one unit of processed data. We assume that each unit of successfully fused and processed data generates a profit of $p(t)$, where $p(t)$ is i.i.d.

and takes value 3 or 1 with equal probabilities. The network controller's objective is to maximize the average utility, i.e., profit minus cost, subject to queue stability.

For ease of presenting the general model later, we define a *network state* $S(t) = (R_1(t), R_2(t), p(t))$, [1] which describes the current network random state. We also denote the controller's *action* at time t by $x(t) = (D_1(t), D_2(t), I_1(t), I_2(t))$, where $D_j(t) = 1$ ($D_j(t) = 0$) means to admit (reject) the new arrivals into queue j, and $I_i(t) = 1$ ($I_i(t) = 0$) means that processor P_i is activated (turned off). The following *no-underflow constraints* must be met at all times when we activate processors P_1, P_2:

$$I_1(t) \leq q_1(t), I_1(t) \leq q_2(t), I_2(t) \leq q_3(t). \tag{6.1}$$

That is, $I_1(t) = 1$ only when q_1 and q_2 are both nonempty, and $I_2(t) = 1$ only if q_3 is nonempty. Note that [JW09] is the first paper to identify such no-underflow constraints and propose an explicit solution that shows that the fraction of time that these constraints are violated converges to zero under certain network assumptions. Here we propose a different approach that ensures that the constraints are never violated, and holds for a broader class of problems. Subject to (6.1), we can write the amount of arrivals into q_1, q_2, q_3, and the service rates of the queues at time t as functions of the network state $S(t)$ and the action $x(t)$, i.e.,

$$A_j(t) = A_j(S(t), x(t)) = D_j(t)R_j(t),\ j = 1, 2, \quad A_3(t) = A_3(S(t), x(t)) = I_1(t). \tag{6.2}$$

$$\mu_j(t) = \mu_j(S(t), x(t)) = I_1(t),\ j = 1, 2, \quad \mu_3(t) = \mu_3(S(t), x(t)) = I_2(t). \tag{6.3}$$

Then we see that the queues evolve according to the following:

$$q_j(t+1) \quad = \quad q_j(t) - \mu_j(t) + A_j(t),\ j = 1, 2, 3,\ \forall\, t. \tag{6.4}$$

[1] The network state here contains just $R_1(t)$, $R_2(t)$ and $p(t)$. More complicated settings, where the amount consumed from queues may also depend on the random link conditions between queues and processors can also be modeled by incorporating the link components into the network state, e.g., [HN11a].

The *instantaneous utility* is given by:

$$f(t) = f(S(t), x(t)) = p(t)I_2(t) - D_1(t)R_1(t) - D_2(t)R_2(t). \tag{6.5}$$

The goal is to maximize the time average value of $f(t)$ subject to network stability.

The constraint (6.1) greatly complicates the design of an optimal scheduling algorithm. This is because the decision made at time t may affect the queue states in future time slots, which can in turn affect the set of possible actions in the future. In the following, we develop the Perturbed Max-Weight algorithm (PMW) for this example. The idea of PMW is to use the usual Max-Weight algorithm, but perturb the weights so as to push the queue sizes towards certain nonzero values. By carefully designing the perturbation, we can simultaneously ensure that the queues always have enough data for processing and the achieved utility is close to optimal.

6.1.2 The perturbed Max-Weight algorithm (PMW)

We now present the construction of the PMW algorithm for this simple example (This is extended to general network models in Section 6.4.) To start, we first define a *perturbation vector* $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)^T$ and the Lyapunov function $L(t) = \frac{1}{2}\sum_{j=1}^{3}[q_j(t) - \theta_j]^2$. We then define the one-slot conditional drift as:

$$\Delta(t) = \mathbb{E}\big\{L(t+1) - L(t) \mid \boldsymbol{q}(t)\big\}, \tag{6.6}$$

where the expectation is taken over the random network state $S(t)$ and the randomness over the actions. Using the queueing dynamics (6.4), it is easy to obtain that:

$$\Delta(t) \leq B - \sum_{j=1}^{3}\mathbb{E}\big\{(q_j(t) - \theta_j)[\mu_j(t) - A_j(t)] \mid \boldsymbol{q}(t)\big\},$$

where $B = 3$. Now we use the "drift-plus-penalty" approach, i.e., QLA, in Chapter 4 to design our algorithm for this problem. To do so, we define a control parameter $V \geq 1$, which will affect our utility-backlog tradeoff, and add to both sides the term $-V\mathbb{E}\big\{f(t) \mid \boldsymbol{q}(t)\big\}$ to get: [2]

$$\Delta(t) - V\mathbb{E}\big\{f(t) \mid \boldsymbol{q}(t)\big\} \leq B - V\mathbb{E}\big\{f(t) \mid \boldsymbol{q}(t)\big\} - \sum_{j=1}^{3} \mathbb{E}\big\{(q_j(t) - \theta_j)[\mu_j(t) - A_j(t)] \mid \boldsymbol{q}(t)\big\}. \tag{6.7}$$

Denote $\Delta_V(t) = \Delta(t) - V\mathbb{E}\big\{f(t) \mid \boldsymbol{q}(t)\big\}$, and plug (6.2), (6.3) and (6.5) into the above, to get:

$$\begin{aligned} \Delta_V(t) \quad \leq \quad & B + \mathbb{E}\big\{D_1(t)R_1(t)[q_1(t) - \theta_1 + V] \mid \boldsymbol{q}(t)\big\} \\ & + \mathbb{E}\big\{D_2(t)R_2(t)[q_2(t) - \theta_2 + V] \mid \boldsymbol{q}(t)\big\} \\ & - \mathbb{E}\big\{I_2(t)[q_3(t) - \theta_3 + p(t)V] \mid \boldsymbol{q}(t)\big\} \\ & - \mathbb{E}\big\{I_1(t)[q_1(t) - \theta_1 + q_2(t) - \theta_2 - (q_3(t) - \theta_3)] \mid \boldsymbol{q}(t)\big\}. \end{aligned} \tag{6.8}$$

We now develop our PMW algorithm by choosing an action at every time slot to *minimize the right-hand side (RHS) of (6.8)* subject to (6.1). The algorithm then works as follows:

PMW: At every time slot, observe $S(t)$ and $\boldsymbol{q}(t)$, and do the following:

1. Data Admission: For each $j = 1, 2$, choose $D_j(t) = 1$, i.e., admit the new arrivals to q_j if:

$$q_j(t) - \theta_j + V < 0, \tag{6.9}$$

else set $D_j(t) = 0$ and reject the arrivals.

[2] The construction is the same as QLA in Section 4.1.2 of Chapter 4.

2. Processor Activation: Choose $I_1(t) = 1$, i.e., activate processor P_1, if $q_1(t) \geq 1$, $q_2(t) \geq 1$, and

$$q_1(t) - \theta_1 + q_2(t) - \theta_2 - (q_3(t) - \theta_3) > 0, \tag{6.10}$$

else choose $I_1(t) = 0$. Similarly, choose $I_2(t) = 1$, i.e., activate processor P_2, if $q_3(t) \geq 1$, and

$$q_3(t) - \theta_3 + p(t)V > 0, \tag{6.11}$$

else choose $I_2(t) = 0$.

3. Queueing update: Update $q_j(t)$, $\forall\, j$, according to (6.4).

One important thing to notice here is that if we use the usual Max-Weight algorithm, i.e., do not use perturbation and set $\theta_j = 0$, $\forall\, j$, then (6.9) implies that we admit a new arrival to q_j only when $q_j + V < 0$, which is impossible because $q_j \geq 0$ and $V \geq 1$. Hence, this example clearly demonstrates the fact that *the usual Max-Weight may not be applicable to processing network problems*. Below, we show that the use of perturbation effectively resolves this problem. The fundamental reason why Max-Weight does not work, and that perturbation is able to resolve this problem are discussed in Section 6.8.

6.1.3 Performance of PMW

Here we analyze the performance of PMW. We first prove the following important claim: *under a proper $\boldsymbol{\theta}$ vector, PMW minimizes the RHS of (6.8) over all possible policies of arrival admission and processor activation, including those that choose actions regardless of the constraint (6.1)*. We then use this claim to analyze the performance of PMW, by comparing the value of the RHS of (6.8) under PMW to that under an alternative policy.

To prove the claim, we first see that the policy that minimizes the RHS of (6.8) without the constraint (6.1) differs from PMW only in the processor activation part, where PMW also considers the constraints $q_1(t) \geq 1$, $q_2(t) \geq 1$ and $q_3(t) \geq 1$. Thus if one can show that these constraints are indeed redundant in the PMW algorithm under a proper $\boldsymbol{\theta}$ vector, i.e., one can activate the processors without considering them but still ensure them, then PMW minimizes the RHS of (6.8) over all possible policies. In the following, we use the following θ_j values:

$$\theta_1 = 2V + 1, \ \theta_2 = 2V + 1, \ \theta_3 = 3V + 1. \tag{6.12}$$

We also assume that $q_j(0) = 1$ for all $j = 1, 2, 3$. This can easily be satisfied by storing an initial backlog in the queues.

We now look at the queue sizes $q_j(t), j = 1, 2, 3$. From (6.11), P_2 is activated, i.e., $I_2(t) = 1$ if and only if:

$$q_3(t) \geq \theta_3 - p(t)V + 1, \quad \text{and} \quad q_3(t) \geq 1. \tag{6.13}$$

Since $p(t) = 3$ or 1, we see that $I_2(t) = 1$ whenever $q_3(t) \geq \theta_3 - V + 1$, but $I_2(t) = 0$ unless $q_3(t) \geq \theta_3 - 3V + 1$. Since q_3 can receive and deliver at most one unit of data at a time, we get:

$$\theta_3 - V + 1 \geq q_3(t) \geq \theta_3 - 3V, \ \forall \ t. \tag{6.14}$$

Using $\theta_3 = 3V + 1$, this implies:

$$2V + 2 \geq q_3(t) \geq 1, \quad \forall \ t. \tag{6.15}$$

This shows that with $\theta_3 = 3V$, the activations of P_2 are always feasible even if the constraint $q_3(t) \geq 1$ is removed. We now look at $q_1(t)$ and $q_2(t)$. We see from (6.9) that for $\theta_1, \theta_2 \geq V$, we have:

$$q_j(t) \leq \theta_j - V, \ j = 1, 2. \tag{6.16}$$

Also, using (6.10) and (6.14), we see that when $I_1(t) = 1$, i.e., when P_1 is turned on, we have:

$$q_1(t) - \theta_1 + q_2(t) - \theta_2 > q_3(t) - \theta_3 \geq -3V. \tag{6.17}$$

Combining (6.17) with (6.16), we see that if $I_1(t) = 1$, we have:

$$q_j(t) \geq 2, \ j = 1, 2. \tag{6.18}$$

This is so because, e.g., if $q_1(t) \leq 1$, then $q_1(t) - \theta_1 \leq 1 - \theta_1 = -2V$. Since $q_2(t) - \theta_2 \leq -V$ by (6.16), we have:

$$q_1(t) - \theta_1 + q_2(t) - \theta_2 \leq -2V - V = -3V,$$

which cannot be greater than $-3V$ in (6.17). Thus by (6.15), (6.18), and the fact that $q_j(0) \geq 1, \forall\, j$, we have:

$$q_j(t) \geq 1, \quad j = 1, 2, 3, \forall\ t. \tag{6.19}$$

This shows that by using the θ_j values in (6.12), PMW automatically ensures that no queue underflow happens, and hence PMW minimizes the RHS of (6.8) over all possible policies. In other words, the perturbation enables us to *ignore* the no-underflow constraint (6.1) when doing scheduling, by properly modifying the weights.

Given the above observation, the utility performance of PMW can now be analyzed using a similar argument as in [NH10]. Specifically, we can first prove that there exists a stationary and randomized policy which chooses scheduling actions purely as a function

of $S(t)$, and achieves $\mathbb{E}\{\mu_j(t) - A_j(t) \mid \boldsymbol{q}(t)\} = 0$ for all j and $\mathbb{E}\{f(t) \mid \boldsymbol{q}(t)\} = f_{av}^* = \frac{1}{2}$, where f_{av}^* is the optimal average utility. Then we can compare the drift under PMW with that under this optimal policy. Note that *this analysis approach would not have been possible here without using the perturbation to ensure (6.19).* Now plugging this policy into (6.7), we obtain:

$$\Delta(t) - V\mathbb{E}\{f(t) \mid \boldsymbol{q}(t)\} \leq B - V f_{av}^*. \tag{6.20}$$

Taking expectations over $\boldsymbol{q}(t)$ on both sides and summing it over $t = 0, 1, ..., T-1$, we get:

$$\mathbb{E}\{L(T) - L(0)\} - V\sum_{t=0}^{T-1}\mathbb{E}\{f(t)\} \leq TB - VTf_{av}^*. \tag{6.21}$$

Now rearranging the terms, dividing both sides by VT, and using the fact that $L(t) \geq 0$, we get:

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{f(t)\} \geq f_{av}^* - \frac{B}{V} - \frac{\mathbb{E}\{L(0)\}}{TV}. \tag{6.22}$$

Taking a liminf as $T \to \infty$, and using $\mathbb{E}\{L(0)\} < \infty$,

$$f_{av}^{PMW} = \liminf_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{f(t)\} \geq f_{av}^* - \frac{B}{V}, \tag{6.23}$$

where f_{av}^{PMW} denotes the time average utility achieved by PMW. This shows that PMW is able to achieve a time average utility that is within $O(1/V)$ of the optimal value, and guarantees $q_j(t) \leq O(V)$ for all times (recall Equations (6.12), (6.15) and (6.16)). Note that PMW is similar to the DMW algorithm developed in [JW09]. However, DMW allows the queues to be empty when activating processors, which may lead to "deficit," whereas PMW effectively avoids this by using a perturbation vector.

In the following, we present the general processing network utility optimization model, and analyze the performance of the general PMW algorithm under this general model.

Our analysis uses a novel combination of Lyapunov drift analysis and duality theory, and is different from that in [NH10]. As we will see, our approach allows one to analyze the algorithm performance without proving the existence of an optimal stationary and randomized algorithm.

6.2 General system model

In this section, we present the general network model. We consider a network controller that operates a general network with the goal of maximizing the time average utility, subject to network stability. The network is assumed to operate in slotted time, i.e., $t \in \{0, 1, 2, ...\}$. We assume that there are $r \geq 1$ queues in the network.

6.2.1 Network state

In every slot t, we use $S(t)$ to denote the current network state, which indicates the current network parameters, such as a vector of channel conditions for each link, or a collection of other relevant information about the current network links and arrivals. We assume that $S(t)$ is i.i.d. every time slot, with a total of M different random network states denoted by $\mathcal{S} = \{s_1, s_2, \ldots, s_M\}$. [3] We let $\pi_{s_i} = \Pr\{S(t) = s_i\}$. The network controller can observe $S(t)$ at the beginning of every slot t, but the π_{s_i} probabilities are not necessarily known.

[3]Note that all our results can easily be extended to the case when $S(t)$ evolves according to a finite state aperiodic and irreducible Markov chain, by using the results developed in [HN10a].

6.2.2 The utility, traffic, and service

At each time t, after observing $S(t) = s_i$ and the network backlog vector, the controller performs an action $x(t)$. This action represents the aggregate decisions made by the controller at t, which can include (such as in the previous example), the set of processors to turn on, the amount of arriving contents to accept, etc.

We denote by $\mathcal{X}^{(s_i)}$ the set of all possible actions for network state s_i, *assuming that all the queues contain enough content to meet the scheduling requirements*. Note that we always have $x(t) = x^{(s_i)}$ for some $x^{(s_i)} \in \mathcal{X}^{(s_i)}$ whenever $S(t) = s_i$. The set $\mathcal{X}^{(s_i)}$ is assumed to be time-invariant and compact for all $s_i \in \mathcal{S}$. If the chosen action $x(t) = x^{(s_i)}$ at time t can be performed, i.e., it is possible and all the queues have enough content, then the utility, traffic, and service generated by $x(t)$ are as follows: [4]

(a) The chosen action has an associated utility given by the utility function $f(t) = f(s_i, x^{(s_i)}) : \mathcal{X}^{(s_i)} \to \mathbb{R}$;

(b) The amount of content generated by the action to queue j is determined by the traffic function $A_j(t) = A_j(s_i, x^{(s_i)}) : \mathcal{X}^{(s_i)} \to \mathbb{R}_+$, in units of content;

(c) The amount of content consumed from queue j by the action is given by the rate function $\mu_j(t) = \mu_j(s_i, x^{(s_i)}) : \mathcal{X}^{(s_i)} \to \mathbb{R}_+$, in units of content.

Note that $A_j(t)$ includes both the exogenous arrivals from outside the network to queue j, and the endogenous arrivals from other queues, i.e., the newly generated contents by processing contents in some other queues, to queue j. We assume that the functions

[4] Here we implicitly assume that each action takes only one unit time. We further discuss this issue in Section 6.2.4.

$f(s_i, \cdot)$, $\mu_j(s_i, \cdot)$ and $A_j(s_i, \cdot)$ are continuous, time-invariant, their magnitudes are uniformly upper bounded by some constant $\delta_{max} \in (0, \infty)$ for all s_i, j, and they are known to the network operator.

In any actual algorithm implementation, however, we see that not all actions in the set $\mathcal{X}^{(s_i)}$ can be performed when $S(t) = s_i$, due to the fact that some queues may not have enough contents for the action. We say that an action $x^{(s_i)} \in \mathcal{X}^{(s_i)}$ is feasible at time t with $S(t) = s_i$ only when the following general *no-underflow constraint* is satisfied:

$$q_j(t) \geq \mu_j(s_i, x^{(s_i)}), \ \forall \ j. \tag{6.24}$$

That is, *all the queues must have contents greater than or equal to what will be consumed.* In the following, we assume that there exists a set of actions $\{x_k^{(s_i)}\}_{i=1,\ldots,M}^{k=1,2,\ldots,r+2}$ with $x_k^{(s_i)} \in \mathcal{X}^{(s_i)}$ and some variables $\vartheta_k^{(s_i)} \geq 0$ for all s_i and k with $\sum_{k=1}^{r+2} \vartheta_k^{(s_i)} = 1$ for all s_i, such that:

$$\sum_{s_i} \pi_{s_i} \Big\{ \sum_{k=1}^{r+2} \vartheta_k^{(s_i)} [A_j(s_i, x_k^{(s_i)}) - \mu_j(s_i, x_k^{(s_i)})] \Big\} \leq -\eta, \tag{6.25}$$

for some $\eta > 0$ for all j. That is, the "stability constraints" are feasible with η-slack. [5] The slackness condition here is slightly different from the one assumed in Chapter 4. There, we consider communication networks and the actions correspond to packet transmissions/receptions. Thus, all the actions are feasible even when the queues are empty. Hence the slackness condition translates into the existence of a stationary and randomized policy that stabilizes the network. Here, the condition ignores the *feasibility* constraint (6.24) of the network actions, i.e., when they are performed, they need enough

[5] The use of $r + 2$ actions here is due to the use of Caratheodory's theorem [BNO03] in the proof of Theorem 23.

content in the queues. Thus, it does not immediately imply the existence of such a feasible stationary randomized policy.

6.2.3 Queueing, average cost, and the objective

Let $\boldsymbol{q}(t) = (q_1(t), ..., q_r(t))^T \in \mathbb{R}_+^r$, $t = 0, 1, 2, ...$ be the queue backlog vector process of the network, in units of contents. Due to the feasibility condition (6.24) of the actions, we see that the queues evolve according to the following dynamics:

$$q_j(t+1) = q_j(t) - \mu_j(t) + A_j(t), \quad \forall j,\, t \geq 0, \tag{6.26}$$

with some $||\boldsymbol{q}(0)|| < \infty$. Note that using a nonzero $q_j(0)$ can be viewed as placing an "initial stock" in the queues to facilitate algorithm implementation. In this chapter, we adopt the following notion of queue stability:

$$\overline{q} \triangleq \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{j=1}^{r} \mathbb{E}\big\{q_j(\tau)\big\} < \infty. \tag{6.27}$$

We also use f_{av}^{Π} to denote the time average utility induced by an action-choosing policy Π, defined as:

$$f_{av}^{\Pi} \triangleq \liminf_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\big\{f^{\Pi}(\tau)\big\}, \tag{6.28}$$

where $f^{\Pi}(\tau)$ is the utility accrued at time τ by policy Π. We call an action-choosing policy *feasible* if at every time slot t it only chooses actions from the possible action set $\mathcal{X}^{(S(t))}$ that satisfy (6.24). We then call a feasible action-choosing policy under which (6.27) holds a *stable* policy, and use f_{av}^{*} to denote the optimal time average utility over all stable policies.

In every slot, the network controller observes the current network state and the queue backlog vector, and chooses a feasible control action that ensures (6.24), with the objective

of maximizing the time average utility subject to network stability. Note that if (6.24) can be ignored, and if any processor only requires contents from a single queue, then this problem falls into the general stochastic network optimization framework considered in [GNT06], in which case it can be solved by using the usual Max-Weight algorithm to achieve a utility that is within $O(1/V)$ of the optimal while ensuring that the average network backlog is $O(V)$.

6.2.4 Discussion of the model

We note that the model is very general and can be used to model many problems that involve such no-underflow constraints. For instance, manufacturing networks where parts are assembled into products, or energy harvesting networks that are powered by finite capacity energy storage devices.

Our model assumes that the network operates in slotted time. This implicitly implies that all the network actions must consume only one unit time. However, in many applications, e.g., stream processing, different processors may take different amounts of time to process different data. Although this problem can easily be incorporated into our network by using a large slot size, such an easy fix may lead to utility loss. Quantifying the utility loss due to slot length selection and designing algorithms that allow for arbitrary heterogeneous action times are interesting problems of our future research.

6.3 Upper bounding the optimal utility

In this section, we first obtain an upper bound on the optimal utility that the network controller can achieve. This upper bound will later be used to analyze the performance of our algorithm. The result is summarized in the following theorem.

Theorem 23. *Suppose that the initial queue backlog* $\boldsymbol{q}(t)$ *satisfies* $\mathbb{E}\{q_j(0)\} < \infty$ *for all* $j = 1, ..., r$. *Then we have:*

$$Vf_{av}^* \leq \phi^*, \tag{6.29}$$

where ϕ^* *is the optimal value of the following* deterministic optimization *problem:*

$$\max: \quad \phi = \sum_{s_i} \pi_{s_i} V \sum_{k=1}^{r+2} a_k^{(s_i)} f(s_i, x_k^{(s_i)}) \tag{6.30}$$

$$s.t. \quad \sum_{s_i} \pi_{s_i} \sum_{k=1}^{r+2} a_k^{(s_i)} A_j(s_i, x_k^{(s_i)}) = \sum_{s_i} \pi_{s_i} \sum_{k=1}^{r+2} a_k^{(s_i)} \mu_j(s_i, x_k^{(s_i)}), \tag{6.31}$$

$$x_k^{(s_i)} \in \mathcal{X}^{(s_i)}, \forall\, s_i, k, \tag{6.32}$$

$$a_k^{(s_i)} \geq 0, \forall\, s_i, k, \sum_k a_k^{(s_i)} = 1, \forall\, s_i. \tag{6.33}$$

Proof. See Section 6.10.1. □

Note that the problem (6.30) only requires that the time average input rate into a queue is equal to its time average output rate. This requirement ignores the action feasibility constraint (6.24), and makes (6.30) easier to solve than the actual scheduling problem. We now look at the dual problem of the problem (6.30). The following lemma shows that the dual problem of (6.30) does not have to include the variables $\{a_k^{(s_i)}\}_{i=1,...,M}^{k=1,...,r+2}$. This lemma will also be useful for our later analysis.

Lemma 8. *The dual problem of (6.30) is given by:*

$$\min: \quad g(\boldsymbol{\gamma}), \;\; s.t. \quad \boldsymbol{\gamma} \in \mathbb{R}^r, \tag{6.34}$$

where the function $g(\boldsymbol{\gamma})$ is defined as follows:

$$g(\boldsymbol{\gamma}) = \sup_{x^{(s_i)} \in \mathcal{X}^{(s_i)}} \sum_{s_i} \pi_{s_i} \left\{ Vf(s_i, x^{(s_i)}) - \sum_j \gamma_j \big[A_j(s_i, x^{(s_i)}) - \mu_j(s_i, x^{(s_i)}) \big] \right\}. \tag{6.35}$$

Moreover, letting $\boldsymbol{\gamma}^*$ be any optimal solution of (6.34), we have $g(\boldsymbol{\gamma}^*) \geq \phi^*$.

Proof. See Section 6.10.2. □

For our later analysis, it is useful to define the following function:

$$g_{s_i}(\boldsymbol{\gamma}) = \sup_{x^{(s_i)} \in \mathcal{X}^{(s_i)}} \left\{ Vf(s_i, x^{(s_i)}) - \sum_j \gamma_j \big[A_j(s_i, x^{(s_i)}) - \mu_j(s_i, x^{(s_i)}) \big] \right\}. \tag{6.36}$$

That is, $g_{s_i}(\boldsymbol{\gamma})$ is the dual function of (6.30) when there is a single network state s_i. We can see from (6.35) and (6.36) that:

$$g(\boldsymbol{\gamma}) = \sum_{s_i} \pi_{s_i} g_{s_i}(\boldsymbol{\gamma}). \tag{6.37}$$

In the following, we use $\boldsymbol{\gamma}^* = (\gamma_1^*, ..., \gamma_r^*)^T$ to denote an optimal solution of the problem (6.34).

6.4 The general perturbed Max-Weight algorithm and its performance

In this section, we develop the general Perturbed Max-Weight algorithm (PMW) to solve our scheduling problem. To start, we first choose a *perturbation vector* $\boldsymbol{\theta} = (\theta_1, ..., \theta_r)^T$. Then we define the following *weighted perturbed* Lyapunov function with some positive constants $\{w_j\}_{j=1}^r$:

$$L(t) = \frac{1}{2} \sum_{j=1}^{r} w_j \big(q_j(t) - \theta_j \big)^2. \tag{6.38}$$

We then define the one-slot conditional drift as in (6.7), i.e., $\Delta(t) = \mathbb{E}\big\{L(t+1) - L(t) \mid \boldsymbol{q}(t)\big\}$. We similarly use the "drift-plus-penalty" approach in Section 6.1 to construct the algorithm. Specifically, we first use the queueing dynamic equation (6.26), and have the following lemma:

Lemma 9. *Under any feasible control policy that can be implemented at time t, we have:*

$$\begin{aligned}\Delta(t) - V\mathbb{E}\big\{f(t) \mid \boldsymbol{q}(t)\big\} \quad &\leq \quad B - V\mathbb{E}\big\{f(t) \mid \boldsymbol{q}(t)\big\} \\ &\quad - \sum_{j=1}^{r} w_j\big(q_j(t) - \theta_j\big)\mathbb{E}\big\{[\mu_j(t) - A_j(t)] \mid \boldsymbol{q}(t)\big\},\end{aligned} \tag{6.39}$$

where $B = \delta_{max}^2 \sum_{j=1}^{r} w_j$.

Proof. See Section 6.10.3. □

The general Perturbed Max-Weight algorithm (PMW) is then obtained by choosing an action $x(t) \in \mathcal{X}^{(S(t))}$ at time t to *minimize the right-hand side (RHS) of (6.39)* subject to (6.24). Specifically, define the function $D^{(s_i)}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x)$ as:

$$D^{(s_i)}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x) \triangleq Vf(s_i,x) + \sum_{j=1}^{r} w_j\big(q_j(t) - \theta_j\big)\big[\mu_j(s_i,x) - A_j(s_i,x)\big]. \tag{6.40}$$

We see that the function $D^{(s_i)}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x)$ is indeed the term inside the conditional expectation on the RHS of (6.39) when $S(t) = s_i$. We now also define $D^{(s_i)*}_{\boldsymbol{\theta},\boldsymbol{q}(t)}$ to be the optimal value of the following problem:

$$\max: \ D^{(s_i)}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x), \quad \text{s.t.,} \ \ x^{(s_i)} \in \mathcal{X}^{(s_i)}. \tag{6.41}$$

Hence $D^{(s_i)*}_{\boldsymbol{\theta},\boldsymbol{q}(t)}$ is the maximum value of $D^{(s_i)}_{\boldsymbol{\theta},\boldsymbol{q}(t)}$ over all possible policies, including those that may not consider the no-underflow constraint (6.24). The general Perturbed Max-Weight algorithm (PMW) then works as follows:

PMW: Initialize the perturbation vector $\boldsymbol{\theta}$. At every time slot t, observe the current network state $S(t)$ and the backlog $\boldsymbol{q}(t)$. If $S(t) = s_i$, choose $x^{(s_i)} \in \mathcal{X}^{(s_i)}$ subject to (6.24) that minimizes $D^{(s_i)}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x)$.

Note that depending on the problem structure, the PMW algorithm can usually be implemented easily, e.g., [NH10], [Nee06c]. Now we analyze the performance of the PMW algorithm. We prove our result under the following condition:

Condition 1. *There exists some finite constant $C \geq 0$, such that at every time slot t with a network state $S(t)$, the value of $D^{(S(t))}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x)$ under PMW is at least $D^{(S(t))*}_{\boldsymbol{\theta},\boldsymbol{q}(t)} - C$.*

The immediate consequence of Condition 1 is that PMW also minimizes the RHS of (6.39), i.e., the conditional expectation, to within C of its minimum value over all possible policies. If $C = 0$, then PMW simultaneously ensures (6.24) and minimizes the RHS of (6.39), e.g., as in the example in Section 6.1. However, we note that Condition 1 does not require the value of $D^{(S(t))}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x)$ to be exactly the same as $D^{(S(t))*}_{\boldsymbol{\theta},\boldsymbol{q}(t)}$. This allows for more flexibility in constructing the PMW algorithm. (See Section 6.6 for an example.) We also note that Condition 1 can be ensured, e.g., by carefully choosing the θ_j values to ensure $q_j(t) \geq \delta_{max}$ for all times [NH10]. We show that, under Condition 1, PMW achieves a time average utility that is within $O(1/V)$ of f^*_{av}, while guaranteeing that the time average network queue size is $O(V) + \sum_j w_j\theta_j$, which is $O(V)$ if $\boldsymbol{\theta} = \Theta(V)$ and $w_j = O(1)$, $\forall\, j$. The following theorem summarizes PMW's performance results.

Theorem 24. *Suppose that (6.25) holds, that Condition 1 holds, and that* $\mathbb{E}\{q_j(0)\} < \infty$ *for all* $j = 1, ..., r$. *Then under PMW, we have:* [6]

$$f_{av}^{PMW} \geq f_{av}^{*} - \frac{B+C}{V}, \tag{6.42}$$

$$\overline{q}^{PMW} \leq \frac{B+C+2V\delta_{max}}{\eta} + \sum_{j=1}^{r} w_j\theta_j. \tag{6.43}$$

Here $B = \delta_{max}^2 \sum_{j=1}^{r} w_j$, η *is the slackness parameter in Section 6.2.2,* f_{av}^{PMW} *is defined in (6.28) to be the time average expected utility of PMW, and* $\overline{q}^{PMW}$ *is the time average expected weighted network backlog under PMW, defined:*

$$\overline{q}^{PMW} \triangleq \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{j=1}^{r} w_j \mathbb{E}\{q_j(\tau)\}.$$

Proof. See Section 6.10.4. □

Theorem 24 shows that if Condition 1 holds, then PMW can be used as in previous networking problems, e.g., [Nee06c], [HN10c], to obtain explicit utility-backlog tradeoffs. We note that a condition similar to Condition 1 was assumed in [DL05]. However, [DL05] only considers the usual Max-Weight algorithm, in which case (6.24) may not be satisfied for all time slots. PMW instead resolves this problem by carefully choosing the perturbation vector. One such example of PMW is the recent work [NH10], which applies PMW to an assembly line scheduling problem and achieves an $[O(1/V), O(V)]$ utility-backlog tradeoff.

We emphasize that though the results in Theorem 24 look similar to those in the data network problems [GNT06], the proof techniques are very different. Our analysis uses a novel combination of Lyapunov drift analysis and duality theory, and allows one to obtain the performance result without proving the existence of an optimal stationary

[6] We see that (6.43) ensures (6.27), hence the network is stable under PMW.

and randomized policy. Also, though the no-underflow constraints make the problems in this chapter very different from the communication network problems, the methodology developed in Chapter 4 for obtaining improved tradeoffs can still be applied to such problems. Indeed, in Chapter 7, we develop a novel algorithm for achieving the $[O(1/V), O([\log(V)]^2)]$ utility-buffer tradeoff for energy harvesting networks, which also involve such no-underflow constraints.

6.5 Discussion of finding the perturbation value

In the PMW algorithm, we have assumed that the value of the perturbation vector $\boldsymbol{\theta}$ can be chosen to ensure Condition 1. However, developing a systematic approach for finding such a perturbation value is still an open problem. In the following section, we show how to find the $\boldsymbol{\theta}$ value for a general class of processing networks. This section highlights the fact that although the value of $\boldsymbol{\theta}$ is application-specific, we can still find its value efficiently in many cases. More discussions regarding the role of $\boldsymbol{\theta}$ can be found in Section 6.8.

6.6 Constructing PMW for stochastic processing networks with output reward

In this section, we look at a specific yet general processing network model, and explicitly construct a PMW algorithm, including finding the proper $\boldsymbol{\theta}$ vector and choosing actions at each time slot.

6.6.1 Network model

We assume that the network is modeled by an acyclic directed graph $\mathcal{G} = (\mathcal{Q}, \mathcal{P}, \mathcal{L})$. Here $\mathcal{Q} = \mathcal{Q}^s \cup \mathcal{Q}^{in}$ is the set of queues, consisting of the set of *source* queues $\mathcal{Q}^s$ where arrivals enter the network, and the set of *internal* queues $\mathcal{Q}^{in}$ where content are stored for further processing. $\mathcal{P} = \mathcal{P}^{in} \cup \mathcal{P}^o$ is the set of processors, consisting of a set of *internal* processors $\mathcal{P}^{in}$, which generate partially processed contents for further processing at other processors, and *output* processors $\mathcal{P}^o$, which generate fully processed contents and deliver them to the output. $\mathcal{L}$ is the set of directed links that connects $\mathcal{Q}$ and $\mathcal{P}$. Note that a link only exists between a queue in $\mathcal{Q}$ and a processor in $\mathcal{P}$. We denote $N_p^{in} = |\mathcal{P}^{in}|$, $N_p^o = |\mathcal{P}^o|$ and $N_p = N_p^{in} + N_p^o$. We also denote $N_q^s = |\mathcal{Q}^s|$, $N_q^{in} = |\mathcal{Q}^{in}|$ and $N_q = N_q^s + N_q^{in}$.

Each processor P_n, when activated, consumes a certain amount of content from a set of *supply* queues, denoted by $\mathbb{Q}_n^S$, and generates some amount of new contents. These new contents either go to a set of *demand* queues, denoted by $\mathbb{Q}_n^D$, if $P_n \in \mathcal{P}^{in}$, or are delivered to the output if $P_n \in \mathcal{P}^o$. For any queue $q_j \in \mathcal{Q}$, we use $\mathbb{P}_j^S$ to denote the set of processors that q_j serves as a supply queue, and use $\mathbb{P}_j^D$ to denote the set of processors that q_j serves as a demand queue. An example of such a network is shown in Fig. 6.2. In the following, we assume that $|\mathbb{Q}_i^D| = 1, \forall\ P_i \in \mathcal{P}^{in}$, i.e., each processor only generates contents for a single demand queue.

We use β_{nj} to denote the amount processor P_n consumes from a queue q_j in $\mathbb{Q}_n^S$ when it is activated. For each $P_i \in \mathcal{P}^{in}$, we also use α_{ih} to denote the amount P_i generates into the queue q_h if $q_h = \mathbb{Q}_i^D$, when it is activated. For a processor $P_k \in \mathcal{P}^o$, we use

α_{ko} to denote the amount of output generated by it when it is turned on. [7] We denote $\beta_{max} = \max_{i,j} \beta_{ij}$, $\beta_{min} = \min_{i,j} \beta_{ij}$ and $\alpha_{max} = \max_{i,j,}[\alpha_{ij}, \alpha_{io}]$. We assume that $\beta_{min}, \beta_{max}, \alpha_{max} > 0$. We also define M_p to be the maximum number of supply queues that any processor can have, define M_q^d to be the maximum number of processors that any queue can serve as a demand queue for, and define M_q^s to be the maximum number of processors that any queue can serve as a supply queue for. We use $R_j(t)$ to denote the amount of content arriving at a source queue $q_j \in \mathcal{Q}^s$ at time t. We assume that $R_j(t)$ is i.i.d. every slot, and that $R_j(t) \leq R_{max}$ for all $q_j \in \mathcal{Q}^s$ and all t. We assume that there are no exogenous arrivals at the queues in $\mathcal{Q}^{in}$.

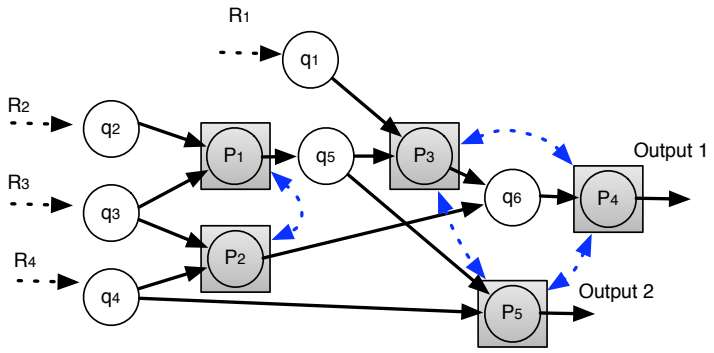

Figure 6.2: A general processing network. A dotted line between two processors means that the processors share some common resources and thus cannot be activated at the same time.

We assume that in every slot t, admitting any unit amount of $R_j(t)$ arrival incurs a cost of $c_j(t)$, and that activating any internal processor $P_i \in \mathcal{P}^{in}$ incurs a cost of $C_i(t)$, whereas activating any output processor $P_k \in \mathcal{P}^o$ generates a profit of $p_k(t)$ per unit output content. [8] We assume that $c_j(t), C_i(t), p_k(t)$ are all i.i.d. every time slot. In the

[7] Note that here we only consider binary actions of processors, i.e., ON/OFF. Our results can also be generalized to the case when there are multiple operation levels under which different amounts of content are consumed and generated.

[8] This can be viewed as the difference between profit and cost associated with these processors.

following, we also assume that $p_{min} \leq p_k(t) \leq p_{max}$, and that $c_{min} \leq c_j(t) \leq c_{max}$ and $C_{min} \leq C_i(t) \leq C_{max}$ for all k, j, i and for all time.

Below, we use $I_n(t) = 1$ to denote the activation decision of P_n, i.e., $I_n(t) = 1$ ($I_n(t) = 0$) means that P_n is activated (turned off). We also use $D_j(t) \in [0, 1]$ to denote the portion of arrivals from $R_j(t)$ that are admitted into q_j. We assume that there exist some general constraints on how the processors can be activated, which can be due to, e.g., resource sharing among processors. We model these constraints by defining an activation vector $\boldsymbol{I}(t) = (I_1(t), ..., I_{N_p}(t))$, and then assume that $\boldsymbol{I}(t) \in \mathcal{I}$ for all times, where $\mathcal{I}$ denotes the set of all possible processor activation decision vectors, assuming all the queues have enough contents for processing. We assume that if a vector $\boldsymbol{I} \in \mathcal{I}$, then by changing one element of $\boldsymbol{I}$ from one to zero, the newly obtained vector $\boldsymbol{I}'$ satisfies $\boldsymbol{I}' \in \mathcal{I}$. Note that the chosen vector $\boldsymbol{I}(t)$ must always ensure the constraint (6.24), which in this case implies that $\boldsymbol{I}(t)$ has to satisfy the following constraint:

$$q_j(t) \geq \sum_{n \in \mathbb{P}_j^S} I_n(t)\beta_{nj}, \quad \forall\ j = 1, ..., r. \tag{6.44}$$

Under this constraint, we see that the queues evolve according to the following queueing dynamics:

$$q_j(t+1) = q_j(t) - \sum_{n \in \mathbb{P}_j^S} I_n(t)\beta_{nj} + D_j(t)R_j(t),\ \forall j \in \mathcal{Q}^s,$$
$$q_j(t+1) = q_j(t) - \sum_{n \in \mathbb{P}_j^S} I_n(t)\beta_{nj} + \sum_{n \in \mathbb{P}_j^D} I_n(t)\alpha_{nj},\ \forall j \in \mathcal{Q}^{in}.$$

Note that we have used $j \in \mathcal{Q}$ to represent $q_j \in \mathcal{Q}$, and use $n \in \mathcal{P}$ to represent $P_n \in \mathcal{P}$ in the above for notation simplicity. The objective is to maximize the time average of the following utility function:

$$f(t) \triangleq \sum_{k \in \mathcal{P}^o} I_k(t)p_k(t)\alpha_{ko} - \sum_{j \in \mathcal{Q}^s} D_j(t)R_j(t)c_j(t) - \sum_{i \in \mathcal{P}^{in}} I_i(t)C_i(t). \tag{6.45}$$

Our model with the objective function (6.45) can be used to model applications where generating completely processed contents is the primary target, e.g., [NH10].

6.6.2 Relation to the general model

We see that in this network, the network state, the action, and the traffic and service functions are as follows:

- The network state is given by: $S(t) = (c_j(t), j \in \mathcal{Q}^s, C_i(t), i \in \mathcal{P}^{in}, p_k(t), k \in \mathcal{P}^o)$.
- The action $x(t) = (D_j(t), j \in \mathcal{Q}^s, I_n(t), n \in \mathcal{P})$.
- The arrival functions are given by: $A_j(t) = A_j(S(t), x(t)) = D_j(t)R_j(t), \ \forall\, q_j \in \mathcal{Q}^s$, and $A_j(t) = A_j(S(t), x(t)) = \sum_{n \in \mathbb{P}_j^D} I_n(t)\alpha_{nj}, \ \forall\, q_j \in \mathcal{Q}^{in}$.
- The service functions are given by: $\mu_j(t) = \mu_j(S(t), x(t)) = \sum_{n \in \mathbb{P}_j^S} I_n(t)\beta_{nj}, \ \forall\ j$.

Thus, we see that this network model falls into the general processing network framework in Section 6.2, and Theorem 24 applies in this case. Therefore, to ensure the algorithm performance, we only have to construct our PMW algorithm to ensure that Condition 1 holds. Also note that in this case, we have:

$$\delta_{max} = \max\left[\nu_{max}, N_p^o p_{max}\alpha_{max}, N_q^s R_{max} c_{max} + N_p^{in} C_{max}\right]. \tag{6.46}$$

Here ν_{max} is defined:

$$\nu_{max} \triangleq \max\left[M_q^d \alpha_{max}, R_{max}, M_q^s \beta_{max}\right]. \tag{6.47}$$

6.6.3 The PMW algorithm

We now obtain the PMW algorithm for this general network. In this case, we look for a perturbation vector that is the same in all entries, i.e., $\boldsymbol{\theta} = \theta\mathbf{1}$. We first compute the

"drift-plus-penalty" expression using the weighted perturbed Lyapunov function defined in (6.38) under some given positive constants $\{w_j\}_{j=1}^r$ and some nonzero constant θ:

$$\begin{aligned}\Delta(t) - V\mathbb{E}\big\{f(t) \mid \boldsymbol{q}(t)\big\} \qquad (6.48)\\ \leq B - \sum_{j\in\mathcal{Q}^s} \mathbb{E}\big\{w_j\big[q_j(t)-\theta\big]\big[\sum_{n\in\mathbb{P}_j^S} I_n(t)\beta_{nj} - R_j(t)D_j(t)\big] \mid \boldsymbol{q}(t)\big\}\\ - \sum_{j\in\mathcal{Q}^{in}} \mathbb{E}\big\{w_j\big[q_j(t)-\theta\big]\big[\sum_{n\in\mathbb{P}_j^S} I_n(t)\beta_{nj} - \sum_{n\in\mathbb{P}_j^D} I_n(t)\alpha_{nj}\big] \mid \boldsymbol{q}(t)\big\}\\ -V\mathbb{E}\big\{\sum_{k\in\mathcal{P}^o} I_k(t)p_k(t)\alpha_{ko} - \sum_{j\in\mathcal{Q}^s} D_j(t)R_j(t)c_j(t) - \sum_{i\in\mathcal{P}^{in}} I_i(t)C_i(t) \mid \boldsymbol{q}(t)\big\}.\end{aligned}$$

Here $B = \delta_{max}^2 \sum_j w_j$ with δ_{max} defined in (6.46). Rearranging the terms in (6.48), we get the following:

$$\begin{aligned}\Delta(t) - V\mathbb{E}\big\{f(t) \mid \boldsymbol{q}(t)\big\} \qquad (6.49)\\ \leq B + \sum_{j\in\mathcal{Q}^s} \mathbb{E}\big\{\big[Vc_j(t) + w_j(q_j(t)-\theta)\big]D_j(t)R_j(t) \mid \boldsymbol{q}(t)\big\}\\ - \sum_{k\in\mathcal{P}^o} \mathbb{E}\big\{I_k(t)\big[\sum_{j\in\mathbb{Q}_k^S} w_j(q_j(t)-\theta)\beta_{kj} + Vp_k(t)\alpha_{ko}\big] \mid \boldsymbol{q}(t)\big\}\\ - \sum_{i\in\mathcal{P}^{in}} \mathbb{E}\big\{I_i(t)\big[\sum_{j\in\mathbb{Q}_i^S} w_j(q_j(t)-\theta)\beta_{ij} - w_h(q_h(t)-\theta)\alpha_{ih} - VC_i(t)\big] \mid \boldsymbol{q}(t)\big\}.\end{aligned}$$

Here in the last term $q_h = \mathbb{Q}_i^D$. We now present the PMW algorithm. We see that in this case the $D_{\boldsymbol{\theta},\boldsymbol{q}(t)}^{(S(t))}(x)$ function is given by:

$$\begin{aligned}D_{\boldsymbol{\theta},\boldsymbol{q}(t)}^{(S(t))}(x) = & -\sum_{j\in\mathcal{Q}^s}\big[Vc_j(t) + w_j(q_j(t)-\theta)\big]D_j(t)R_j(t) \qquad (6.50)\\ & + \sum_{k\in\mathcal{P}^o} I_k(t)\big[\sum_{j\in\mathbb{Q}_k^S} w_j(q_j(t)-\theta)\beta_{kj} + Vp_k(t)\alpha_{ko}\big]\\ & + \sum_{i\in\mathcal{P}^{in}} I_i(t)\big[\sum_{j\in\mathbb{Q}_i^S} w_j(q_j(t)-\theta)\beta_{ij} - w_h(q_h(t)-\theta)\alpha_{ih} - VC_i(t)\big].\end{aligned}$$

Our goal is to design PMW in a way such that under any network state $S(t)$, the value of $D^{(S(t))}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x)$ is close to $D^{(S(t))*}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x)$, which is the maximum value of $D^{(S(t))}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x)$ without the no-underflow constraint (6.44), i.e.,

$$D^{(S(t))*}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x) = \max_{D_j(t)\in[0,1],\boldsymbol{I}(t)\in\mathcal{I}} D^{(S(t))}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x).$$

Specifically, PMW works as follows:

PMW: Initialize $\boldsymbol{\theta}$ (to be specified in (6.55)). At every time slot t, observe $S(t)$ and $\boldsymbol{q}(t)$, and do the following:

1. Content Admission: Choose $D_j(t) = 1$, i.e., admit all new arrivals to $q_j \in \mathcal{Q}^s$ if:

$$Vc_j(t) + w_j(q_j(t) - \theta) < 0, \tag{6.51}$$

   else set $D_j(t) = 0$.

2. Processor Activation: For each $P_i \in \mathcal{P}^{in}$, define its weight $W_i^{(in)}(t)$ as:

$$W_i^{(in)}(t) = \big[\sum_{q_j\in\mathbb{Q}_i^S} w_j[q_j(t) - \theta]\beta_{ij} - w_h[q_h(t) - \theta]\alpha_{ih} - VC_i(t)\big]^+, \tag{6.52}$$

   where $q_h = \mathbb{Q}_i^D$. Similarly, for each $P_k \in \mathcal{P}^o$, define its weight $W_k^{(o)}(t)$ as:

$$W_k^{(o)}(t) = \big[\sum_{q_j\in\mathbb{Q}_k^S} w_j[q_j(t) - \theta]\beta_{kj} + Vp_k(t)\alpha_{ko}\big]^+. \tag{6.53}$$

   Then, choose an activation vector $\boldsymbol{I}(t) \in \mathcal{I}$ to maximize:

$$W(t) \triangleq \sum_{i\in\mathcal{P}^{in}} I_i(t)W_i^{(in)}(t) + \sum_{k\in\mathcal{P}^o} I_k(t)W_k^{(o)}(t), \tag{6.54}$$

   subject to the following *queue edge constraints:*

   (a) For each $P_i \in \mathcal{P}^{in}$, set $I_i(t) = 1$, i.e., activate processor P_i, only if:

   - $q_j(t) \geq M_q^s\beta_{max}$ for all $q_j \in \mathbb{Q}_i^S$ & $q_h(t) \leq \theta$, where $q_h = \mathbb{Q}_i^D$.

   (b) For each $P_k \in \mathcal{P}^o$, choose $I_k(t) = 1$ only if:

   - $q_j(t) \geq M_q^s\beta_{max}$ for all $q_j \in \mathbb{Q}_k^S$.

The approach of imposing the queue edge constraints was inspired by the work [Nee10a], where similar constraints are imposed for routing problems. Note that without these queue edge constraints, PMW is the same as the action that maximizes $D^{(s_i)}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x)$ without the no-underflow constraint (6.44).

6.6.4 Performance

Here we show that PMW indeed ensures that the value of $D^{(S(t))}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x)$ is within some additive constant of $D^{(S(t))*}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x)$. In the following, we denote $w_{max} = \max_j w_j$ and $w_{min} = \min_j w_j$. We also assume that:

$$\theta \geq \max\big[\frac{V\alpha_{max}p_{max}}{w_{min}\beta_{min}}, \frac{Vc_{min}}{w_{min}} + M_q^s\beta_{max}\big]. \tag{6.55}$$

We also assume that the $\{w_j\}_{j=1}^r$ values are chosen such that for any processor $P_i \in \mathcal{P}^{in}$ with demand queue q_h, we have for any supply queue $q_j \in \mathbb{Q}_i^S$ that:

$$w_j\beta_{ij} \geq w_h\alpha_{ih}. \tag{6.56}$$

As we will see in the proof of Lemma 10, these w_j values are chosen to resolve the technical issues caused by the different content generation and consumption rates. We note that (6.55) can easily be satisfied and only requires $\theta = \Theta(V)$. A way of choosing the $\{w_j\}_{j=1}^r$ values to satisfy (6.56) is given in Section 6.10.5. Note that in the special case when $\beta_{ij} = \alpha_{ij} = 1$ for all i, j, simply using $w_j = 1, \forall\, j$ meets the condition (6.56).

We first look at the queueing bounds. By (6.51), q_j admits new arrivals only when $q_j(t) < \theta - Vc_{min}/w_j$. Thus:

$$q_j(t) \leq \theta - Vc_{min}/w_j + R_{max}, \quad \forall\ q_j \in \mathcal{Q}^s, t. \tag{6.57}$$

Now by the processor activation rule, we also see that:

$$0 \leq q_j(t) \leq \theta + M_q^d \alpha_{max}, \quad \forall\ q_j \in \mathcal{Q}^{in}, t. \tag{6.58}$$

This is because under the PMW algorithm, a processor is activated only when all its supply queues have at least $M_q^s \beta_{max}$ units of contents, and when its demand queue has at most θ units of contents. The first requirement ensures that when we activate a processor, all its supply queues have enough contents, while the second requirement ensures that $q_j(t) \leq \theta + M_q^d \alpha_{max}$. Using the definition of ν_{max} in (6.47), we can compactly write (6.57) and (6.58) as:

$$0 \leq q_j(t) \leq \theta + \nu_{max}, \quad \forall\ q_j \in \mathcal{Q}, t. \tag{6.59}$$

Recall that by the discussion in Section 6.6.2, the problem described in this section falls into the general framework presented in Section 6.2. Hence, to prove the performance of the PMW algorithm, it suffices to prove the following lemma, which shows that Condition 1 holds for some finite constant C under PMW.

Lemma 10. *Suppose that (6.55) and (6.56) hold. Then under PMW,* $D_{\boldsymbol{\theta},\boldsymbol{q}(t)}^{(S(t))}(x) \geq D_{\boldsymbol{\theta},\boldsymbol{q}(t)}^{(S(t))*}(x) - C$*, where* $C = N_p w_{max} M_p \nu_{max} \beta_{max}$.

Proof. See Section 6.10.6. □

We now use Theorem 24 to have the following corollary concerning the performance of PMW in this case:

Corollary 3. *Suppose that (6.25), (6.55) and (6.56) hold. Then under PMW, (6.59) holds, and that:*

$$f_{av}^{PMW} \quad \geq \quad f_{av}^* - \frac{B+C}{V}, \tag{6.60}$$

$$\overline{q}^{PMW} \leq \frac{B + C + 2V\delta_{max}}{\eta} + \theta \sum_{j=1}^{r} w_j, \tag{6.61}$$

where $C = N_p w_{max} M_p \nu_{max} \beta_{max}$, f_{av}^{PMW} *and* $\overline{q}^{PMW}$ *are the time average expected utility and time average expected weighted backlog under PMW, respectively, and* δ_{max} *is given in (6.46).* ■

Also, since (6.55) only requires $\theta = \Theta(V)$, and $w_j = \Theta(1)$ for all j (shown in Section 6.10.5), we see that PMW achieves an $[O(1/V), O(V)]$ utility-backlog tradeoff in this case.

6.7 Simulation

In this section, we simulate the example given in Fig. 6.2. In this example, we assume that each $R_j(t)$ is Bernoulli being 0 or 2 with equal probability. For each $P_i \in \mathcal{P}^{in}$, i.e., P_1, P_2, P_3, $C_i(t)$ is assumed to be 1 or 10 with probabilities 0.3 and 0.7, respectively. For the output processors $P_k \in \mathcal{P}^o$, i.e., P_4 and P_5, we assume that $p_k(t) = 1$ or 5 with probabilities 0.8 and 0.2, respectively. We assume that each processor, when activated, takes one unit of content from each of its supply queues and generates two units of contents into its demand queue (or to the output if it is an output processor). Due to the activation constraints, P_1 and P_2 can not be activated at the same time. Also, only one among P_3, P_4, P_5 can be turned on at any time. Note that in this case, we have $c_j(t) = 0$ for all source queues q_j. It can be seen that in this case $M_p = M_q^s = M_q^d = 2$, $\beta_{max} = \beta_{min} = 1$, and $\alpha_{max} = 2$. Using the results in Section 6.10.5, we choose $w_6 = 1$, $w_1 = w_4 = w_5 = 2$, $w_2 = w_3 = 4$. We also use $\theta = 10V$ according to (6.55). We simulate the PMW algorithm for $V \in \{5, 7, 10, 15, 20, 50, 100\}$. Each simulation is run over 5×10^6 slots.

Fig. 6.3 shows the utility and backlog performance of the PMW algorithm. We see that as V increases, the average utility performance quickly converges to the optimal value. The average backlog size also only grows linearly in V. Fig. 6.4 also shows three

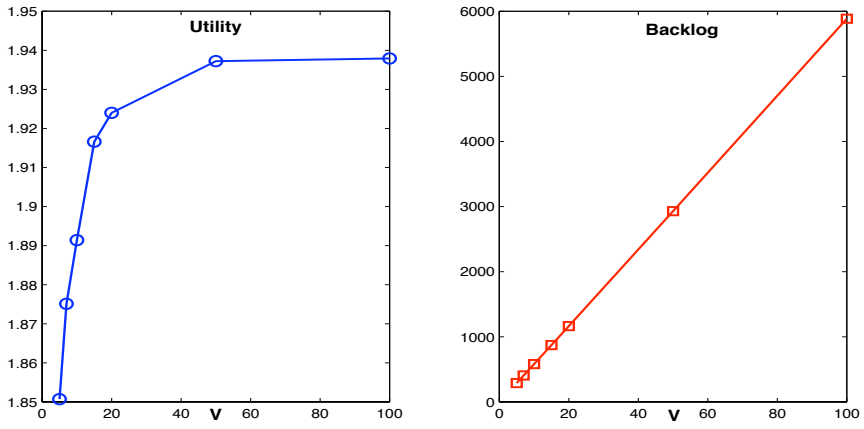

Figure 6.3: Utility and backlog performance of PMW.

sample path queue processes in the first 10^4 slots under $V = 100$. We see that no queue has an underflow. This shows that all the activation decisions of PMW are feasible. It is also easy to verify that the queueing bounds (6.57) and (6.58) hold. We observe

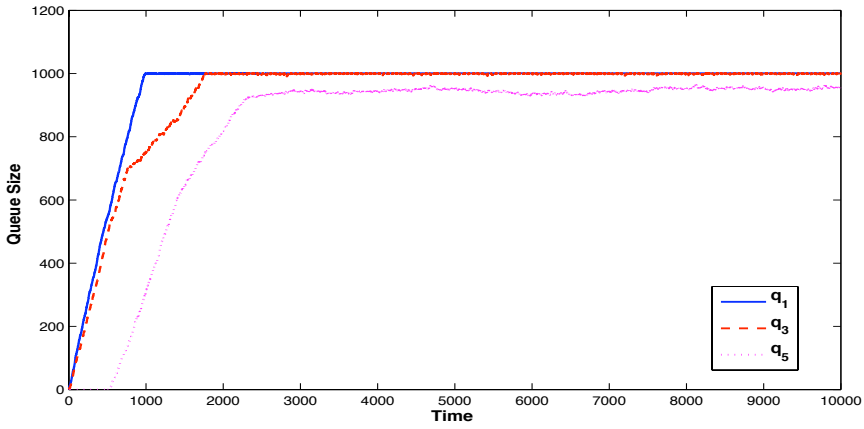

Figure 6.4: Sample path backlog processes with $V = 100$.

in Fig. 6.4 that the queue sizes usually fluctuate around certain fixed values. This is consistent with the exponential attraction result in Theorem 6. Hence our results can

also be extended, using the results developed in [HN11a], to achieve an average utility that is within $O(1/V)$ of the optimal with only $\Theta([\log(V)]^2)$ average backlog size.

6.8 Perturbation and tracking the Lagrange multiplier

Note that although the main difference between the PMW algorithm and the usual Max-Weight algorithm is the use of the perturbation vector $\boldsymbol{\theta}$, this seemingly small modification has a large effect on algorithm design for processing networks. To see this, first recall that we have shown in Section 6.1.2 that *the usual Max-Weight algorithm cannot be applied to this problem.*

The reason perturbation resolves this issue is as follows. It has been shown in Chapter 4 that Max-Weight applied to a queueing problem is equivalent to solving a corresponding deterministic optimization problem using the randomized incremental subgradient method (RISM) [BNO03], with the backlog vector being the Lagrange multiplier. Thus *Max-Weight relies on using the backlog vector to track the optimal Lagrange multiplier value* for achieving the desired utility performance, and the backlog vector under Max-Weight will move towards the optimal Lagrange multiplier. However, in processing network problems, due to the equality constraints in (6.32), the optimal Lagrange multiplier value may be *negative*. In this case, we can no longer use only the queue size to represent the Lagrange multiplier as the queue will be stuck at zero and the Max-Weight algorithm will not run properly. This is shown in Fig. 6.5.

The PMW algorithm instead resolves this problem by using the *queue size minus the perturbation* to represent the Lagrange multiplier. This is equivalent to "lifting" the

Lagrange multiplier by the perturbation value, and making it trackable by the queue size. If we choose the perturbation carefully enough to ensure that no further deviation from the optimal Lagrange multiplier will happen, we can guarantee that no underflow ever happens. Then under the PMW algorithm, the queue size minus the perturbation will move towards the optimal Lagrange multiplier and we can thus achieve the close-to-optimal utility performance, as in the data network case.

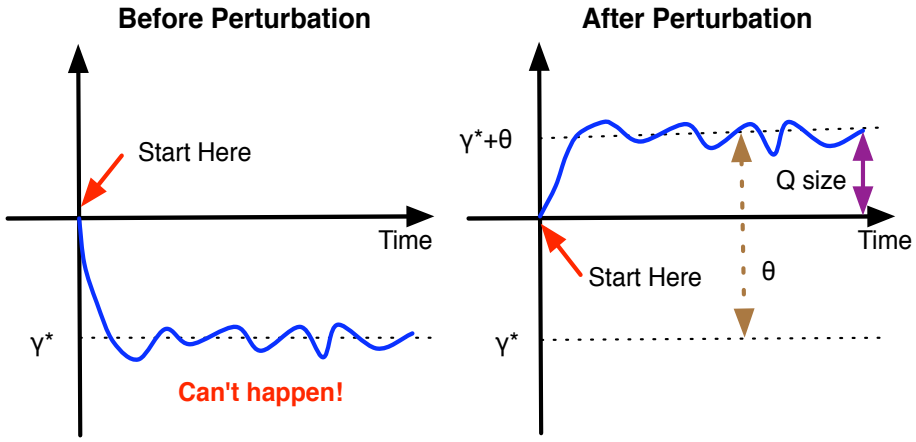

Figure 6.5: An explanation on why perturbation is needed and effective.

6.9 Chapter summary

In this chapter, we develop the Perturbed Max-Weight algorithm (PMW) for utility optimization problems in complex networks that involve the "no-underflow" constraints. PMW is based on the usual Max-Weight algorithm for data networks. It has two main functionalities: queue underflow prevention and utility optimal scheduling. PMW simultaneously achieves both objectives by carefully perturbing the weights used in the usual

Max-Weight algorithm. We show that PMW is able to achieve an $[O(1/V), O(V)]$ utility-backlog tradeoff. The PMW algorithm developed here can be applied to problems in the areas of data fusion, stream processing and cloud computing.

6.10 Proofs of the chapter

6.10.1 Proof of Theorem 23

We prove Theorem 23 in this section, using an argument similar to the one used in [HN10c].

Proof. (Theorem 23) Consider any stable scheduling policy Π, i.e., the conditions (6.24) and (6.27) are satisfied under Π. Let $\{(f(0), \boldsymbol{A}(0), \boldsymbol{\mu}(0)), (f(1), \boldsymbol{A}(1), \boldsymbol{\mu}(1)), ...\}$ be a sequence of (utility, arrival, service) triple generated by Π. Then there exists a subsequence of times $\{T_i\}_{i=1,2,...}$ such that $T_i \to \infty$ and that the limiting time average utility over times T_i is equal to the liminf average utility under Π (defined by (6.28)). Now define the conditional average of utility, and arrival minus service over T slots to be:

$$(\phi^{(s_i)}(T); \epsilon_1^{(s_i)}(T); ...; \epsilon_r^{(s_i)}(T)) \triangleq \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\big\{f(t); \epsilon_1(t); ...; \epsilon_r(t) \mid S(t) = s_i\big\},$$

where $\epsilon_j(t) = A_j(t) - \mu_j(t)$. Using Caratheodory's theorem, it can be shown, as in [HN10c] that, there exists a set of variables $\{a_k^{(s_i)}(T)\}_{k=1}^{r+2}$ and a set of actions $\{x_k^{(s_i)}(T)\}_{k=1}^{r+2}$ such that:

$$\phi^{(s_i)}(T) = \sum_{k=1}^{r+2} a_k^{(s_i)}(T) f(s_i, x_k^{(s_i)}(T)),$$

and for all $j = 1, ..., r$ that:

$$\epsilon_j^{(s_i)}(T) = \sum_{k=1}^{r+2} a_k^{(s_i)}(T)[A_j(s_i, x_k^{(s_i)}(T)) - \mu_j(s_i, x_k^{(s_i)}(T))].$$

Now using the continuity of $f(s_i,\cdot), A_j(s_i,\cdot), \mu_j(s_i,\cdot)$, and the compactness of all the actions sets $\mathcal{X}^{(s_i)}$, we can thus find a sub-subsequence $\tilde{T}_i \to \infty$ of $\{T_i\}_{i=1,2,...}$ that:

$$a_k^{(s_i)}(\tilde{T}_i) \to a_k^{(s_i)}, x_k^{(s_i)}(\tilde{T}_i)) \to x_k^{(s_i)}, \phi^{(s_i)}(\tilde{T}_i) \to \phi^{(s_i)}, \epsilon_j^{(s_i)}(\tilde{T}_i) \to \epsilon_j^{(s_i)}, \forall\, j = 1, ..., r.$$

Therefore the time average utility under the policy Π can be expressed as:

$$f_{av}^{\Pi} = \sum_{s_i} \pi_{s_i} \phi^{(s_i)} = \sum_{s_i} \pi_{s_i} \sum_{k=1}^{r+2} a_k^{(s_i)} f(s_i, x_k^{(s_i)}). \tag{6.62}$$

Similarly, the average arrival rate minus the average service rate under Π can be written as:

$$\begin{aligned} \epsilon_j &= \sum_{s_i} \pi_{s_i} \epsilon_j^{(s_i)} \\ &= \sum_{s_i} \pi_{s_i} \sum_{k=1}^{r+2} a_k^{(s_i)} [A_j(s_i, x_k^{(s_i)}) - \mu_j(s_i, x_k^{(s_i)})] \\ &\leq 0. \end{aligned} \tag{6.63}$$

The last inequality is due to the fact that Π is a stable policy and that $\mathbb{E}\{q_j(0)\} < \infty$, hence the average arrival rate to any q_j must be no more than the average service rate of the queue [Nee03]. However, by (6.24) we see that what is consumed from a queue is always no more that what is generated into the queue. This implies that the input rate into a queue is always no less than its output rate. Thus, $\epsilon_j \geq 0$ for all j. Therefore we conclude that $\epsilon_j = 0$ for all j. Using this fact and (6.62), we see that $V f_{av}^{\Pi} \leq \phi^*$, where ϕ^* is given in (6.30). This proves Theorem 23. □

6.10.2 Proof of Lemma 8

We prove Lemma 8 here.

Proof. (Lemma 8) It is easy to see from (6.30) that the dual function is given by:

$$\hat{g}(\boldsymbol{\gamma}) = \sup_{x_k^{(s_i)}, a_k^{(s_i)}} \sum_{s_i} \pi_{s_i} \Bigg\{ \sum_{k=1}^{r+2} a_k^{(s_i)} V f(s_i, x_k^{(s_i)}) - \sum_j \gamma_j \sum_{k=1}^{r+2} a_k^{(s_i)} \big[A_j(s_i, x_k^{(s_i)}) - \mu_j(s_i, x_k^{(s_i)})\big] \Bigg\}. \tag{6.64}$$

Due to the use of the $\{a_k^{(s_i)}\}_{i=1,...,M}^{k=1,...,r+2}$ variables, it is easy to see that $\hat{g}(\boldsymbol{\gamma}) \geq g(\boldsymbol{\gamma})$. However, if $\{x^{(s_i)}\}_{i=1}^M$ is a set of maximizers of $g(\boldsymbol{\gamma})$, then the set of variables $\{x_k^{(s_i)}, a_k^{(s_i)}\}_{i=1,...,M}^{k=1,...,r+2}$ where for each s_i, $x_k^{(s_i)} = x^{(s_i)}$ for all k, and $a_1^{(s_i)} = 1$ with $a_k^{(s_i)} = 0$ for all $k \geq 2$, will also be maximizers of $\hat{g}(\boldsymbol{\gamma})$. Thus $g(\boldsymbol{\gamma}) \geq \hat{g}(\boldsymbol{\gamma})$. This shows that $g(\boldsymbol{\gamma}) = \hat{g}(\boldsymbol{\gamma})$, and hence $g(\boldsymbol{\gamma})$ is the dual function of (6.30). $g(\boldsymbol{\gamma}^*) \geq \phi^*$ follows from weak duality [BNO03]. □

6.10.3 Proof of Lemma 9

Here we prove Lemma 9.

Proof. Using the queueing equation (6.26), we have:

$$\begin{aligned}
[q_j(t+1) - \theta_j]^2 &= [(q_j(t) - \mu_j(t) + A_j(t)) - \theta_j]^2 \\
&= [q_j(t) - \theta_j]^2 + (\mu_j(t) - A_j(t))^2 - 2\big(q_j(t) - \theta_j\big)[\mu_j(t) - A_j(t)] \\
&\leq [q_j(t) - \theta_j]^2 + 2\delta_{max}^2 - 2\big(q_j(t) - \theta_j\big)[\mu_j(t) - A_j(t)].
\end{aligned}$$

Multiplying both sides with $\frac{w_j}{2}$ and summing the above over $j = 1, ..., r$, we see that:

$$L(t+1) - L(t) \leq B - \sum_{j=1}^{r} w_j \big(q_j(t) - \theta_j\big)[\mu_j(t) - A_j(t)],$$

where $B = \delta_{max}^2 \sum_{j=1}^r w_j$. Now add to both sides the term $-Vf(t)$, we get:

$$L(t+1) - L(t) - Vf(t) \leq B - Vf(t) - \sum_{j=1}^{r} w_j \big(q_j(t) - \theta_j\big)[\mu_j(t) - A_j(t)]. \tag{6.65}$$

Taking expectations over the random network state $S(t)$ on both sides conditioning on $\boldsymbol{q}(t)$ proves the lemma. □

6.10.4 Proof of Theorem 24

Here we prove Theorem 24. We first have the following simple lemma.

Lemma 11. *For any network state s_i, we have:*

$$D^{(s_i)*}_{\boldsymbol{\theta},\boldsymbol{q}(t)} = g_{s_i}((\boldsymbol{q}(t)-\boldsymbol{\theta})\otimes \boldsymbol{w}), \tag{6.66}$$

where $\boldsymbol{w} = (w_1, ..., w_r)^T$ and $\boldsymbol{a}\otimes\boldsymbol{b} = (a_1b_1, ..., a_nb_n)^T$.

Proof. By comparing (6.41) with (6.36), we see that the lemma follows. □

Proof. (Theorem 24) We first recall the equation (6.65) as follows:

$$L(t+1) - L(t) - Vf(t) \leq B - Vf(t) - \sum_{j=1}^{r} w_j\big(q_j(t)-\theta_j\big)[\mu_j(t) - A_j(t)]. \tag{6.67}$$

Using $D^{(s_i)}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x)$ defined in (6.40), this can be written as:

$$L(t+1) - L(t) - Vf(t) \leq B - D^{(S(t))}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x(t)).$$

Here $x(t)$ is PMW's action at time t. According to Condition 1, we see that for any network state $S(t) = s_i$, PMW ensures (6.24), and that:

$$D^{(s_i)}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x) \geq D^{(s_i)*}_{\boldsymbol{\theta},\boldsymbol{q}(t)} - C.$$

Using (6.66), this implies that under PMW,

$$L(t+1) - L(t) - Vf(t) \leq B - g_{s_i}((\boldsymbol{q}(t)-\boldsymbol{\theta})\otimes\boldsymbol{w}) + C.$$

Taking expectations over the random network state on both sides conditioning on $\boldsymbol{q}(t)$, and using (6.37), i.e., $g(\boldsymbol{\gamma}) = \sum_{s_i}\pi_{s_i}g_{s_i}(\boldsymbol{\gamma})$, we get:

$$\Delta(t) - V\mathbb{E}\big\{f(t)\mid \boldsymbol{q}(t)\big\} \leq B + C - g((\boldsymbol{q}(t)-\boldsymbol{\theta})\otimes\boldsymbol{w}). \tag{6.68}$$

Now using Theorem 23 and Lemma 8, we have:

$$Vf^*_{av} \leq \phi^* \leq g(\boldsymbol{\gamma}^*) \leq g((\boldsymbol{q}(t)-\boldsymbol{\theta})\otimes\boldsymbol{w}).$$

Therefore,

$$\Delta(t) - V\mathbb{E}\big\{f(t) \mid \boldsymbol{q}(t)\big\} \leq B + C - Vf^*_{av}. \tag{6.69}$$

Taking expectations over $\boldsymbol{q}(t)$ on both sides and summing the above over $t = 0, ..., T-1$, we get:

$$\mathbb{E}\big\{L(T) - L(0)\big\} - \sum_{t=0}^{T-1} V\mathbb{E}\big\{f(t)\big\} \leq T(B+C) - TVf^*_{av}.$$

Rearranging terms, dividing both sides by VT, using the facts that $L(t) \geq 0$ and $\mathbb{E}\big\{L(0)\big\} < \infty$, and taking the liminf as $T \to \infty$, we get:

$$f^{PMW}_{av} \geq f^*_{av} - (B+C)/V. \tag{6.70}$$

This proves (6.42). Now we prove (6.43). First, by using the definition of $\hat{g}(\boldsymbol{\gamma})$ in (6.64), and plugging in the $\{x_k^{(s_i)}, \vartheta_k^{(s_i)}\}_{i=1,...,M}^{k=1,...,r+2}$ variables in the η-slackness assumption (6.25) in Section 6.2.2, we see that:

$$\hat{g}((\boldsymbol{q}(t) - \boldsymbol{\theta}) \otimes \boldsymbol{w}) \geq \eta \sum_{j=1}^{r} w_j[q_j(t) - \theta_j] - V\delta_{max}. \tag{6.71}$$

This by Lemma 8 implies that:

$$g((\boldsymbol{q}(t) - \boldsymbol{\theta}) \otimes \boldsymbol{w}) \geq \eta \sum_{j=1}^{r} w_j[q_j(t) - \theta_j] - V\delta_{max}.$$

Using this in (6.68), we get:

$$\Delta(t) - V\mathbb{E}\big\{f(t) \mid \boldsymbol{q}(t)\big\} \leq B + C + V\delta_{max} - \eta \sum_{j=1}^{r} w_j[q_j(t) - \theta_j].$$

We can now use a similar argument as above to get:

$$\eta \sum_{t=0}^{T-1} \sum_{j=1}^{r} w_j \mathbb{E}\big\{[q_j(t) - \theta_j]\big\} \leq T(B+C) + 2TV\delta_{max} + \mathbb{E}\big\{L(0)\big\}.$$

Dividing both sides by ηT and taking the limsup as $T \to \infty$, we get:

$$\overline{q}^{PMW} \quad \leq \quad \frac{B + C + 2V\delta_{max}}{\eta} + \sum_{j=1}^{r} w_j\theta_j.$$

This completes the proof the theorem. □

6.10.5 Choosing the $\{w_j\}_{j=1}^r$ values

Here we describe how to choose the $\{w_j\}_{j=1}^r$ values to satisfy (6.56). We first let K be the maximum number of processors that any path going from a queue to an output processor can have. We have that $K \leq |N_p|$ since there is no cycle in the network. The following algorithm terminates in K iterations. We use $w_j(k)$ to denote the value of w_j in the k^{th} iteration. In the following, we use q_{h_n} to denote the demand queue of a processor P_n.

1. At Iteration 1, denote the set of queues that serve as supply queues for any output processor by $\mathbb{Q}_1^l$, i.e.,

$$\mathbb{Q}_1^l = \{q_j : \mathbb{P}_j^S \cap \mathcal{P}^o \neq \phi\}.$$

   Then, set $w_j(1) = 1$ for each $q_j \in \mathbb{Q}_1^l$. Also, set $w_j(1) = 0$ for all other $q_j \notin \mathbb{Q}_1^l$.

2. At Iteration $k = 2, ..., K$, denote by $\mathbb{Q}_k^l$ the set of queues that serve as supply queues for any processor whose demand queue is in $\mathbb{Q}_{k-1}^l$, i.e.,

$$\mathbb{Q}_k^l = \{q_j : \exists\, P_n \in \mathbb{P}_j^S \;\; s.t. \;\; \mathbb{Q}_n^D \in \mathbb{Q}_{k-1}^l\}.$$

   Then, set:

$$w_j(k) = \max\big[w_j(k-1), \max_{n \in \mathbb{P}_j^S} \frac{w_{h_n}(k-1)\alpha_{nh_n}}{\beta_{nj}}\big], \tag{6.72}$$

   where α_{nh_n} is the amount P_n generates into q_{h_n}, which is the demand queue of P_n. Also, set $w_j(k) = w_j(k-1)$ for all $q_j \notin \mathbb{Q}_k^l$.

3. Output the $\{w_j\}_{j=1}^r$ values.

The following lemma shows that the above algorithm outputs a set of $\{w_j\}_{j=1}^r$ values that satisfy (6.56).

Lemma 12. *The $\{w_j\}_{j=1}^r$ values generated by the above algorithm satisfy (6.56).*

Proof. (Proof of Lemma 12) The proof consists of two main steps. In the first step, we show that the algorithm updates each w_j value at least once. This shows that all the w_j values for all the queues that serve as demand queues are updated at least once. In the second step, we show that if q_h is the demand queue of a processor $P_i \in \mathcal{P}^{in}$, then every time after w_h is updated, the algorithm will also update w_j for any $q_j \in \mathbb{Q}_i^S$ before it terminates. This ensures that (6.56) holds for any $P_i \in \mathcal{P}^{in}$ and hence proves the lemma.

First we see that after K iterations, we must have $\mathcal{Q} \subset \bigcup_{\tau=1}^K \mathbb{Q}_\tau^l$. This is because at Iteration k, we include in $\bigcup_{\tau=1}^k \mathbb{Q}_\tau^l$ all the queues starting from which there exists a path to an output processor that contains k processors. Thus all the w_j values are updated at least once.

Now consider a queue q_h. Suppose q_h is the demand queue of a processor $P_i \in \mathcal{P}^{in}$. We see that there exists a time $\hat{k} \leq K$ at which w_h is last modified. Suppose w_h is last modified at Iteration $\hat{k} < K$, in which case $q_h \in \mathbb{Q}_{\hat{k}}^l$. Then all the queues $q_j \in \mathbb{Q}_i^S$ will be in $\mathbb{Q}_{\hat{k}+1}^l$. Thus their w_j values will be modified at Iteration $\hat{k}+1 \leq K$. This implies that at Iteration $\hat{k}+1$, we will have $w_j(\hat{k}+1)\beta_{ij} \geq w_h(\hat{k})\alpha_{ih}$. Since $q_h \notin \mathbb{Q}_k^l$ for $k \geq \hat{k}+1$, we have $w_h(k) = w_h(\hat{k})$ for all $k \geq \hat{k}+1$. Therefore $w_j(k)\beta_{ij} \geq w_h(k)\alpha_{ih}\ \forall\ \hat{k}+1 \leq k \leq K$, because $w_j(k)$ is not decreasing.

Therefore the only case when the algorithm can fail is when w_h is updated at Iteration $k = K$, in which case w_h may increase but the w_j values for $q_j \in \mathbb{Q}_i^S$ are not modified accordingly. However, since w_h is updated at Iteration $k = K$, this implies that there exists a path from q_h to an output processor that has K processors. This in turn implies

that starting from any $q_j \in \mathbb{Q}_i^S$, there exists a path to an output processor that contains $K+1$ processors. This contradicts the definition of K. Thus the lemma follows. $\square$

As a concrete example, we consider the example in Fig. 6.2, with the assumption that each processor, when activated, consumes one unit of content from each of its supply queues and generates two units of contents into its demand queue. In this example, we see that $K = 3$. Thus the algorithm works as follows:

1. Iteration 1, denote $\mathbb{Q}_1^l = \{q_4, q_5, q_6\}$, set $w_4(1) = w_5(1) = w_6(1) = 1$. For all other queues, set $w_j(1) = 0$.

2. Iteration 2, denote $\mathbb{Q}_2^l = \{q_1, q_2, q_3, q_4, q_5\}$, set $w_1(2) = w_2(2) = w_3(2) = w_4(2) = w_5(2) = 2$. Set $w_6(2) = 1$.

3. Iteration 3, denote $\mathbb{Q}_3^l = \{q_2, q_3\}$, set $w_2(3) = w_3(3) = 4$. Set $w_1(3) = w_4(3) = w_5(3) = 2$, $w_6(3) = 1$.

4. Terminate and output $w_1 = w_4 = w_5 = 2$, $w_2 = w_3 = 4$, $w_6 = 1$.

6.10.6 Proof of Lemma 10

Here we prove Lemma 10 by comparing the values of the three terms in $D^{(S(t))}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x)$ in (6.50) under PMW versus their values under the action that maximizes $D^{(S(t))}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x)$ in (6.50) subject to only the constraints $D_j(t) \in [0,1]$, $\forall\, j \in \mathcal{Q}^s$ and $\boldsymbol{I}(t) \in \mathcal{I}$, called the max-action. That is, under the max-action, $D^{(S(t))}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x) = D^{(S(t))*}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x)$. Note that the max-action differs from PMW only in that it does not consider the queue edge constraint.

Proof. (A) We see that the first term, i.e., $-\sum_{j\in\mathcal{Q}^s}\left[Vc_j(t)+w_j(q_j(t)-\theta)\right]D_j(t)R_j(t)$ is maximized under PMW. Thus its value is the same as that under the max-action.

(B) We now show that for any processor $P_n \in \mathcal{P}$, if it violates the queue edge constraint, then its weight is bounded by $M_p w_{max}\nu_{max}\beta_{max}$. This will then be used in Part (C) below to show that the value of $D^{(S(t))}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x)$ under PMW is within a constant of $D^{(S(t))*}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x)$.

(B-I) For any $P_i \in \mathcal{P}^{in}$, the following are the only two cases under which P_i violates the queue edge constraint.

1. Its demand queue $q_h(t) \geq \theta$. In this case, it is easy to see from (6.52) and (6.59) that:
$$W_i^{(in)}(t) \leq \sum_{j\in\mathbb{Q}_i^S} w_j\nu_{max}\beta_{ij} \leq M_p w_{max}\nu_{max}\beta_{max}. \tag{6.73}$$
2. At least one of P_i's supply queue has a queue size less than $M_q^s\beta_{max}$. In this case, we denote $\hat{\mathbb{Q}}_i^S = \{q_j \in \mathbb{Q}_i^S : q_j(t) \geq M_q^s\beta_{max}\}$. Then, we see that:
$$\begin{aligned} W_i^{(in)}(t) &= \sum_{j\in\hat{\mathbb{Q}}_i^S} w_j[q_j(t)-\theta]\beta_{ij} - w_h[q_h(t)-\theta]\alpha_{ih} + \sum_{j\in\mathbb{Q}_i^S/\hat{\mathbb{Q}}_i^S} w_j[q_j(t)-\theta]\beta_{ij} - VC_i(t) \\ &\leq \sum_{j\in\hat{\mathbb{Q}}_i^S} w_j\nu_{max}\beta_{ij} + w_h\theta\alpha_{ih} + \sum_{j\in\mathbb{Q}_i^S/\hat{\mathbb{Q}}_i^S} w_j[M_q^s\beta_{max}-\theta]\beta_{ij}. \end{aligned}$$
Here $q_h = \mathbb{Q}_i^D$. Now by our selection of $\{w_j\}_{j=1}^r$, $w_j\beta_{ij} \geq w_h\alpha_{ih}$ for any $q_j \in \mathbb{Q}_i^S$. Also using $\nu_{max} \geq M_q^s\beta_{max}$, we have:
$$W_i^{(in)}(t) \leq M_p w_{max}\nu_{max}\beta_{max}. \tag{6.74}$$

(B - II) For any $P_k \in \mathcal{P}^o$, we see that it violates the queue edge constraint only when at least one of its supply queues has size less than $M_q^s\beta_{max}$. In this case, we see that:
$$W_k^{(o)}(t) \leq \sum_{j\in\hat{\mathbb{Q}}_k^S} w_j[q_j(t)-\theta]\beta_{kj} + Vp_k(t)\alpha_{ko} + \sum_{j\in\mathbb{Q}_k^S/\hat{\mathbb{Q}}_k^S} w_j(M_q^s\beta_{max}-\theta)\beta_{ij}$$

$$\leq M_p w_{max} \nu_{max} \beta_{max} + V \alpha_{max} p_{max} - w_{min} \theta \beta_{min}.$$

This by (6.55) implies that:

$$W_k^{(o)}(t) \leq M_p w_{max} \nu_{max} \beta_{max}. \tag{6.75}$$

Using (6.73), (6.74) and (6.75), we see that whenever a processor violates the queue edge constraint, its weight is at most $M_p w_{max} \nu_{max} \beta_{max}$.

(C) We now show that the value of $D^{(S(t))}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x)$ under PMW satisfies $D^{(S(t))}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x) \geq D^{(S(t))*}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x) - C$, where $C = N_p M_p w_{max} \nu_{max} \beta_{max}$.

To see this, let $\boldsymbol{I}^*(t)$ be the activation vector obtained by the max-action, and let $W^*(t)$ be the value of (6.54) under $\boldsymbol{I}^*(t)$. We also use $\boldsymbol{I}^{PMW}(t)$ and $W^{PMW}(t)$ to denote the activation vector chosen by the PMW algorithm and the value of (6.54) under $\boldsymbol{I}^{PMW}(t)$. We now construct an alternate activation vector $\hat{\boldsymbol{I}}(t)$ by changing all elements in $\boldsymbol{I}^*(t)$ corresponding to the processors that violate the queue edge constraints to zero. Note then $\hat{\boldsymbol{I}}(t) \in \mathcal{I}$ is a feasible activation vector at time t, under which no processor violates the queue edge constraint. By Part (B) above, we see that the value of (6.54) under $\hat{\boldsymbol{I}}(t)$, denoted by $\hat{W}(t)$, satisfies:

$$\hat{W}(t) \geq W^*(t) - N_p M_p w_{max} \nu_{max} \beta_{max}.$$

Now since $\boldsymbol{I}^{PMW}(t)$ maximizes the value of (6.54) subject to the queue edge constraints, we have:

$$W^{PMW}(t) \geq \hat{W}(t) \geq W^*(t) - N_p w_{max} M_p \nu_{max} \beta_{max}.$$

Thus, by combining the above and Part (A), we see that PMW maximizes the $D^{(S(t))}_{\boldsymbol{\theta},\boldsymbol{q}(t)}(x)$ to within $C = N_p M_p w_{max} \nu_{max} \beta_{max}$ of the maximum. □

Chapter 7

Utility optimal scheduling in energy harvesting networks

Recent developments in hardware design have enabled many general wireless networks to support themselves by harvesting energy from the environment, for instance, by converting mechanical vibration into energy [MMMA+01], by using solar panels [RKH+05], by utilizing thermoeletric generators [CC08], or by converting ambient radio power into energy [GKK+09]. Such harvesting methods are also referred to as "recycling" energy [ene10]. This energy harvesting ability is crucial for many network design problems. It frees the network devices from having an "always on" energy source and provides a way of operating the network with a potentially infinite lifetime. These two advantages are particularly useful for networks that work autonomously, e.g., wireless sensor networks that perform monitoring tasks in dangerous fields [WALJ+06], tactical networks [HC10], or wireless handheld devices that operate over a longer period [GR09].

However, to take full advantage of the energy harvesting technology, efficient scheduling algorithms must consider the finite capacity for energy storage at each network node, and the "no-energy-outage" constraint which requires that the network nodes cannot spend more energy than what is stored. These two constraints impose great challenges

on algorithm design for such networks. In this chapter, we propose a general framework for modeling energy harvesting network problems based on the general complex network model presented in Chapter 6, and develop an optimal online energy harvesting and scheduling algorithm using the perturbation technique developed in Chapter 6.

7.1 The network model

We consider a general interconnected multi-hop network that operates in slotted time. The network is modeled by a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N} = \{1, 2, ..., N\}$ is the set of the N nodes in the network, and $\mathcal{L} = \{[n, m],\ n, m \in \mathcal{N}\}$ is the set of communication links in the network. For each node n, we use $\mathcal{N}_n^{(o)}$ to denote the set of nodes b with $[n, b] \in \mathcal{L}$, and use $\mathcal{N}_n^{(in)}$ to denote the set of nodes a with $[a, n] \in \mathcal{L}$. We then define $d_{max} \triangleq \max_n(|\mathcal{N}_n^{(in)}|, |\mathcal{N}_n^{(o)}|)$ to be the maximum in-degree/out-degree that any node $n \in \mathcal{N}$ can have.

7.1.1 The traffic and utility model

At every time slot, the network decides how many packets destined for node c to admit at node n. We call these packets the *commodity* c data and use $R_n^{(c)}(t)$ to denote the amount of new commodity c data admitted. We assume that $0 \leq R_n^{(c)}(t) \leq R_{max}$ for all n, c with some finite R_{max} at all time. [1] We assume that each commodity c at every node n is associated with a utility function $U_n^{(c)}(\overline{r}^{nc})$, where $\overline{r}^{nc}$ is the time average rate of the

[1] Note that this setting implicitly assumes that nodes always have packets to admit. The case when the number of packets available is random can also be incorporated into our model and solved by introducing auxiliary variables, as in [NML08]. Also note that this traffic admission model can be viewed as "shaping" the arrivals from some external sending nodes. One future extension of our model is to also evaluate the backlogs at these sending nodes.

commodity c traffic admitted into node n, defined as $\overline{r}^{nc} = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{R_n^{(c)}(\tau)\}$ (assume to exist for now). Each $U_n^{(c)}(r)$ function is assumed to be increasing, continuously differentiable, and strictly concave in r with a bounded first derivative and $U_n^{(c)}(0) = 0$. We use β^{nc} to denote the maximum first derivative of $U_n^{(c)}(r)$, i.e., $\beta^{nc} = (U_n^{(c)})'(0)$ and denote

$$\beta \triangleq \max_{n,c} \beta^{nc}. \tag{7.1}$$

7.1.2 The transmission model

[2] In order to deliver the data to their destinations, each node needs to allocate power over its outgoing links for data transmission at every time slot. To model the effect that the transmission rates typically also depend on the link conditions and that the link conditions may be time varying, we let $\boldsymbol{S}(t)$ be the network *channel state*, i.e., the N-by-N matrix where the (n, m) component of $\boldsymbol{S}(t)$ denotes the channel condition between nodes n and m. We assume that $\boldsymbol{S}(t)$ takes values in some finite set $\mathcal{S} = (s_1, ..., s_{M_s})$. We assume in the following that the energy state (defined later) and $\boldsymbol{S}(t)$ pair is i.i.d. every slot. At every time slot, if $\boldsymbol{S}(t) = s_i$, then the power allocation vector $\boldsymbol{P}(t) = (P_{[n,m]}(t), [n, m] \in \mathcal{L})$, where $P_{[n,m]}(t)$ is the power allocated to link $[n, m]$ at time t, must be chosen from some feasible power allocation set $\mathcal{P}^{(s_i)}$. We assume that $\mathcal{P}^{(s_i)}$ is compact for all s_i, and that every power vector in $\mathcal{P}^{(s_i)}$ satisfies the constraint that for each node n, $0 \leq \sum_{b\in\mathcal{N}_n^{(o)}} P_{[n,b]}(t) \leq P_{max}$ for some $P_{max} < \infty$. Also, we assume that setting any $P_{[n,m]}$ in a vector $\boldsymbol{P} \in \mathcal{P}^{(s_i)}$ to zero yields another power vector that is still in $\mathcal{P}^{(s_i)}$. Given the channel state $\boldsymbol{S}(t)$ and the power allocation vector $\boldsymbol{P}(t)$, the transmission rate

[2] The transmission model is similar to the one used in Chapter 5. We restate it here for completeness.

over the link $[n,m]$ is given by the rate-power function $\mu_{[n,m]}(t) = \mu_{[n,m]}(\boldsymbol{S}(t), \boldsymbol{P}(t))$. For each s_i, we assume that the function $\mu_{[n,m]}(s_i, \boldsymbol{P})$ satisfies the following properties:

Property 1. *For any $\boldsymbol{P}, \boldsymbol{P'} \in \mathcal{P}^{(s_i)}$, where $\boldsymbol{P'}$ is obtained by changing any single component $P_{[n,m]}$ in $\boldsymbol{P}$ to zero, we have for some finite constant $\delta > 0$ that:* [3]

$$\mu_{[n,m]}(s_i, \boldsymbol{P}) \leq \mu_{[n,m]}(s_i, \boldsymbol{P'}) + \delta P_{[n,m]}. \tag{7.2}$$

Property 2. *If $\boldsymbol{P'}$ is obtained by setting the entry $P_{[n,b]}$ in $\boldsymbol{P}$ to zero, then:*

$$\mu_{[a,m]}(s_i, \boldsymbol{P}) \leq \mu_{[a,m]}(s_i, \boldsymbol{P'}),\ \forall\, [a,m] \neq [n,b]. \tag{7.3}$$

Property 1 states that the rate obtained over a link $[n,m]$ is upper bounded by some linear function of the power allocated to it; whereas Property 2 states that reducing the power over any link does not reduce the rate over any other links. We see that Properties 1 and 2 can be satisfied by most rate-power functions, e.g., when the rate function is differentiable and has finite directional derivatives with respect to power [Nee06c], and the link rates do not improve with increased interference.

We also assume that there exists some finite constant μ_{max} such that $\mu_{[n,m]}(t) \leq \mu_{max}$ for all time slots under any power allocation vector and any channel state $\boldsymbol{S}(t)$. [4] In the following, we also use $\mu^{(c)}_{[n,b]}(t)$ to denote the rate allocated to the commodity c data over link $[n,b]$ at time t. At every time t, we have:

$$\sum_c \mu^{(c)}_{[n,b]}(t) \leq \mu_{[n,b]}(t), \forall\ [n,b]. \tag{7.4}$$

[3] This is also known as Lipschitz-continuity [BNO03].

[4] In our transmission model, we did not explicitly take into account the reception power. We can easily incorporate that it our model at the expense of more complicated notation. In that case, our algorithm will also optimize over the reception power consumption, and the results in this chapter still hold.

7.1.3 The energy queue model

We now specify the energy model. Every node in the network is assumed to be powered by a *finite* capacity energy storage device, e.g., a battery or ultra-capacitor [SMJG10]. We model such a device using an *energy queue*. We use the energy queue size at node n at time t, denoted by $E_n(t)$, to measure the amount of energy left in the storage device at node n at time t. We assume that each node n knows its own current energy availability $E_n(t)$. In any time slot t, the power allocation vector $\boldsymbol{P}(t)$ must satisfy the following "energy-availability" constraint: [5]

$$\sum_{b\in\mathcal{N}_n^{(o)}} P_{[n,b]}(t) \leq E_n(t), \quad \forall\ n. \tag{7.5}$$

That is, the consumed power must be no more than what is available. Each node in the network is assumed to be capable of harvesting energy from the environment, using, for instance, solar panels [SMJG10]. However, the amount of harvestable energy in a time slot is typically not fixed and varies over time. We use $h_n(t)$ to denote the amount of energy harvestable by node n at time t, and denote by $\boldsymbol{h}(t) = (h_1(t), ..., h_N(t))$ the harvestable energy vector at time t, called the *energy state*. We assume that $\boldsymbol{h}(t)$ takes values in some finite set $\mathcal{H} = \{\boldsymbol{h}_1, ..., \boldsymbol{h}_{M_h}\}$. In the following, we carry out the algorithm construction and analysis assuming that the pair $[\boldsymbol{h}(t), \boldsymbol{S}(t)]$ is i.i.d. over slots (possibly correlated in the same slot), with distribution $\pi_{\boldsymbol{h}_i,s_j}$ and marginal distributions $\pi_{\boldsymbol{h}_i}$ and $\pi_{\boldsymbol{s}_j}$, respectively. We then extend the results to the case when they are Markovian.

We assume that there exists $h_{max} < \infty$ such that $h_n(t) \leq h_{max}$ for all n, t. The energy harvested at time t is assumed to be available for use at time $t + 1$. In the following,

[5] We measure time in unit size "slots," so that our power $P_{[n,b]}(t)$ has units of energy/slot, and $P_{[n,b]}(t)\times$ (1 slot) is the resulting energy use in one slot. For simplicity, we suppress the implicit multiplication by 1 slot when converting between power and energy.

it is convenient for us to assume that each energy queue has infinite capacity, and that each node can decide whether or not to harvest energy in each slot. We model this harvesting decision by using $e_n(t) \in [0, h_n(t)]$ to denote the amount of energy that is *actually* harvested at time t. We show later that our algorithm always harvests energy when the energy queue is below a finite threshold of size $O(1/\epsilon)$ and drops it otherwise. Thus, it can be implemented with finite capacity storage devices. We also discuss why the algorithms developed under our model is also very useful in practice.

7.1.4 Queueing dynamics

Let $\boldsymbol{Q}(t) = (Q_n^{(c)}(t), n, c \in \mathcal{N})$, $t = 0, 1, 2, ...$ be the data queue backlog vector in the network, where $Q_n^{(c)}(t)$ is the amount of commodity c data queued at node n. We assume the following queueing dynamics:

$$Q_n^{(c)}(t+1) \leq \big[Q_n^{(c)}(t) - \sum_{b \in \mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)}(t)\big]^+ + \sum_{a \in \mathcal{N}_n^{(in)}} \mu_{[a,n]}^{(c)}(t) + R_n^{(c)}(t), \qquad (7.6)$$

with $Q_n^{(c)}(0) = 0$ for all $n, c \in \mathcal{N}$, $Q_c^{(c)}(t) = 0 \; \forall \, t$. The inequality in (7.6) is due to the fact that some nodes may not have enough commodity c packets to fill the allocated rates. In this chapter, we say that the network is *stable* if the following is met:

$$\overline{\boldsymbol{Q}} \triangleq \limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{n,c} \mathbb{E}\big\{Q_n^{(c)}(\tau)\big\} < \infty. \qquad (7.7)$$

Similarly, let $\boldsymbol{E}(t) = (E_n(t), n \in \mathcal{N})$ be the vector of the energy queue sizes. Due to the energy availability constraint (7.5), we see that for each node n, the energy queue $E_n(t)$ evolves according to the following: [6]

$$E_n(t+1) = E_n(t) - \sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}(t) + e_n(t), \tag{7.8}$$

with $E_n(0) = 0$ for all n. [7] Note again that by using the queueing dynamic (7.8), we start by assuming that each energy queue has infinite capacity. Later we show that under our algorithms, all the $E_n(t)$ values are *determinstically* upper bounded. Thus, we only need a finite energy capacity in algorithm implementation.

7.1.5 Utility maximization with energy management

The goal of the network is thus to design a joint flow control, routing and scheduling, and energy management algorithm that at every time slot admits the right amount of data $R_n^{(c)}(t)$, chooses a power allocation vector $\boldsymbol{P}(t) \in \mathcal{P}^{(s_i)}$ subject to (7.5), and transmits packets accordingly, so as to maximize the utility function:

$$U_{tot}(\overline{\boldsymbol{r}}) \;\; = \;\; \sum_{n,c} U_n^{(c)}(\overline{r}^{nc}), \tag{7.9}$$

subject to the network stability constraint (7.7). Here $\overline{\boldsymbol{r}} = (\overline{r}^{nc}, \forall\, n, c \in \mathcal{N})$ is the vector of the average expected admitted rates. Below, we refer to this problem as the *Utility Maximization with Energy Management problem* (UMEM).

[6] Note that we do not explicitly consider energy leakage due to the imperfections of the energy storage devices. This is a valid assumption if the rate of energy leakage is very small compared to the amount spent in each time slot.

[7] We can also pre-store energy in the energy queue and initialize $E_n(0)$ to any finite positive value up to its capacity. The results here will not be affected.

7.1.6 Discussion of the model

(I) Our model is quite general and can be used to model many networks where nodes are powered by finite capacity batteries, for instance, a field monitoring sensor network [WALJ+06], or many mobile ad hoc networks [PPC06]. Also, our model allows the harvestable energy to be correlated among network nodes. This is particularly useful: in practice, nodes that are colocated may have similar harvestable energy conditions.

(II) Although our model looks similar to the utility maximization model considered in [GNT06] and Chapter 4, the problem considered here is much more complicated. The main difficulty here is imposed by the constraint (7.5). Indeed, *(7.5) couples the current power allocation action and future actions, in that a current action may cause the energy queue to be empty and hence block some power allocation actions in the future.* The work in [GGT10] overcomes this "no-underflow" requirement by enforcing a positive drift constraint on the harvested energy and using Lyapunov optimization with this new constraint. Our approach is different and uses a modified Lyapunov function, which simplifies analysis and provides more explicit performance guarantees for the multi-hop case. Our MESA algorithm also fundamentally improves the resulting buffer size tradeoffs from $O(1/\epsilon)$ to $O([\log(1/\epsilon)]^2)$.

(III) Finally, note that our algorithm can also be shown to perform well under *arbitrary* $\boldsymbol{S}(t)$ and $\boldsymbol{h}(t)$ processes using the universal scheduling technique developed in [Nee10a].

7.2 Related work

There have been many previous articles developing algorithms for energy harvesting networks. [SMJG10] develops algorithms for a single sensor node for achieving maximum capacity and minimizing delay when the rate-power curve is linear. [KHZS07] considers the problem of optimal power management for sensor nodes, under the assumption that the harvested energy satisfies a leaky-bucket type property. [SK10] looks at the problem of designing energy-efficient schemes for maximizing the decay exponent of the queue length. [GGT10] develops scheduling algorithms to achieve close-to-optimal utility for energy harvesting networks with time varying channels. [LSS05] develops an energy-aware routing scheme that approaches optimality as the network size increases. Outside the energy harvesting context, [LSS07] considers the problem of maximizing the lifetime of a network with finite energy capacity and constructs a scheme that achieves a close-to-maximum lifetime. [Nee06c] and [Nee07] develop algorithms for minimizing the time average network energy consumption for stochastic networks with "always on" energy sources. However, most of the existing results focus on single-hop networks and often require sufficient statistical knowledge of the harvestable energy, and results for multi-hop networks often do not give explicit queueing bounds and do not provide explicit characterizations of the needed energy storage capacities.

7.3 Upper bounding the optimal network utility

In this section, we first obtain an upper bound on the optimal utility. This upper bound will be useful for our later analysis. The result is presented in the following theorem, in

which we use $\boldsymbol{r}^*$ to denote the optimal solution of the UMEM problem, subject to the constraint that the network nodes are powered by finite capacity energy storage devices. The V parameter in the theorem can be any positive constant that is greater or equal to 1, and is included for our later analysis.

Theorem 25. *The optimal network utility $U_{tot}(\boldsymbol{r}^*)$ satisfies: $VU_{tot}(\boldsymbol{r}^*) \leq \phi^*$, where ϕ^* is obtained over the class of stationary and randomized policies that have the following structure: allocate constant admission rates r^{nc} every slot; when $\boldsymbol{S}(t) = s_i$, choose a power vector $\boldsymbol{P}_k^{(s_i)}$ and allocate service rate $\mu_{[n,b]}^{(c)}(s_i, \boldsymbol{P}_k^{(s_i)})$ to node n with probability $\varrho_k^{(s_i)}$; and harvest energy $e_{n,k}^{(\boldsymbol{h}_i)}$ with probability $\varphi_k^{(\boldsymbol{h}_i)}$ when $\boldsymbol{h}(t) = \boldsymbol{h}_i$, subject to (7.4), (7.6) and (7.8), without regard to the energy availability constraint (7.5), to satisfy:*

$$\max: \ \phi = V \sum_{n,c} U_n^{(c)}(r^{nc}) \tag{7.10}$$

$$s.t. \quad r^{nc} + \sum_{s_i} \pi_{s_i} \sum_{k=1}^{K} \varrho_k^{(s_i)} \sum_{a \in \mathcal{N}_n^{(in)}} \mu_{[a,n]}^{(c)}(s_i, \boldsymbol{P}_k^{(s_i)}) \leq \sum_{s_i} \pi_{s_i} \sum_{k=1}^{K} \varrho_k^{(s_i)} \sum_{b \in \mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)}(s_i, \boldsymbol{P}_k^{(s_i)}), \forall \ (n,c), \tag{7.11}$$

$$\sum_{s_i} \pi_{s_i} \sum_{k=1}^{K} \varrho_k^{(s_i)} \sum_{b \in \mathcal{N}_n^{(o)}} P_{k,[n,b]}^{(s_i)} = \sum_{\boldsymbol{h}_j} \pi_{\boldsymbol{h}_j} \sum_{k=1}^{K} \varphi_k^{(\boldsymbol{h}_j)} e_{n,k}^{(\boldsymbol{h}_j)}, \forall n, \tag{7.12}$$

$$\boldsymbol{P}_k^{(s_i)} \in \mathcal{P}^{(s_i)}, 0 \leq \varrho_k^{(s_i)}, \varphi_k^{(\boldsymbol{h}_j)} \leq 1, \forall s_i, k, \boldsymbol{h}_j,$$

$$\sum_{k=1}^{K} \varrho_k^{(s_i)} = 1, \sum_{k=1}^{K} \varphi_k^{(\boldsymbol{h}_j)} = 1, \forall s_i, \boldsymbol{h}_j,$$

$$0 \leq r^{nc} \leq R_{max}, \forall\, (n,c), 0 \leq e_{n,k}^{(\boldsymbol{h}_j)} \leq h_n^{(\boldsymbol{h}_j)}, \ \forall\ n, k, \boldsymbol{h}_j.$$

Here π_{s_i} and $\pi_{\boldsymbol{h}_j}$ are the marginal distribution of the random channel state s_i and energy state $\boldsymbol{h}_j$, and $K = N^2 + N + 2$. [8]

[8] The number K is due to the use of Caratheodory's Theorem in the proof argument used in Chapter 6.

Proof. The proof argument is essentially the same as the one used in proving Theorem 23 in Chapter 6. Hence we omit it here. □

In the theorem, (7.11) says that the rate of incoming data to node n is no more than the transmission rate out, and the equality constraint (7.12) says that the rate of harvested energy is equal to the energy consumption rate. We note that *Theorem 25 indeed holds under more general ergodic* $\boldsymbol{S}(t)$ *and* $\boldsymbol{h}(t)$ *processes*, e.g., when $\boldsymbol{S}(t)$ and $\boldsymbol{h}(t)$ evolve according to some finite state irreducible and aperiodic Markov chains.

7.4 Engineering the queues

In this section, we present our Energy-limited Scheduling Algorithm (ESA) for the UMEM problem. ESA is designed based on the Lyapunov optimization technique developed in Chapter 6 and [GNT06]. The idea of ESA is to construct a Lyapunov scheduling algorithm with *perturbed* weights for determining the energy harvesting, power allocation, routing and scheduling decisions. We show that, by carefully perturbing the weights, one can ensure that whenever we allocate power to the links, there is always enough energy in the energy queues.

7.4.1 The ESA Algorithm

To start, we first choose a *perturbation* vector $\boldsymbol{\theta} = (\theta_n, n \in \mathcal{N})$ (to be specified later). We then define a *perturbed* Lyapunov function as follows:

$$L(t) \triangleq \frac{1}{2} \sum_{n,c \in \mathcal{N}} \left[Q_n^{(c)}(t)\right]^2 + \frac{1}{2} \sum_{n \in \mathcal{N}} \left[E_n(t) - \theta_n\right]^2. \tag{7.13}$$

The intuition behind the use of the $\boldsymbol{\theta}$ vector is that by keeping the Lyapunov function value small, we indeed "push" the $E_n(t)$ value towards θ_n. Thus by carefully choosing the value of θ_n, we can ensure that the energy queues always have enough energy for transmission. This is shown in Fig. 7.1.

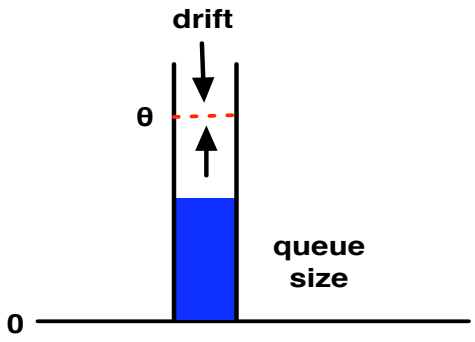

Figure 7.1: The intuition behind perturbation

Now denote $\boldsymbol{Z}(t) = (\boldsymbol{Q}(t), \boldsymbol{E}(t))$, and define a one-slot conditional Lyapunov drift as follows:

$$\Delta(t) \triangleq \mathbb{E}\big\{L(t+1) - L(t) \mid \boldsymbol{Z}(t)\big\}. \tag{7.14}$$

Here the expectation is taken over the randomness of the channel state and the energy state, as well as the randomness in choosing the data admission action, the power allocation action, the routing and scheduling action, and the energy harvesting action. For notational simplicity, we also define:

$$\Delta_V(t) \triangleq \Delta(t) - V\mathbb{E}\big\{\sum_{n,c} U_n^{(c)}(R_n^{(c)}(t)) \mid \boldsymbol{Z}(t)\big\}. \tag{7.15}$$

We have the following lemma regarding the drift:

Lemma 13. *Under any data admission action, power allocation action, routing and scheduling action, and energy harvesting action, which satisfies the energy availability constraint (7.5), that can be implemented at time t, we have:*

$$\begin{aligned}\Delta_V(t) \leq B &+ \sum_{n\in\mathcal{N}}(E_n(t)-\theta_n)\mathbb{E}\big\{e_n(t)\mid \boldsymbol{Z}(t)\big\} \\ &-\mathbb{E}\Big\{\sum_{n,c}\big[VU_n^{(c)}(R_n^{(c)}(t))-Q_n^{(c)}(t)R_n^{(c)}(t)\big]\mid \boldsymbol{Z}(t)\Big\} \\ &-\mathbb{E}\Big\{\sum_n\Big[\sum_c\sum_{b\in\mathcal{N}_n^{(o)}}\mu_{[n,b]}^{(c)}(t)\big[Q_n^{(c)}(t)-Q_b^{(c)}(t)\big] \\ &\qquad +(E_n(t)-\theta_n)\sum_{b\in\mathcal{N}_n^{(o)}}P_{[n,b]}(t)\Big]\mid \boldsymbol{Z}(t)\Big\}.\end{aligned} \tag{7.16}$$

Here $B = N^2(\frac{3}{2}d_{max}^2\mu_{max}^2 + R_{max}^2) + \frac{N}{2}(P_{max}+h_{max})^2$, and d_{max} is defined in Section 7.1 as the maximum in-degree or out-degree of any node in the network.

Proof. See Section 7.9.1. □

We now present the ESA algorithm. The idea of the algorithm is to approximately minimize the right-hand side (RHS) of (7.16) subject to the energy-availability constraint (7.5). In ESA, we use a parameter $\gamma \triangleq R_{max}+d_{max}\mu_{max}$, which is used in the link weight definition to allow deterministic upper bounds on queue sizes.

Energy-limited Scheduling Algorithm (ESA): Initialize $\boldsymbol{\theta}$. At every slot t, observe $\boldsymbol{Q}(t)$, $\boldsymbol{E}(t)$, $\boldsymbol{S}(t)$, and do:

- Energy Harvesting: If $E_n(t)-\theta_n < 0$, perform energy harvesting and store the harvested energy, i.e., $e_n(t)=h_n(t)$. Else set $e_n(t)=0$. Note that this decision on $e_n(t)$ indeed minimizes the $(E_n(t)-\theta_n)\mathbb{E}\big\{e_n(t)\mid \boldsymbol{Z}(t)\big\}$ term in (7.16).

- Data Admission: Choose $R_n^{(c)}(t)$ to be the optimal solution of the following optimization problem:

$$\max: \ VU_n^{(c)}(r) - Q_n^{(c)}(t)r, \ \text{ s.t. } 0 \leq r \leq R_{max}. \tag{7.17}$$

Note that this decision minimizes the terms involving $R_n^{(c)}(t)$ in the RHS of (7.16).

- Power Allocation: Define the weight of the commodity c data over link $[n,b]$ as:

$$W_{[n,b]}^{(c)}(t) \triangleq \left[Q_n^{(c)}(t) - Q_b^{(c)}(t) - \gamma\right]^+. \tag{7.18}$$

Then define the link weight $W_{[n,b]}(t) = \max_c W_{[n,b]}^{(c)}(t)$, and choose $\boldsymbol{P}(t) \in \mathcal{P}^{(s_i)}$ to maximize:

$$G(\boldsymbol{P}(t)) \triangleq \sum_n \left[\sum_{b \in \mathcal{N}_n^{(o)}} \mu_{[n,b]}(t) W_{[n,b]}(t) + (E_n(t) - \theta_n) \sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}(t) \right], \tag{7.19}$$

subject to the energy availability constraint (7.5).

- Routing and Scheduling: For every node n, find any $c^* \in \arg\max_c W_{[n,b]}^{(c)}(t)$. If $W_{[n,b]}^{(c^*)}(t) > 0$, set:

$$\mu_{[n,b]}^{(c^*)}(t) = \mu_{[n,b]}(t), \tag{7.20}$$

that is, allocate the full rate over the link $[n,b]$ to any commodity that achieves the maximum positive weight over the link. Use idle-fill if needed. If $W_{[n,b]}^{(c^*)}(t) = 0$, set $\mu_{[n,b]}^{(c)}(t) = 0$ for all c over link $[n,b]$. [9]

- Queue Update: Update $Q_n^{(c)}(t)$ and $E_n(t)$ according to the dynamics (7.6) and (7.8), respectively.

[9] Note that we still use the same power allocation $P_{[n,b]}(t)$ (can be nonzero) in the case when $W_{[n,b]}^{(c^*)}(t) = 0$, although all the rates $\mu_{[n,b]}^{(c)}(t)$ are zero. We show that doing this still yields performance that can be pushed arbitrarily close to optimal. In the actual implementation, however, we can always save the power $P_{[n,b]}(t)$ when $\mu_{[n,b]}^{(c)}(t) = 0 \ \forall c$. Similar performance results can also be obtained.

The combined Power Allocation and Routing and Scheduling step would have minimized the terms involving $\mu^{(c)}_{[n,b]}(t)$ and $\boldsymbol{P}(t)$ in the RHS of (7.16) if we had defined $\gamma = 0$. However, we have included a non-zero γ in the differential backlog definition (7.18), resulting in a decision that comes within an *additive constant* of minimizing the RHS of (7.16). The advantage of using this γ is that it leads to a deterministic bound on all queue sizes, as we show in the next section.

In the energy harvesting step of ESA, node n performs energy harvesting only when the energy volume is less than θ_n, and hence $E_n(t) \leq \theta_n + h_{max}$ for all t. This feature is very important because it allows one to implement ESA with finite energy storage capacity. More importantly, we show in later sections that it provides us with a very easy way to size our energy storage devices if we want to achieve a utility that is within $O(\epsilon)$ of the optimal: use energy storage devices of size $O(1/\epsilon)$. In practice, once the energy storage capacity is determined, we can always modify ESA by having the nodes perform energy harvesting in every time slot, in which case nodes always have more energy than that under ESA, and the same utility performance can be achieved.

7.4.2 Implementation of ESA

(I) First we note that ESA only requires the knowledge of the *instant* channel state $\boldsymbol{S}(t)$, the queue sizes $\boldsymbol{Q}(t)$ and $\boldsymbol{E}(t)$. *It does not even require any knowledge of the energy state process* $\boldsymbol{h}(t)$. This is very useful in practice when the knowledge of the energy source is difficult to obtain. ESA is also very different from previous algorithms for energy harvesting networks, e.g., [SMJG10] [KHZS07], where sufficient statistical knowledge of the energy source is often required.

(II) Note that the implementation of ESA involves maximizing (7.19). Thus ESA's complexity is the same as the widely used Max-Weight algorithms, which in general requires centralized control and can be NP-hard [GNT06]. However, in cases when the links do not interfere with each other, ESA can easily be implemented in a distributed manner, where each node only has to know about the queue sizes at its neighbor nodes and can decide on the power allocation locally. Moreover, one can look for constant factor approximation solutions of (7.19), e.g., [LS06] and Sections 4.7 and 5.2.1 in [GNT06]. Such approximation results can usually be found in a distributed manner in polynomial time, and ESA can be shown to achieve a utility that is at least a constant factor of $U_{tot}(\boldsymbol{r}^*)$ under these solutions.

7.5 Performance analysis

We now present the performance results of the ESA algorithm. In the following, we first present the results under i.i.d. network randomness and give its proof in Section 7.9. We later extend the performance results of ESA to the case when the network randomness is Markovian. Below, the parameter β is the largest first derivative of the utility functions defined in (7.1), and the parameter θ_n is defined

$$\theta_n \triangleq \delta\beta V + P_{max}. \tag{7.21}$$

7.5.1 ESA under I.I.D. randomness

Theorem 26. *Under the ESA algorithm with β and θ_n defined in (7.1) and (7.21), we have the following:*

(a) The data queues and the energy queues satisfy the following for all time steps under any arbitrary $\boldsymbol{S}(t)$ *and* $\boldsymbol{h}(t)$ *processes:*

$$0 \leq Q_n^{(c)}(t) \leq \beta V + R_{max}, \ \forall\ (n,c), \tag{7.22}$$

$$0 \leq E_n(t) \leq \theta_n + h_{max}, \ \forall\ n. \tag{7.23}$$

Moreover, when a node n allocates nonzero power to any of its outgoing links, $E_n(t) \geq P_{max}$.

(b) Let $\overline{\boldsymbol{r}}(T) = (\overline{r}^{nc}(T), \forall\,(n,c))$ be the time average admitted rate vector achieved by ESA up to time T, i.e., $\overline{r}^{nc}(T) = \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big\{R_n^{(c)}(t)\big\}$. Then:

$$\liminf_{T\to\infty} U_{tot}(\overline{\boldsymbol{r}}(T)) = \liminf_{T\to\infty}\sum_{n,c} U_n^{(c)}(\overline{r}^{nc}(T)) \geq U_{tot}(\boldsymbol{r}^*) - \frac{\tilde{B}}{V}, \tag{7.24}$$

where $\boldsymbol{r}^*$ is an optimal solution of the UMEM problem, and $\tilde{B} = B + N^2\gamma d_{max}\mu_{max}$, which is $\Theta(1)$, i.e., independent of V.

Proof. See Section 7.9.2. □

We note the following of Theorem 26:

(I) Part (a) is a *sample path* result. Hence, it holds even under *non-stationary* $\boldsymbol{S}(t)$ and $\boldsymbol{h}(t)$ processes.

(II) By taking $\epsilon = 1/V$, Part (a) implies that the average data queue size is $O(1/\epsilon)$. Combining this with Part (b), we see that ESA achieves an $[O(\epsilon), O(1/\epsilon)]$ utility-backlog tradeoff for the UMEM problem.

(III) We see from Part (a) that the energy queue size is deterministically upper bounded by a constant of size $O(1/\epsilon)$. This provides an explicit characterization of the size

of the energy storage device needed for achieving the desired utility performance. Such explicit bounds are particularly useful for practical system deployments.

(IV) Note that we prove a utility performance bound, i.e., (7.24), that is slightly different from the objective of $U_{tot}(\lim_{T\to\infty}\overline{\boldsymbol{r}}(T))$. The reason is that the limit $\lim_{T\to\infty}\overline{\boldsymbol{r}}(T)$ may not exist. However, whenever the limit does exist, we can replace lim inf with the regular limit and push the limit inside the summation. Then, (7.24) becomes $U_{tot}(\lim_{T\to\infty}\overline{\boldsymbol{r}}(T)) \geq U_{tot}(\boldsymbol{r}^*) - \frac{\tilde{B}}{V}$.

7.5.2 ESA under Markovian randomness

We now extend our results to the more general setting where the channel state $\boldsymbol{S}(t)$ and the energy state $\boldsymbol{h}(t)$ both evolve according to some finite state irreducible and aperiodic Markov chains. In this case π_{s_i} and $\pi_{\boldsymbol{h}_i}$ represent the steady state probability of the events $\{\boldsymbol{S}(t) = s_i\}$ and $\{\boldsymbol{h}(t) = \boldsymbol{h}_i\}$, respectively. In this case, the performance results of ESA are summarized in the following theorem:

Theorem 27. *Suppose that $[\boldsymbol{S}(t), \boldsymbol{h}(t)]$ evolves according to some finite state irreducible and aperiodic Markov chain. Then under ESA, we have: (a) the bounds (7.22) and (7.23) still hold; (b) the average utility is within $O(1/V)$ of $U_{tot}(\boldsymbol{r}^*)$, i.e., $\liminf_{T\to\infty} U_{tot}(\overline{\boldsymbol{r}}(T)) = \liminf_{T\to\infty}\sum_{n,c} U_n^{(c)}(\overline{r}^{nc}(T)) \geq U_{tot}(\boldsymbol{r}^*) - O(1/V)$.*

Proof. Part (a) follows from Theorem 26, since (7.22) and (7.23) are sample-path results. The utility performance follows from Section 4.10.5 in Chapter 4.

□

7.6 Reducing the buffer size

In this section, we show that it is possible to achieve the same $O(\epsilon)$ close-to-optimal utility performance guarantee using energy storage devices of only $O([\log(1/\epsilon)]^2)$ sizes, while guaranteeing a much smaller average data queue size, i.e., $O([\log(1/\epsilon)]^2)$. Our algorithm is motivated by the "exponential attraction" result developed in Chapter 4, which states that the probability for the network backlog vector to deviate from some fixed point typically decreases exponentially with the deviation distance. This suggests that most of the queue backlogs are kept in the queues to maintain a "proper" queue vector value to base the decisions on. If we can somehow learn the value of this vector, then we can "subtract out" a large amount of data and energy backlog from the network and reduce the required buffer sizes. Below, we present the Modified-ESA (MESA) algorithm to achieve this goal.

7.6.1 The Modified-ESA algorithm

To start, for a given ϵ, we let $V = 1/\epsilon$, and define $M = 4[\log(V)]^2$. We then associate with each node n a *virtual* energy queue process $\hat{E}_n(t)$ and a set of *virtual* data queues $\hat{Q}_n^{(c)}(t)$, $\forall\, c$. We also associate with each node n an *actual* energy queue with size M. We assume that V is chosen to be such that $\frac{M}{2} > \alpha_{max} \triangleq \max[P_{max}, h_{max}]$. MESA consists of two phases: Phase I runs the system using the virtual queue processes, to discover the "attraction point" values of the queues (as explained below). Phase II then uses these values to carefully perform the actions so as to ensure energy availability and reduce network delay. We emphasize that, although MESA looks similar to the FQLA algorithms

developed in Chapter 4, it only uses *finite* energy storage capacities. This feature makes it very different from FQLA and requires a new analysis for its performance.

Modified-ESA (MESA): Initialize $\boldsymbol{\theta}$. Perform:

- Phase I: Choose a sufficiently large T. From time $t = 0, ..., T$, run ESA using $\hat{\boldsymbol{Q}}(t)$ and $\hat{\boldsymbol{E}}(t)$ as the data and energy queues. Obtain two vectors $\boldsymbol{\mathcal{Q}} = (\mathcal{Q}_n^{(c)}, \forall\,(n,c))$ and $\boldsymbol{\mathcal{E}} = (\mathcal{E}_n, \forall\, n)$ with $\mathcal{Q}_n^{(c)} = [\hat{Q}_n^{(c)}(T) - \frac{M}{2}]^+$ and $\mathcal{E}_n = [\hat{E}_n(T) - \frac{M}{2}]^+$.

- Phase II: Reset $t = 0$. Initialize $\hat{\boldsymbol{E}}(0) = \boldsymbol{\mathcal{E}}$ and $\hat{\boldsymbol{Q}}(0) = \boldsymbol{\mathcal{Q}}$. Also set $\boldsymbol{Q}(0) = \mathbf{0}$ and $\boldsymbol{E}(0) = \mathbf{0}$. In every time slot, first run the ESA algorithm based on $\hat{\boldsymbol{Q}}(t)$, $\hat{\boldsymbol{E}}(t)$, and $\boldsymbol{S}(t)$, to obtain the action variables, i.e., the corresponding $e_n(t)$, $R_n^{(c)}(t)$, and $\mu_{[n,b]}^{(c)}(t)$ values. Perform Data Admisson, Power Allocation, and Routing and Scheduling exactly as ESA, plus the following:

  - Energy harvesting: If $\hat{E}_n(t) < \mathcal{E}_n$, let $\tilde{e}_n(t) = [e_n(t) - (\mathcal{E}_n - \hat{E}_n(t))]^+$. Harvest $\tilde{e}(t)$ amount of energy, i.e., update $E_n(t)$ as follows:

    $$E_n(t+1) = \min\big[[E_n(t) - \sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}(t)]^+ + \tilde{e}_n(t), M\big].$$

    Else if $\hat{E}_n(t) > \mathcal{E}_n + M$, do not spend any power and update $E_n(t)$ according to:

    $$E_n(t+1) = \min\big[E_n(t) + e_n(t), M\big].$$

    Else update $E_n(t)$ according to:

    $$E_n(t+1) = \min\big[[E_n(t) - \sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}(t)]^+ + e_n(t), M\big].$$

  - Packet Dropping: For any node n with $\hat{E}_n(t) < \mathcal{E}_n + P_{max}$ or $\hat{E}_n(t) > \mathcal{E}_n + M$, drop all the packets that should have been transmitted, i.e., change the input

into any $Q_n^{(c)}(t)$ to (use idle fill whenever a node does not have enough data to send):

$$A_n^{(c)}(t) = R_n^{(c)}(t) + \sum_{a \in \mathcal{N}_n^{(in)}} \mu_{[a,n]}^{(c)}(t) 1_{[F_a(t)]}.$$

Here $1_{[\cdot]}$ is the indicator function and $F_a(t)$ is the event that $\hat{E}_a(t) \in [\mathcal{E}_a + P_{max}, \mathcal{E}_a + M]$. Then further modify the routing and scheduling action under ESA as follows:

* If $\hat{Q}_n^{(c)}(t) < \mathcal{Q}_n^{(c)}$, let $\tilde{A}_n^{(c)}(t) = \left[A_n^{(c)}(t) - [\mathcal{Q}_n^{(c)} - \hat{Q}_n^{(c)}(t)]\right]^+$, update $Q_n^{(c)}(t)$ by:

$$Q_n^{(c)}(t+1) \leq \left[Q_n^{(c)}(t) - \sum_{b \in \mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)}(t)\right]^+ + \tilde{A}_n^{(c)}(t).$$

* If $\hat{Q}_n^{(c)}(t) \geq \mathcal{Q}_n^{(c)}$, update $Q_n^{(c)}(t)$ by:

$$Q_n^{(c)}(t+1) \leq \left[Q_n^{(c)}(t) - \sum_{b \in \mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)}(t)\right]^+ + A_n^{(c)}(t).$$

– Update $\hat{\boldsymbol{E}}(t)$ and $\hat{\boldsymbol{Q}}(t)$ using (7.8) and (7.6).

Note here we have used the $[\cdot]^+$ operator for updating $E_n(t)$ in the energy harvesting part. This is due to the fact that the power allocation decisions are now made *based on* $\hat{\boldsymbol{E}}(t)$ *but not* $\boldsymbol{E}(t)$. If $\hat{E}_n(t)$ never gets below $\mathcal{E}_n$ or above $\mathcal{E}_n + M$, then we always have $E_n(t) = \hat{E}_n(t) - \mathcal{E}_n$. Similarly, if $\hat{Q}_n^{(c)}(t)$ is always above $\mathcal{Q}_n^{(c)}$ and $\hat{E}_n(t)$ is always in $[\mathcal{E}_n + P_{max}, \mathcal{E}_n + M]$, then we always have $Q_n^{(c)}(t) = \hat{Q}_n^{(c)}(t) - \mathcal{Q}_n^{(c)}$. MESA is designed to ensure that $\hat{Q}_n^{(c)}(t)$ and $\hat{E}_n(t)$ mostly stay in these "right" ranges. We see in the following lemma that, although $\hat{Q}_n^{(c)}(t)$ and $\hat{E}_n(t)$ can go out of the ranges, our algorithm ensures that the queue processes are in fact close to each other.

Lemma 14. *For all time steps t, we have the following:*

$$0 \leq Q_n^{(c)}(t) \leq [\hat{Q}_n^{(c)}(t) - \mathcal{Q}_n^{(c)}]^+ + \gamma, \ \ \forall\, (n,c), \tag{7.25}$$

$$\min\big[[\hat{E}_n(t) - \mathcal{E}_n]^+, M\big] \leq E_n(t), \ \ \forall\, n. \tag{7.26}$$

Proof. See Section 7.9.3. □

By Lemma 14, when $\hat{E}_n(t) \in [\mathcal{E}_n + P_{max}, \mathcal{E}_n + M]$, we have $E_n(t) \geq [\hat{E}_n(t) - \mathcal{E}_n]^+ \geq P_{max}$. Thus all the power allocations are valid under MESA, i.e., under MESA, although the power allocation decision is made based on $\hat{\boldsymbol{E}}(t)$, the energy availability constraint is still ensured for all time.

7.6.2 Performance of MESA

To study the performance of MESA, we first denote by $g(\boldsymbol{\upsilon}, \boldsymbol{\nu})$ the dual function of the problem (7.10). The following lemma shows that the dual function can be written in a form that is without the variables $\varrho_k^{(s_i)}$ and $\varphi_k^{(\boldsymbol{h}_i)}$. This fact greatly simplifies the evaluation of the dual function.

Lemma 15. *The dual problem of (7.10) is given by:*

$$\min: \ \ g(\boldsymbol{\upsilon}, \boldsymbol{\nu}), \quad s.t. \ \ \boldsymbol{\upsilon} \succeq \mathbf{0}, \boldsymbol{\nu} \in \mathbb{R}^N, \tag{7.27}$$

where $\boldsymbol{\upsilon} = (\upsilon_n^{(c)}, \forall\, (n,c))$, $\boldsymbol{\nu} = (\nu_n, \forall\, n)$, and $g(\boldsymbol{\upsilon}, \boldsymbol{\nu})$ is the dual function defined by

$$\begin{aligned} g(\boldsymbol{\upsilon}, \boldsymbol{\nu}) = \sup_{r^{nc}, \boldsymbol{P}^{(s_i)}, e_n^{(\boldsymbol{h}_j)}} & \sum_{s_i} \pi_{s_i} \sum_{\boldsymbol{h}_j} \pi_{\boldsymbol{h}_j} \Big\{ V \sum_{n,c} U_n^{(c)}(r^{nc}) \\ & - \sum_{n,c} \upsilon_n^{(c)} \big[r^{nc} + \sum_{a \in \mathcal{N}_n^{(in)}} \mu_{[a,n]}^{(c)}(s_i, \boldsymbol{P}^{(s_i)}) \\ & - \sum_{b \in \mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)}(s_i, \boldsymbol{P}^{(s_i)}) \big] - \sum_n \nu_n \big[\sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}^{(s_i)} - e_n^{(\boldsymbol{h}_j)} \big] \Big\}. \end{aligned} \tag{7.28}$$

Proof. The proof is the same as that of Lemma 8 in Chapter 6. □

We now summarize the performance results of MESA in the following theorem. In the theorem, we denote $\boldsymbol{y} = (\boldsymbol{\upsilon}, \boldsymbol{\nu})$, and write $g(\boldsymbol{\upsilon}, \boldsymbol{\nu})$ as a function of $\boldsymbol{y}$.

Theorem 28. *Suppose that $\boldsymbol{y}^* = (\boldsymbol{\upsilon}^*, \boldsymbol{\nu}^*)$ is finite and unique, that $\boldsymbol{\theta}$ is chosen such that $\theta_n + \nu_n^* > 0$, $\forall\ n$, and that for all $\boldsymbol{y} = (\boldsymbol{\upsilon}, \boldsymbol{\nu})$ with $\boldsymbol{\upsilon} \succeq \mathbf{0}, \boldsymbol{\nu} \in \mathbb{R}^N$, the dual function $g(\boldsymbol{y})$ satisfies:*

$$g(\boldsymbol{y}^*) \geq g(\boldsymbol{y}) + L||\boldsymbol{y}^* - \boldsymbol{y}||, \tag{7.29}$$

for some constant $L > 0$ independent of V, that the system is in steady state at time T, and that a steady state distribution for the queues exists under ESA. Then under MESA with a sufficiently large V, with probability $1 - O(\frac{1}{V^4})$, we have:

$$\overline{\boldsymbol{Q}} \leq O([\log(V)]^2), \tag{7.30}$$

$$\liminf_{T\to\infty} U_{tot}(\overline{\boldsymbol{r}}(T)) \geq U_{tot}(\boldsymbol{r}^*) - O(1/V), \tag{7.31}$$

where $U_{tot}(\overline{\boldsymbol{r}}(T))$ is defined in Theorem 26. Furthermore, the fraction of packets dropped in the packet dropping step is $O(\frac{1}{V^{3\log(V)/2}})$.

Proof. See Section 7.9.4. □

Note that the theorem also holds when $[\boldsymbol{S}(t), \boldsymbol{h}(t)]$ are Markovian as in Theorem 27. The condition (7.29) is indeed the condition needed for proving the exponential attraction result (Theorem 5) in Chapter 4. It has been observed that (7.29) typically holds in practice, particularly when the network action set is finite, in which case the dual function $g(\boldsymbol{y})$ is polyhedral in $\boldsymbol{y}$ (see Section 4.2.3 in Chapter 4 for more discussion). It has been shown that in this case, the queue backlog vector pair is "exponentially attracted" to the fixed point $(\boldsymbol{\upsilon}^*, \boldsymbol{\nu}^* + \boldsymbol{\theta}) = \Theta(V)$, in that the probability of deviating decreases exponentially with the deviation distance. Therefore, the probability of deviating by

some $\Theta([\log(V)]^2)$ distance is $1/V^{\log(V)}$, which is very small when V is large. Theorem 28 then shows that under this condition, one can significantly reduce the energy capacity needed to achieve the $O(\epsilon)$ close-to-optimal utility performance and greatly reduce the network congestion.

7.7 Simulation

In this section, we provide simulation results of our algorithms. We consider a data collection network shown in Fig. 7.2. Such a network typically appears in the sensor network scenario where sensors are used to sense data and forward them to the sink. In this network, there are 6 nodes. The node S represents the sink node, the nodes $1, 2, 3$ sense data and deliver them to node S via the relay of nodes $4, 5$.

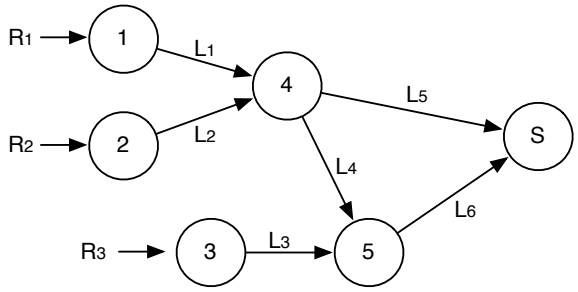

Figure 7.2: A data collection network.

The channel state of each communication link, represented by a directed edge, is i.i.d. every time slot and can be either "G=Good" or "B=Bad" with equal probability. One unit of power can serve two packets over a link when the channel state is good, but can only serve one when the channel is bad. We assume that $R_{max} = 3$ and the utility functions are given by: $U_1^{(S)}(r) = U_2^{(S)}(r) = U_3^{(S)}(r) = \log(1 + r)$ and $U_4^{(S)}(r) = U_5^{(S)}(r) = 0$. For simplicity, we also assume that all the links do not interfere with each other. We

assume that for each node, the available energy $h_n(t)$ is i.i.d. and $h_n(t) = 2/0$ with equal probability.

It is easy to see that in this case, we can use $\beta = 1$, $\delta = 2$, $\mu_{max} = 2$, $d_{max} = 2$, $P_{max} = 2$, and $\gamma = d_{max}\mu_{max} + R_{max} = 7$. Using Theorem 26, we set $\theta_n = \delta\beta V + P_{max} = 2V + 2$. We simulate $V \in \{20, 30, 40, 50, 80, 100, 200\}$. Each simulation is run for 10^6 slots. The simulation results are plotted in Fig. 7.3. We see that the total network utility converges quickly to very close to the optimal value, which can be shown to be roughly 2.03, and that the average data queue size and the average energy queue size both grow linearly in V.

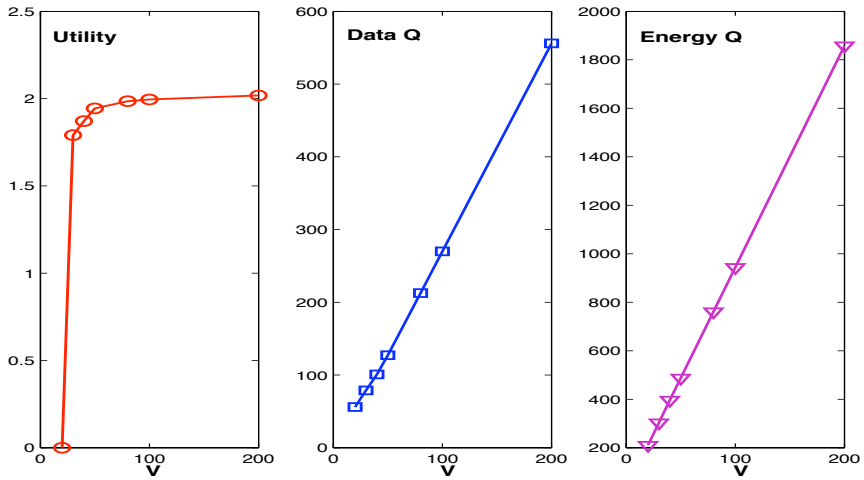

Figure 7.3: Simulation results of ESA.

Fig. 7.4 also shows two sample-path data queue processes and two energy queue processes under $V = 100$. It can be verified that all the queue sizes satisfy the queueing bounds in Theorem 26. We also observe the "exponential attraction" behavior of the queues, as shown in Chapter 4. However, different from the simulation results in Chapter 4, we see that the queue size of $Q_1^{(S)}(t)$ does not approach the fixed point from below. It instead first has a "burst" in the early time slots. This is due to the fact that the system

"waits" for $E_1(t)$ to come close enough to its fixed point. Such an effect can be mitigated by storing an initial energy of size θ in the energy queue.

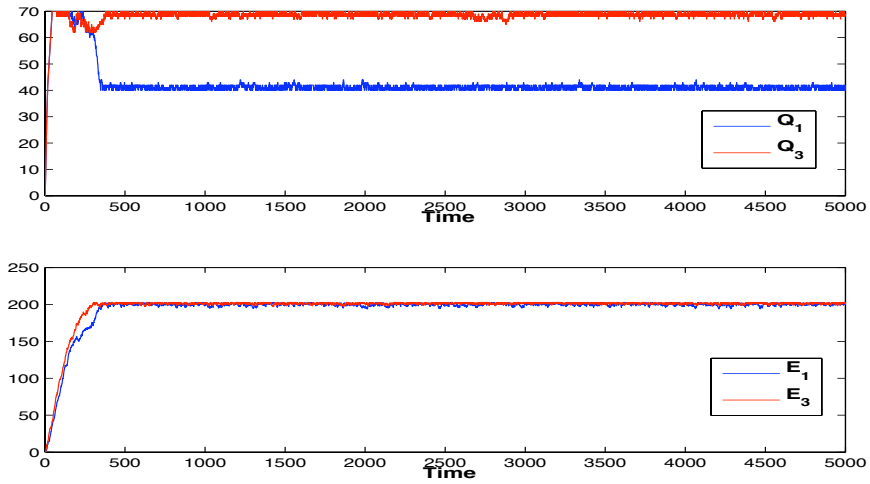

Figure 7.4: Sample path queue processes.

We also simulate the MESA algorithm for the same network with the same $\boldsymbol{\theta}$ vector. We use $T = 50V$ in Phase I for obtaining the vectors $\boldsymbol{\mathcal{E}}$ and $\boldsymbol{\mathcal{Q}}$. Fig. 7.5 plots the performance results. We observe that no packet is dropped throughout the simulations under any V values. The utility again quickly converges to the optimal as V increases. We also see from the second and third plots that the actual queues only grow poly-logarithmically in V, i.e., $O([\log(V)]^2)$, while the virtual queues, which are the same as the actual queues under ESA, grows linearly in V. This shows a good match between the simulations and Theorem 28.

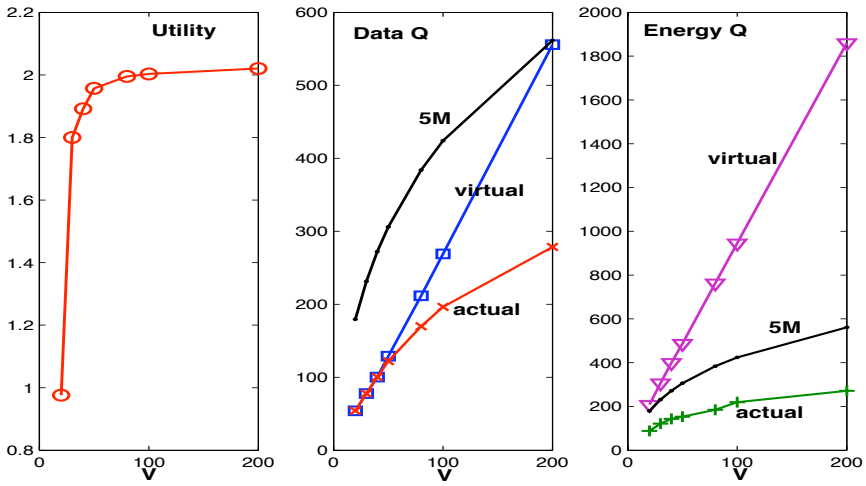

Figure 7.5: Simulation results of MESA. $5M$ is the total network energy buffer size.

7.8 Chapter summary

In this chapter, we use the perturbation technique in Chapter 6 to develop the Energy-limited Scheduling Algorithm (ESA) for achieving optimal utility in general energy harvesting networks equipped with finite capacity energy storage device. ESA is an online algorithm and does not require any knowledge of the harvestable energy processes. We show that ESA achieves an average utility that is within $O(\epsilon)$ of the optimal for any $\epsilon > 0$ using energy storage devices of size $O(1/\epsilon)$, while guaranteeing that the time average network congestion is $O(1/\epsilon)$. We then also develop the Modified-ESA algorithm (MESA), and show that MESA can achieve the same $O(\epsilon)$ utility performance using energy storage devices of only size $O([\log(\frac{1}{\epsilon})]^2)$.

7.9 Proofs of the chapter

7.9.1 Proof of Lemma 13

Here we prove Lemma 13.

Proof. First by squaring both sides of (7.6), and using the fact that for any $x \in \mathbb{R}$, $([x]^+)^2 \leq x^2$, we have:

$$[Q_n^{(c)}(t+1)]^2 - [Q_n^{(c)}(t)]^2 \leq [\sum_{b\in\mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)}(t)]^2 + [\sum_{a\in\mathcal{N}_n^{(in)}} \mu_{[a,n]}^{(c)}(t) + R_n^{(c)}(t)]^2 \tag{7.32}$$
$$-2Q_n^{(c)}(t)\big[\sum_{b\in\mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)}(t) - \sum_{a\in\mathcal{N}_n^{(in)}} \mu_{[a,n]}^{(c)}(t) - R_n^{(c)}(t)\big].$$

Multiplying both sides by $\frac{1}{2}$, and defining $\hat{B} = \frac{3}{2}d_{max}^2\mu_{max}^2 + R_{max}^2$, [10] we have:

$$\frac{1}{2}\big([Q_n^{(c)}(t+1)]^2 - [Q_n^{(c)}(t)]^2\big) \leq \hat{B} \tag{7.33}$$
$$-Q_n^{(c)}(t)\big[\sum_{b\in\mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)}(t) - \sum_{a\in\mathcal{N}_n^{(in)}} \mu_{[a,n]}^{(c)}(t) - R_n^{(c)}(t)\big].$$

Using a similar approach, we get that:

$$\frac{1}{2}\big([E_n(t+1)-\theta_n]^2 - [E_n(t)-\theta_n]^2\big) \tag{7.34}$$
$$\leq \hat{B}' - [E_n(t)-\theta_n]\big[\sum_{b\in\mathcal{N}_n^{(o)}} P_{[n,b]}(t) - e_n(t)\big],$$

where $\hat{B}' = \frac{1}{2}(P_{max}+h_{max})^2$. Now by summing (7.33) over all (n,c) and (7.34) over all n, and by defining $B = N^2\hat{B} + N\hat{B}' = N^2(\frac{3}{2}d_{max}^2\mu_{max}^2 + R_{max}^2) + \frac{1}{2}N(P_{max}+h_{max})^2$, we have:

$$L(t+1) - L(t) \leq B - \sum_{n,c} Q_n^{(c)}(t)\big[\sum_{b\in\mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)}(t) - \sum_{a\in\mathcal{N}_n^{(in)}} \mu_{[a,n]}^{(c)}(t) - R_n^{(c)}(t)\big]$$
$$-\sum_n [E_n(t)-\theta_n]\big[\sum_{b\in\mathcal{N}_n^{(o)}} P_{[n,b]}(t) - e_n(t)\big].$$

Taking expectations on both sides over the random channel and energy states and the randomness over actions conditioning on $\boldsymbol{Z}(t)$, subtracting from both sides the term $V\mathbb{E}\big\{\sum_{n,c} U_n^{(c)}(R_n^{(c)}(t)) \mid \boldsymbol{Z}(t)\big\}$, and rearranging the terms, we see that the lemma follows.

□

[10] This uses $\sum_{b\in\mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)}(t) \leq d_{max}\mu_{max}$ and $\sum_{a\in\mathcal{N}_n^{(in)}} \mu_{[a,n]}^{(c)}(t) + R_n^{(c)}(t) \leq d_{max}\mu_{max} + R_{max}$.

7.9.2 Proof of Theorem 26

Here we prove Theorem 26.

Proof. (Part (a)) We first prove (7.22) using a similar argument as in [Nee06c]. It is easy to see that it holds for $t = 0$, since $Q_n^{(c)}(0) = 0$ for all (n, c). Now assume that $Q_n^{(c)}(t) \leq \beta V + R_{max}$ for all (n, c) at t, we want to show that it holds for time $t + 1$. First, if node n does not receive any new commodity c data, then $Q_n^{(c)}(t) \leq Q_n^{(c)}(t + 1) \leq \beta V + R_{max}$. Second, if node n receives endogenous commodity c data from any other node b, then we must have:

$$Q_n^{(c)}(t) \leq Q_b^{(c)}(t) - \gamma \leq \beta V + R_{max} - \gamma.$$

However, since any node can receive at most γ commodity c packets in every slot, we have $Q_n^{(c)}(t + 1) \leq \beta V + R_{max}$. Finally, if node n does not receives endogenous arrivals but receives exogenous packets from outside, then according to (7.17), we must have $Q_n^{(c)}(t) \leq \beta V$. Hence $Q_n^{(c)}(t + 1) \leq \beta V + R_{max}$.

Now it is also easy to see from the energy storage part of ESA that $E_n(t) \leq \theta_n + h_{max}$, which proves (7.23).

We now show that if $E_n(t) < P_{max}$, then $G(\boldsymbol{P}(t))$ in (7.19) will be maximized by choosing $P_{[n,m]}(t) = 0$ for all $m \in \mathcal{N}_n^{(o)}$ at node n. To see this, first note that since all the data queues are upper bounded by $\beta V + R_{max}$, we have: $W_{[n,b]}(t) \leq \left[\beta V - d_{max}\mu_{max}\right]^+$ for all $[n, b]$ and for all time.

Now let the power vector that maximizes $G(\boldsymbol{P}(t))$ be $\boldsymbol{P}^*$ and assume that there exists some $P^*_{[n,m]}$ that is positive. We now create a new power allocation vector $\boldsymbol{P}$ by

setting only $P^*_{[n,m]} = 0$ in $\boldsymbol{P}^*$. Then we have the following, in which we have written $\mu_{[n,m]}(\boldsymbol{S}(t), \boldsymbol{P}(t))$ only as a function of $\boldsymbol{P}(t)$ to simplify notation:

$$\begin{aligned} & G(\boldsymbol{P}^*) - G(\boldsymbol{P}) \\ &= \sum_n \sum_{b \in \mathcal{N}_n^{(o)}} \big[\mu_{[n,b]}(\boldsymbol{P}^*) - \mu_{[n,b]}(\boldsymbol{P})\big] W_{[n,b]}(t) + (E_n(t) - \theta_n) P^*_{[n,m]} \\ &\leq \big(\mu_{[n,m]}(\boldsymbol{P}^*) - \mu_{[n,m]}(\boldsymbol{P})\big) W_{[n,m]}(t) + (E_n(t) - \theta_n) P^*_{[n,m]}. \end{aligned}$$

Here in the last step we have used (7.3) in Property 2 of $\mu_{[n,m]}(\cdot, \boldsymbol{P})$, which implies that $\mu_{[n,b]}(\boldsymbol{P}^*) - \mu_{[n,b]}(\boldsymbol{P}) \leq 0$ for all $b \neq m$. Now suppose $E_n(t) < P_{max}$. We see then $E_n(t) - \theta_n < -\delta\beta V$. Using Property 1 and the fact that $W_{[n,m]}(t) \leq \big[\beta V - d_{max}\mu_{max}\big]^+$, the above implies:

$$G(\boldsymbol{P}^*) - G(\boldsymbol{P}) < \big[\beta V - d_{max}\mu_{max}\big]^+ \delta P^*_{[n,m]} - \delta\beta V P^*_{[n,m]} < 0.$$

This shows that $\boldsymbol{P}^*$ cannot have been the power vector that maximizes $G(t)$ if $E_n(t) < P_{max}$. Therefore $E_n(t) \geq P_{max}$ whenever node n allocates any nonzero power over any of its outgoing links. Hence all the power allocation decisions are feasible. This shows that the constraint (7.5) is indeed *redundant* in ESA and completes the proof of Part (a).

(Part (b)) We now prove Part (b). We first show that ESA approximately minimizes the RHS of (7.16). To see this, note from Part (A) that ESA indeed minimizes the following function at time t:

$$\begin{aligned} D(t) = & \sum_{n \in \mathcal{N}} (E_n(t) - \theta_n) e_n(t) \\ & - \sum_{n,c \in \mathcal{N}} \big[V U_n^{(c)}(R_n^{(c)}(t)) - Q_n^{(c)}(t) R_n^{(c)}(t)\big] \\ & - \sum_{n \in \mathcal{N}} \Big[\sum_c \sum_{b \in \mathcal{N}_n^{(o)}} \mu^{(c)}_{[n,b]}(t) \big[Q_n^{(c)}(t) - Q_b^{(c)}(t) - \gamma\big] + (E_n(t) - \theta_n) \sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}(t) \Big], \end{aligned} \tag{7.35}$$

subject to only the constraints: $e_n(t) \in [0, h_n(t)]$, $R_n^{(c)}(t) \in [0, R_{max}]$, $\boldsymbol{P}(t) \in \mathcal{P}^{(s_i)}$ and (7.4), i.e., without the energy-availability constraint (7.5). Now define $\tilde{D}(t)$ as follows:

$$\begin{aligned}\tilde{D}(t) = &\sum_{n\in\mathcal{N}} (E_n(t) - \theta_n) e_n(t) \\ &- \sum_{n,c\in\mathcal{N}} \big[V U_n^{(c)}(R_n^{(c)}(t)) - Q_n^{(c)}(t) R_n^{(c)}(t)\big] \\ &- \sum_{n\in\mathcal{N}} \Bigg[\sum_c \sum_{b\in\mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)}(t)\big[Q_n^{(c)}(t) - Q_b^{(c)}(t)\big] + (E_n(t) - \theta_n) \sum_{b\in\mathcal{N}_n^{(o)}} P_{[n,b]}(t)\Bigg].\end{aligned} \tag{7.36}$$

Note that $\tilde{D}(t)$ is indeed the function inside the expectation on the RHS of the drift bound (7.16). It is easy to see from the above that:

$$D(t) = \tilde{D}(t) + \sum_n \sum_c \sum_{[n,b]\in\mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)}(t)\gamma.$$

Since ESA minimizes $D(t)$, we see that:

$$\tilde{D}^E(t) + \sum_n \sum_c \sum_{b\in\mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)E}(t)\gamma \le \tilde{D}^{ALT}(t) + \sum_n \sum_c \sum_{b\in\mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)ALT}(t)\gamma,$$

where the superscript E represents the ESA algorithm, and ALT represents any other alternate policy. Since

$$0 \le \sum_n \sum_c \sum_{b\in\mathcal{N}_n^{(o)}} \mu_{[n,b]}^{(c)}(t)\gamma \le N^2\gamma d_{max}\mu_{max},$$

we have:

$$\tilde{D}^E(t) \le \tilde{D}^{ALT}(t) + N^2\gamma d_{max}\mu_{max}. \tag{7.37}$$

That is, the value of $\tilde{D}(t)$ under ESA is no greater than its value under *any* other alternative policy plus a constant, including the ones that ignore the energy availability constraint (7.5). Further, Part (a) shows that the energy availability constraint (7.5) is naturally satisfied under ESA without explicitly being enforced. Now using the definition of $\tilde{D}(t)$, (7.16) can be rewritten as:

$$\Delta(t) - V\mathbb{E}\big\{\sum_{n,c} U_n^{(c)}(R_n^{(c)}(t)) \mid \boldsymbol{Z}(t)\big\} \le B + \mathbb{E}\big\{\tilde{D}^E(t) \mid \boldsymbol{Z}(t)\big\}.$$

Using (7.37), we get:

$$\Delta(t) - V\mathbb{E}\big\{\sum_{n,c} U_n^{(c)}(R_n^{(c)}(t)) \mid \boldsymbol{Z}(t)\big\} \leq \tilde{B} + \mathbb{E}\big\{\tilde{D}^{ALT}(t) \mid \boldsymbol{Z}(t)\big\}, \tag{7.38}$$

where $\tilde{B} = B + N^2\gamma d_{max}\mu_{max}$. Plugging into (7.38) the policy in Theorem 25, which by comparing (7.10) and (7.36) can easily be shown to result in $\mathbb{E}\big\{\tilde{D}^{ALT}(t) \mid \boldsymbol{Z}(t)\big\} = \phi^*$, and using the fact that $\phi^* \geq VU_{tot}(\boldsymbol{r}^*)$, we have:

$$\Delta(t) - V\mathbb{E}\big\{\sum_{n,c} U_n^{(c)}(R_n^{(c)}(t)) \mid \boldsymbol{Z}(t)\big\} \leq \tilde{B} - VU_{tot}(\boldsymbol{r}^*).$$

Taking expectations over $\boldsymbol{Z}(t)$ and summing the above over $t = 0, ..., T-1$, we have:

$$\mathbb{E}\big\{L(T) - L(0)\big\} - V\sum_{t=0}^{T-1}\mathbb{E}\big\{\sum_{n,c} U_n^{(c)}(R_n^{(c)}(t))\big\} \leq T\tilde{B} - TVU_{tot}(\boldsymbol{r}^*).$$

Rearranging the terms, using the facts that $L(t) \geq 0$ and $L(0) = 0$, dividing both sides by VT, we get:

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big\{\sum_{n,c} U_n^{(c)}(R_n^{(c)}(t))\big\} \geq U_{tot}(\boldsymbol{r}^*) - \tilde{B}/V.$$

Using Jensen's inequality, we see that:

$$\sum_{n,c} U_n^{(c)}(\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big\{R_n^{(c)}(t)\big\}) \geq U_{tot}(\boldsymbol{r}^*) - \tilde{B}/V.$$

Taking a liminf as $T \to \infty$ and using the definition of $\overline{r}^{nc}(T)$, i.e., $\overline{r}^{nc}(T) = \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big\{R_n^{(c)}(t)\big\}$, we have

$$\liminf_{T\to\infty} \sum_{n,c} U_n^{(c)}(\overline{r}^{nc}(T)) \geq U_{tot}(\boldsymbol{r}^*) - \tilde{B}/V.$$

This completes the proof of Part (b). □

7.9.3 Proof of Lemma 14

Here we prove Lemma 14. We recall that $M = 4[\log(V)]^2$ is the size of the energy buffer.

Proof. We first prove (7.25). Define an intermediate process $\tilde{Q}_n^{(c)}(t)$ that evolves exactly as $Q_n^{(c)}(t)$ except that it does not discard packets when $\hat{E}_n(t) < \mathcal{E}_n + P_{max}$ or $\hat{E}_n(t) >$

$\mathcal{E}_n + M$. We see then $Q_n^{(c)}(t) \leq \tilde{Q}_n^{(c)}(t)$. Using Lemma 4 in Chapter 4, we see that: $\tilde{Q}_n^{(c)}(t) \leq [\hat{Q}_n^{(c)}(t) - \mathcal{Q}_n^{(c)}]^+ + \gamma$. Hence $Q_n^{(c)}(t) \leq [\hat{Q}_n^{(c)}(t) - \mathcal{Q}_n^{(c)}]^+ + \gamma$ and (7.25) follows. [11]

We now look at (7.26). First, it holds at time 0 since $0 = \hat{E}_n(0) - \mathcal{E}_n = E_n(0)$. Now suppose that it holds for $t = 0, 1, ..., k$. We want to show that it holds for $t = k+1$. First note that if $\hat{E}_n(k+1) \leq \mathcal{E}_n$, then (7.26) always holds because $E_n(k)$ is nonnegative for all k. Therefore, in the following we only consider the case when $\hat{E}_n(k+1) > \mathcal{E}_n$, i.e.,

$$[\hat{E}_n(k+1) - \mathcal{E}_n]^+ = \hat{E}_n(k+1) - \mathcal{E}_n. \tag{7.39}$$

Also note that since all the actions are made based on $\hat{\boldsymbol{Q}}(k)$ and $\hat{\boldsymbol{E}}(k)$, by Theorem 26, we always have $\hat{E}_n(k) \geq \sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}(k)$, thus:

$$\hat{E}_n(k+1) = \hat{E}_n(k) - \sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}(k) + e_n(k). \tag{7.40}$$

We consider the following three cases:

(I) $\hat{E}_n(k) < \mathcal{E}_n$. Since $\hat{E}_n(k+1) > \mathcal{E}_n$, we must have $\mathcal{E}_n - \hat{E}_n(k) \leq e_n(k)$. Then according to the harvesting rule,

$$\begin{aligned}
E_n(k+1) &= \min\big[[E_n(k) - \sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}(k)]^+ + e_n(k) - \mathcal{E}_n + \hat{E}_n(k), M\big] \\
&\geq \min\big[[\hat{E}_n(k) + E_n(k) - \sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}(k)]^+ + e_n(k) - \mathcal{E}_n, M\big] \\
&\geq \min\big[\hat{E}_n(k) - \sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}(k) + e_n(k) - \mathcal{E}_n, M\big] \\
&= \min\big[\hat{E}_n(k+1) - \mathcal{E}_n, M\big] \\
&= \min\big[[\hat{E}_n(k+1) - \mathcal{E}_n]^+, M\big].
\end{aligned}$$

[11] Note that Lemma 4 in Chapter 4 concerns only about data queues; whereas here we also have the energy queues. However, by neglecting the effect of them, the same lemma applies.

Here the first inequality uses the property of $[\cdot]^+$, and the second inequality uses $E_n(k) \geq 0$ and $\hat{E}_n(k) \geq \sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}(k)$.

(II) $\hat{E}_n(k) > \mathcal{E}_n + M$. In this case, we see by the induction assumption that $E_n(k) \geq \min\left[[\hat{E}_n(k) - \mathcal{E}_n]^+, M\right] = M$, which implies that $E_n(k) = M$. Then, by the update rule, we see that:

$$E_n(k+1) = \min\left[E_n(k) + e_n(k), M\right] = M. \tag{7.41}$$

Thus (7.26) still holds.

(III) $\mathcal{E}_n \leq \hat{E}_n(k) \leq \mathcal{E}_n + M$. In this case, we have by induction that:

$$E_n(k) \geq \min\left[[\hat{E}_n(k) - \mathcal{E}_n]^+, M\right] = \hat{E}_n(k) - \mathcal{E}_n. \tag{7.42}$$

We have two sub-cases:

(III-A) If $\hat{E}_n(k+1) - \mathcal{E}_n \leq M$, then using (7.39) and (7.40), we have:

$$\begin{aligned}
\min\left[[\hat{E}_n(k+1) - \mathcal{E}_n]^+, M\right] &= \hat{E}_n(k) - \sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}(k) + e_n(k) - \mathcal{E}_n \\
&\leq \min\left[\left[[\hat{E}_n(k) - \mathcal{E}_n]^+ - \sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}(k)\right]^+ + e_n(k), M\right] \\
&\leq \min\left[[E_n(k) - \sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}(k)]^+ + e_n(k), M\right] \\
&= E_n(k+1).
\end{aligned}$$

Here the first inequality uses the property of the operator $[\cdot]^+$, and the second inequality uses the induction that $E_n(k) \geq \min\left[[\hat{E}_n(k) - \mathcal{E}_n]^+, M\right] = [\hat{E}_n(k) - \mathcal{E}_n]^+$.

(III-B) If $\hat{E}_n(k+1) - \mathcal{E}_n > M$, then we must have $\hat{E}_n(k) \geq \mathcal{E}_n + M - \alpha_{max}$, and that $E_n(k) \geq \hat{E}_n(k) - \mathcal{E}_n \geq M - \alpha_{max}$. Using the fact that $\frac{M}{2} \geq \alpha_{max} \geq P_{max}$, we see that $E_n(k) - \sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}(k) \geq 0$. Thus:

$$E_n(k+1) = \min\left[E_n(k) - \sum_{b \in \mathcal{N}_n^{(o)}} P_{[n,b]}(k) + e_n(k), M\right]$$

$$\begin{aligned} &\geq \min\big[\hat{E}_n(k) - \sum_{b\in\mathcal{N}_n^{(o)}} P_{[n,b]}(k) + e_n(k) - \mathcal{E}_n, M\big] \\ &= \min[\hat{E}_n(k+1) - \mathcal{E}_n, M], \end{aligned}$$

which implies $E_n(k+1) = M$. Thus (7.26) holds. This completes the proof of (7.26) and proves the lemma. □

7.9.4 Proof of Theorem 28

Here we prove Theorem 28. We use the following "exponential attraction" theorem, which is a modified version of Theorem 6 in Chapter 4. In the theorem, we write $g(\boldsymbol{\upsilon}, \boldsymbol{\nu})$ as a function of $\boldsymbol{y} = (\boldsymbol{\upsilon}, \boldsymbol{\nu})$, and use $\boldsymbol{y}^*$ to denote an optimal solution of $g(\boldsymbol{y})$.

Theorem 29. *Suppose that $[\boldsymbol{h}(t), S(t)]$ evolves according some finite state irreducible and aperiodic Markov chain, that $\boldsymbol{y}^* = (\boldsymbol{\upsilon}^*, \boldsymbol{\nu}^*)$ is finite and unique, that $\boldsymbol{\theta}$ is chosen such that $\theta_n + \nu_n^* > 0$, $\forall\ n$, and that for all $\boldsymbol{y} = (\boldsymbol{\upsilon}, \boldsymbol{\nu})$ with $\boldsymbol{\upsilon} \succeq \mathbf{0}, \boldsymbol{\nu} \in \mathbb{R}^N$, the dual function $g(\boldsymbol{y})$ satisfies:*

$$g(\boldsymbol{y}^*) \geq g(\boldsymbol{y}) + L||\boldsymbol{y}^* - \boldsymbol{y}||, \tag{7.43}$$

for some constant $L > 0$ independent of V. Then $\boldsymbol{y}^* = \Theta(V)$, and that under ESA, there exists constants $D, K, c^* = \Theta(1)$, i.e., all independent of V, such that for any $m \in \mathbb{R}_+$,

$$\mathcal{P}^{(r)}(D, Km) \quad \leq \quad c^* e^{-m}, \tag{7.44}$$

where $\mathcal{P}^{(r)}(D, Km)$ is defined:

$$\mathcal{P}^{(r)}(D, Km) \triangleq \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} Pr\{\mathcal{E}(\tau, m)\}, \tag{7.45}$$

with $\mathcal{E}(t, m)$ being the following deviation event*:*

$$\begin{aligned} \mathcal{E}(t, m) = \{\exists\,(n, c), |Q_n^{(c)}(t) - \upsilon_n^{(c)*}| > D + Km\} \\ \cup\{\exists\ n,\ |(E_n(t) - \theta_n) - \nu_n^*| > D + Km\}. \end{aligned} \tag{7.46}$$

Proof. The proof is similar to the proof of Theorem 5 in Chapter 4 and is thus omitted here. □

Now we present the proof of Theorem 28.

Proof. (Theorem 28) Since a steady state distribution for the queues exists under the ESA algorithm, we see that $\mathcal{P}^{(r)}(D, Km)$ is the steady state probability that event $\mathcal{E}(t,m)$ happens. Now consider a large V value that satisfies $\frac{M}{8} = \frac{1}{2}[\log(V)]^2 \geq 2D$ and $\log(V) \geq 16K$, and define:

$$m^* \triangleq \frac{\frac{1}{2}[\log(V)]^2 - D}{K} \geq \frac{\frac{1}{4}[\log(V)]^2}{K} \geq 4\log(V). \tag{7.47}$$

Since at time T, the system is in its steady state, by using (7.44) and (7.47), we see that

$$\begin{aligned} \Pr(\mathcal{E}(T, m^*)) &= \Pr(\mathcal{E}(T, \frac{\frac{1}{2}[\log(V)]^2 - D}{K})) \\ &\leq c^* e^{-m^*} \\ &\leq c^* e^{-4\log(V)} = O(1/V^4). \end{aligned}$$

Using the definition of $\mathcal{E}(t,m)$, we see that when V is large enough, i.e., when (7.47) holds, with probability $1 - O(1/V^4)$, the vectors $\hat{\boldsymbol{E}}(T)$ and $\hat{\boldsymbol{Q}}(T)$ satisfy the following for all n, c:

$$|\hat{Q}_n^{(c)}(T) - \upsilon_n^{(c)*}| \leq \frac{M}{8}, |\hat{E}_n(T) - (\theta_n + \nu_n^*)| \leq \frac{M}{8}. \tag{7.48}$$

Using the fact that $\mathcal{Q}_n^{(c)} = [\hat{Q}_n^{(c)}(T) - \frac{M}{2}]^+$ and $\mathcal{E}_n = [\hat{E}_n(T) - \frac{M}{2}]^+$, (7.48) and the facts that $M = 4[\log(V)]^2$ and $\boldsymbol{y}^* = (\boldsymbol{\upsilon}^*, \boldsymbol{\nu}^*) = \Theta(V)$, we see that when V is large enough, with probability $1 - O(1/V^4)$, we have:

$$-\frac{3M}{8} \geq \mathcal{Q}_n^{(c)} - \upsilon_n^{(c)*} \geq -\frac{5M}{8}, \forall\, (n,c) \text{ s.t. } \upsilon_n^{(c)*} \neq 0, \tag{7.49}$$

$$\mathcal{Q}_n^{(c)} = \upsilon_n^{(c)*}, \forall\, (n,c) \text{ s.t. } \upsilon_n^{(c)*} = 0, \tag{7.50}$$

$$-\frac{3M}{8} \geq \mathcal{E}_n - (\theta_n + \nu_n^*) \geq -\frac{5M}{8}, \forall\, n. \tag{7.51}$$

Having established (7.49)-(7.51), (7.30) can now be proven using (7.25) in Lemma 14 and a same argument as in the proof of Theorem 9 in Chapter 4.

Now we consider (7.31). Since at every time t, MESA performs ESA's data admission, and routing and scheduling actions, if there was no packet dropping, then MESA achieves the same utility performance as ESA. However, since all the utility functions have bounded derivatives, to prove the utility performance of MESA, it suffices to show that the average rate of the packets dropped is $O(\epsilon) = O(1/V)$.

To prove this, we first see that packet dropping happens at time t only when the following event $\hat{\mathcal{E}}(t)$ happens, i.e.,

$$\begin{aligned}\hat{\mathcal{E}}(t) = \{\exists\ n, \hat{E}_n(t) < \mathcal{E}_n + P_{max}\} \\ \cup\{\exists\ n, \hat{E}_n(t) > \mathcal{E}_n + M\} \\ \cup\{\exists\ (n,c), \hat{Q}_n^{(c)}(t) < \mathcal{Q}_n^{(c)}\}.\end{aligned} \tag{7.52}$$

However, assuming (7.49)-(7.51) hold, we have: $\mathcal{E}_n + P_{max} \leq (\theta_n + \nu_n^*) - \frac{3M}{8} + P_{max}$, $\mathcal{E}_n + M \geq (\theta_n + \nu_n^*) + \frac{3M}{8}$ and $\mathcal{Q}_n^{(c)} \leq \upsilon_n^{(c)*} - \frac{3M}{8}$ for all $\upsilon_n^{(c)*} \neq 0$. Therefore the following event must happen for $\hat{\mathcal{E}}(t)$ to happen:

$$\begin{aligned}\tilde{\mathcal{E}}(t) \quad = \quad & \{\exists\, n,\ \hat{E}_n(t) < (\theta_n + \nu_n^*) - \frac{3M}{8} + P_{max}\} \\ & \cup\{\exists\, n,\ \hat{E}_n(t) > (\theta_n + \nu_n^*) + \frac{3M}{8}\} \\ & \cup\{\exists\, (n,c),\ \hat{Q}_n^{(c)}(t) < \upsilon_n^{(c)*} - \frac{3M}{8},\ \forall\, \upsilon_n^{(c)*} \neq 0\}.\end{aligned}$$

Therefore $\hat{\mathcal{E}}(t) \subset \tilde{\mathcal{E}}(t)$. However, it is easy to see from (7.46) that $\tilde{\mathcal{E}}(t) \subset \mathcal{E}(t, \tilde{m})$ with $\tilde{m} = (\frac{3M}{8} - P_{max} - D)/K = (\frac{3}{2}[\log(V)]^2 - P_{max} - D)/K$. Therefore $\hat{\mathcal{E}}(t) \subset \mathcal{E}(t, \tilde{m})$. Using (7.44) again, we see that:

$$\begin{aligned} \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \Pr(\hat{\mathcal{E}}(\tau)) &\leq \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \Pr(\mathcal{E}(\tau, \tilde{m})) \\ &\leq c^* e^{-(\frac{3[\log(V)]^2}{2} - P_{max} - D)/K}. \end{aligned} \tag{7.53}$$

Using the facts that $\frac{1}{2}[\log(V)]^2 \geq 2D$ and $\log(V) \geq 16K$, we see that:

$$\frac{\frac{3[\log(V)]^2}{2} - D}{K} \geq \frac{\frac{5[\log(V)]^2}{4}}{K} \geq 20\log(V). \tag{7.54}$$

Thus we conclude that: [12]

$$\limsup_{t\to\infty} \sum_{\tau=0}^{t-1} \Pr(\hat{\mathcal{E}}(\tau)) \leq \frac{c^* e^{P_{max}/K}}{V^{20}} = O(1/V).$$

Since at every time slot, the total amount of packets dropped is no more than $2N^2\mu_{max} + NR_{max}$, we see that the average rate of packets dropped is $O(1/V)$.

Finally, by (7.53) and (7.54), we see that the packet drop rate is $O(1/V^{\frac{3\log(V)}{2}})$. This completes the proof of Theorem 28. □

[12] Note here that the term $\frac{1}{V^{20}}$ is due to the M value we choose. We can choose a different M value to get a different exponent.

Part IV

Conclusion and Future Work

Chapter 8

Conclusion and future work

In this thesis, we extend the Lyapunov network optimization technique to resolve two important and challenging problems in the queueing network area, i.e., delay and underflow. By establishing a novel connection between the Lyapunov technique and a deterministic optimization program, we develop two extensions of the Lyapunov technique. Specifically, we present two systematic ways for constructing algorithms to achieve good delay performance for optimization problems in communication networks, and to perform optimal scheduling in complex networks in the presence of no-underflow constraints. These two extensions greatly enlarge the range of problems that can be solved by the Lyapunov optimization technique and contribute to developing an optimal control theory for stochastic networks. The development of the results provides us with the following insights into algorithm design for queueing network problems.

• Queue as Lagrange multiplier: To achieve optimal network utility, Max-Weight algorithms use the queue vector to represent the Lagrange multiplier. Thus, the implementation of Max-Weight can be viewed as applying a dual subgradient method to an

underlying optimization problem. This important feature ensures the performance of the Max-Weight algorithms.

• Delay reduction as Lagrange multiplier engineering: The problem of reducing network queueing delay is equivalent to engineering the Lagrange multiplier, so that it goes through a trajectory that has a small sum value. In this case, to develop delay-efficient algorithms, it is very important to take the structure of the underlying dual function into account.

• Underflow prevention as lifting the Lagrange multiplier: The no-underflow constraint requires that the output rates of some queues in the network must be exactly equal to their input rates. This requirement makes some entries of the Lagrange multiplier negative. Hence, Max-Weight can no longer be applied, because it only uses the nonnegative queue vector to track the Lagrange multiplier. In this case, perturbation resolves this problem by lifting the Lagrange multiplier values to make it entry-wise positive. This again highlights the importance of Lagrange multiplier engineering for queueing network algorithm design.

There are many interesting directions in which the results in this thesis can be further extended.

• Better buffer provision with respect to network size: The results developed in Chapters 4, 6 and 7, though being effective, all treat the network sizes as given parameters, and only explore buffer reduction with respect to the proximity to the optimal utility. Specifically, the results show that the required buffer sizes are $O(1/\epsilon)$ for an $O(\epsilon)$ close-to-optimal utility. However, the constants multiplying these order terms are $\Theta(N^2)$, where N is the network size. Such an scaling law can be very inefficient, and incur large network

congestion when the network size is large. Thus, it is important to construction algorithms that have smaller multiplicative constants, e.g., $O(N)$. The redundant constraint approach developed in Chapter 5 can likely be useful in this case.

• Multi-time-scale network control: The models in this thesis all assume time-slotted systems, where the network dynamics and actions change every time slot. However, in practice, different components of the network may evolve according to different time scales. For instance, in a sensor network, the nodes may go to sleep and wake up every tens of seconds, whereas they may schedule packet transmissions and receptions every hundred milliseconds. As another example, in a processing network, assembling two different parts may take a longer time than painting each one of them. Thus, one important extension of the current work is to develop multi-time-scale stochastic network control techniques. Such an extension is very challenging, and will find applications in many important networking problems as well as in the operations research area.

• Game theoretic optimal network control: The results in this work can be viewed as being derived in the optimal network control regime, where all the network components are assumed to comply with the rules and behave accordingly. However, in practice, due to the abundant information available to the network components, as well as the self-interests of them, the network nodes may not always behave in this "normal" manner. For instance, in a transportation network, the nodes may want to reduce its own delay and choose to traverse a path that is less congested, rather than going to a suggested route by the controller, or customers using a network access point may want to anticipate the prices and transmit packets strategically, rather than being price-takers. In this case, it is important to develop network control algorithms that also take into account such

selfish behavior of the network components. Such algorithms will be very useful in areas such as transportation networks and the Smart Grid.

- Delay-efficient low-complexity algorithm design: Another interesting problem is to see how the techniques developed here can be combined with the low-complexity CSMA algorithm design technique developed recently in [JW10]. This CSMA-type technique is very suitable for distributed algorithm design in, e.g., peer-to-peer systems. However, these CSMA-type algorithms tend to have unsatisfying delay performance. Also, it is not clear whether it can easily be applied to problems involving the no-underflow constraints. It will therefore be interesting to see how the results here can be used together with this technique to develop low-complexity algorithms for queueing networks.

Bibliography

[Alt99] E. Altman. *Constrained Markov Decision Processes.* Chapman and Hall/CRC, 1999.

[AOS04] D. Acemoglu, A. Ozdaglar, and R. Srikant. The marginal user principle for resource allocation in wireless networks. *Proceedings of IEEE Conf. on Decision and Control*, Dec. 2004.

[BC03] G. R. Bitran and R. Caldentey. An overview of pricing models for revenue management. *Manufacturing and Service Operations Management*, Vol. 5, No. 3, pp. 203-209, Summer 2003.

[Ber03] D. P. Bertsekas. *Nonlinear Programming, 2nd Edition.* Athena Scientific, 2003.

[Ber07] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Vols. I and II.* Boston: Athena Scientific, 2005 and 2007.

[BN07] N. Buchbinder and J. Naor. *The Design of Competitive Online Algorithms via a PrimalDual Approach.* Foundations and Trends in Theoretical Computer Science Vol. 3, Nos. 2-3, pp. 93-263, 2007.

[BNO03] D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar. *Convex Analysis and Optimization.* Boston: Athena Scientific, 2003.

[BS02] T. Basar and R. Srikant. Revenue-maximizing pricing and capacity expansion in a many-users regime. *Proceedings of IEEE INFOCOM*, 2002.

[BSS09] L. Bui, R. Srikant, and A. Stolyar. Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing. *Proceedings of IEEE INFOCOM 2009 Mini-Conference*, April 2009.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[BZ99] E. K. Browning and M. A. Zupan. *Microeconomic theory and applications.* Addison-Wesley Educational Publishers, Inc., 1999.

[CC08] S. Chalasani and J. M. Conrad. A survey of energy harvesting sources for embedded systems. *IEEE Southeastcon*, 2008.

[CFK95] M. A. Crew, C. S. Fernando, and P. R. Kleindorfer. The theory of peak-load pricing: A survey. *Journal of Regulatory Economics*, 8(3):215–48, November 1995.

[CL] F. Chung and L. Lu. Concentration inequalities and martingale inequalities: a survey. *Internet Math.*, 3 (2006-2007), 79–127.

[CNT05] C. Curescu and S. Nadjm-Tehrani. Price/utility-based optimized resource allocation in wireless ad hoc networks. *IEEE SECON*, 85- 95, 2005.

[DL05] J. G. Dai and W. Lin. Maximum pressure policies in stochastic processing networks. *Operations Research, Vol 53, 197-218*, 2005.

[Dur05] R. Durret. *Probability: Theory and Examples.* Cambridge University Press, 3rd Edition, 2005.

[ene10] Power from thin air. *Economist*, June 10, 2010.

[ES07] A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. *IEEE/ACM Trans. Netw.*, 15(6):1333–1344, 2007.

[Fol99] G. B. Folland. *Real Analysis: Modern Techniques and their Applications.* Wiley, 2nd edition, 1999.

[FP03] E. J. Friedman and D. C. Parkes. Pricing wifi at starbucks– issues in online mechanism design. *Proceedings of Fourth ACM Conf. on Elec. Commerce (EC'03), pp.240-241*, 2003.

[Gal96] R. G. Gallager. *Discrete Stochastic Processes.* Kluwer Academic Publishers, 1996.

[GGT10] M. Gatzianas, L. Georgiadis, and L. Tassiulas. Control of wireless networks with rechargeable batteries. *IEEE Trans. on Wireless Communications, Vol. 9, No. 2*, Feb. 2010.

[GJ07] P. Gupta and T. Javidi. Towards throughput and delay-optimal routing for wireless ad-hoc networks. *Asilomar Conference on Signals, Systems and Computers*, Nov 2007.

[GKK+09] M. Gorlatova, P. Kinget, I. Kymissis, D. Rubenstein, X. Wang, and G. Zussman. Challenge: Ultra-low-power energy-harvesting active networked tags (EnHANTs). *Proceedings of MobiCom*, Sept. 2009.

[GNT06] L. Georgiadis, M. J. Neely, and L. Tassiulas. *Resource Allocation and Cross-Layer Control in Wireless Networks.* Foundations and Trends in Networking Vol. 1, no. 1, pp. 1-144, 2006.

[GR09] D. Graham-Rowe. Wireless power harvesting for cell phones. *MIT Technology Review*, June, 2009.

[HC10] T. R. Halford and K. M. Chugg. Barrage relay networks. *Information Theory and Applications Workshop (ITA)*, 2010.

[HMNK11] L. Huang, S. Moeller, M. J. Neely, and B. Krishnamachari. Lifo-backpressure achieves near optimal utility-delay tradeoff. *Proceedings of WiOpt*, May 2011.

[HN07] L. Huang and M. J. Neely. The optimality of two prices: Maximizing revenue in a stochastic network. *Proceedings of 45th Annual Allerton Conference on Communication, Control, and Computing (invited paper)*, Sept. 2007.

[HN09] L. Huang and M. J. Neely. Delay reduction via Lagrange multipliers in stochastic network optimization. *Proceedings of WiOpt*, June 2009.

[HN10a] L. Huang and M. J. Neely. Max-weight achieves the exact $[O(1/V), O(V)]$ utility-delay tradeoff under Markov dynamics. *arXiv:1008.0200v1*, 2010.

[HN10b] L. Huang and M. J. Neely. Utility optimal scheduling in processing networks. *ArXiv Technical Report, avXiv:1008.1862v1*, 2010.

[HN10c] L. Huang and M. J. Neely. The optimality of two prices: Maximizing revenue in a stochastic network. *IEEE/ACM Transactions on Networking, Vol. 18, No.2*, April 2010.

[HN10d] L. Huang and M. J. Neely. Delay efficient scheduling via redundant constraints in multihop networks. *Proceedings of WiOpt 2011*, May, 2010.

[HN11a] L. Huang and M. J. Neely. Delay reduction via Lagrange multipliers in stochastic network optimization. *IEEE Transactions on Automatic Control, Volume 56, Issue 4, pp. 842-857*, April 2011.

[HN11b] L. Huang and M. J. Neely. Utility optimal scheduling in energy harvesting networks. *Proceedings of MobiHoc*, May 2011.

[HNar] L. Huang and M. J. Neely. Delay efficient scheduling via redundant constraints in multihop networks. *Elsevier's Performance Evaluation*, To appear.

[HR51] O. Hanner and H. Rådström. A generalization of a theorem of Ȩenchel. *Proceedings of American Mathematical Society*, vol. 2, no. 4, pp. 589-593, Aug. 1951.

[JW09] L. Jiang and J. Walrand. Stable and utility-maximizing scheduling for stochastic processing networks. *Allerton Conference on Communication, Control, and Computing*, 2009.

[JW10] Libin Jiang and Jean 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*, Jun. 2010.

[KA03] N. J. Keon and G. Anandalingam. Optimal pricing for multiple services in telecommunications networks offering quality-of-service guarantees. *IEEE/ACM Trans. Netw.*, 11(1):66–80, 2003.

[Kel97a] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, vol. 8, pp. 33-37, 1997.

[Kel97b] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, vol. 8, pp. 33-37, 1997.

[KHZS07] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. Power management in energy harvesting sensor networks. *ACM Trans. on Embedded Computing Systems, Vol.6, Issue 4*, Sept. 2007.

[LCL07] R. K. Lam, D. Chiu, and J. C. S. Lui. On the access pricing and network scaling issues of wireless mesh networks. *IEEE Transactions on Computers, vol. 56, No. 11*, Nov. 2007.

[LL99] S. H. Low and D. E. Lapsley. Optimization flow control, i: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, vol. 7(6): 861-75, Dec. 1999.

[LPC08] Y. Li, A. Papachristodoulou, and M. Chiang. Stability of congestion control schemes with delay sensitive traffic. *Proceedings of IEEE ACC, Seattle, WA*, June 2008.

[LPW08] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 1st Edition, Dec 2008.

[LS05] X. Lin and N. B. Shroff. Simplification of network dynamics in large systems. *IEEE/ACM Transactions on Networking*, vol. 13, no. 4, pp. 813-826, August 2005.

[LS06] X. Lin and N. B. Shroff. The impact of imperfect scheduling on cross-layer congestion control in wireless networks. *IEEE/ACM Transactions on Networking, vol. 14, no. 2, pp302-315*, April 2006.

[LSS05] L. Lin, N. B. Shroff, and R. Srikant. Asymptotically optimal power-aware routing for multihop wireless networks with renewable energy sources. *Proceedings of INFOCOM*, 2005.

[LSS07] L. Lin, N. B. Shroff, and R. Srikant. Energy-aware routing in sensor networks: A large system appraoch. *Ad Hoc Networks, Vol. 5, Issue 6, 818-831*, 2007.

[MB02] P. Marbach and R. Berry. Downlink resource allocation and pricing for wireless networks. *Proceedings of IEEE INFOCOM*, 2002.

[McA02] R. Preston McAfee. Coarse matching. *Econometrica, Vol. 70, Number 5, 2025-2034*, September 2002.

[Mit04] M. Mitzenmacher. Digital fountains: A survey and look forward. *Information Theory Workshop*, 2004.

[MMMA+01] S. Meninger, J. O. Mur-Miranda, R. Amirtharajah, A. Chandrakasan, and J. H. Lang. Vibration-to-eletric energy conversion. *IEEE Trans. on VLSI, Vol. 9, No.1*, Feb. 2001.

[MMV95] J. K. MacKie-Mason and H. R. Varian. Pricing congestible network resources. *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1141–1149, 1995.

[MSKG10] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali. Routing without routes: The backpressure collection protocol. *9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), pp. 279-290*, 2010.

[Mun00] J. R. Munkres. *Topology.* NJ: Prentice Hall, Inc., 2000.

[MW06] J. Musacchio and J. Walrand. Wifi access point pricing as a dynamic game. *IEEE/ACM Trans. Netw.*, 14(2):289–301, 2006.

[Nee03] M. J. Neely. *Dynamic Power Allocation and Routing for Satellite and Wireless Networks with Time Varying Channels.* PhD thesis, Massachusetts Institute of Technology, Laboratory for Information and Decision Systems (LIDS), 2003.

[Nee06a] M. J. Neely. Intelligent packet dropping for optimal energy-delay tradeoffs in wireless downlinks. *Proceedings of the 4th Int. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, April 2006.

[Nee06b] M. J. Neely. Super-fast delay tradeoffs for utility optimal fair scheduling in wireless networks. *IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Nonlinear Optimization of Communication Systems, vol. 24, no. 8, pp. 1489-1501*, Aug. 2006.

[Nee06c] M. J. Neely. Energy optimal control for time-varying wireless networks. *IEEE Transactions on Information Theory 52(7): 2915-2934*, July 2006.

[Nee07] M. J. Neely. Optimal energy and delay tradeoffs for multi-user wireless downlinks. *IEEE Transactions on Information Theory vol. 53, no. 9, pp. 3095-3113*, Sept. 2007.

[Nee09a] M. J. Neely. Delay analysis for maximal scheduling with flow control in wireless networks with bursty traffic. *IEEE Transactions on Networking, vol. 17, no. 4, pp. 1146-1159*, August 2009.

[Nee09b] M. J. Neely. Delay analysis for max weight opportunistic scheduling in wireless systems. *IEEE Transactions on Automatic Control, Vol. 54, No. 9, pp. 2137-2150*, Sept. 2009.

[Nee10a] M. J. Neely. Universal scheduling for networks with arbitrary traffic, channels, and mobility. *Proceedings of IEEE Conf. on Decision and Control (CDC)*, Atlanta, GA, Dec 2010.

[Nee10b] M. J. Neely. Stability and capacity regions or discrete time queueing networks. *arXiv:1003.3396v1*, March 2010.

[NH10] M. J. Neely and L. Huang. Dynamic product assembly and inventory control for maximum profit. *IEEE Conference on Decision and Control (CDC), Atlanta, Georgia*, Dec. 2010.

[NML08] M. J. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. *IEEE/ACM Trans. on Networking, vol. 16, no. 2, pp. 396-409*, April 2008.

[NMR05] M. J. Neely, E. Modiano, and C. E. Rohrs. Dynamic power allocation and routing for time-varying wireless networks. *IEEE Journal on Selected Areas in Communications, Vol 23, NO.1, pp. 89-103*, January 2005.

[NU08] M. J. Neely and R. Urgaonkar. Opportunism, backpressure, and stochastic optimization with the wireless broadcast advantage. *Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA (Invited paper)*, Oct. 2008.

[NZJ09] M. Naghshvar, H. Zhuang, and T. Javidi. A general class of throughput optimal routing policies in multi-hop wireless networks. *arXiv:0908.1273v1*, Aug 2009.

[PPC06] L. Pelusi, A. Passarella, and M. Conti. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *IEEE Communications Magazine, Vol. 44, Issue 11*, Nov. 2006.

[PT00] I. Ch. Paschalidis and J. N. Tsitsiklis. Congestion-dependent pricing of network services. *IEEE/ACM Trans. Netw.*, 8(2):171–184, 2000.

[RKH+05] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. B. Srivastava. Design considerations for solar energy harvesting wireless embedded systems. *Proceedings of IEEE IPSN*, April 2005.

[Ros96] S. Ross. *Stochastic Processes*. John Wiley and Sons Inc., New York, 1996.

[SdV09] B. Sadiq and S. Baekand G. de Veciana. Delay-optimal opportunistic scheduling and approximations: The log rule. *Proceedings of IEEE INFOCOM*, April 2009.

[SK10] R. Srivastava and C. E. Koksal. Basic tradeoffs for energy management in rechargeable sensor networks. *ArXiv Techreport arXiv: 1009.0569v1*, Sept. 2010.

[SMJG10] V. Sharma, U. Mukherji, V. Joseph, and S. Gupta. Optimal energy management policieis for energy harvesting sensor nodes. *IEEE Trans. on Wireless Communication, Vol.9, Issue 4.*, April 2010.

[SS06] S. Shakkottai and R. Srikant. Economics of network pricing with multiple isps. *IEEE/ACM Trans. Netw.*, 14(6):1233–1245, 2006.

[SSS04] S. Shakkottai, R. Srikant, and A. Stolyar. Pathwise optimality of the exponential scheduling rule for wireless channels. *Advances in Applied Probability,*, December 2004.

[TE92] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Trans. on Automatic Control, vol. 37, no. 12, pp. 1936-1949*, Dec. 1992.

[TE93] L. Tassiulas and A. Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Transactions on Information Theory, Vol. 39, No. 2, pp. 466-478*, March1993.

[UN08] R. Urgaonkar and M. J. Neely. Opportunistic scheduling with reliability guarantees in cognitive radio networks. *Proceedings of INFOCOM*, April 2008.

[Vaz03] V. V. Vazirani. *Approximation Algorithms.* Springer, 2003.

[VL07] V. J. Venkataramanan and X. Lin. Structural properties of ldp for queue-length based wireless scheduling algorithms. *45th Annual Allerton Conference on Communication, Control, and Computing, Monticello, Illinois,*, September 2007.

[WALJ+06] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. *7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2006.

[YC08] Y. Yi and M. Chiang. Stochastic network utility maximization: A tribute to Kelly's paper published in this journal a decade ago. *European Transactions on Telecommunications*, vol. 19, no. 4, pp. 421-442, June 2008.

[YSR09] L. Ying, S. Shakkottai, and A. Reddy. On combining shortest-path and back-pressure routing over multihop wireless networks. *Proceedings of IEEE INFOCOM*, April 2009.

[YST08] L. Ying, R. Srikant, and D. Towsley. Cluster-based back-pressure routing algorithm. *Proceedings of IEEE INFOCOM*, April 2008.

[YZC09] Y. Yi, J. Zhang, and M. Chiang. Delay and effective throughput of wireless scheduling in heavy trafc regimes: Vacation model for complexity. *Proceedings of MobiHoc*, May 2009.

[ZDT07] Z. Zhang, D. Dey, and Y. Tan. Pricing communication services with delay guarantee. *INFORMS Journal on Computing, Vol.19, No.2, pp248-260*, 2007.

Appendix

Duality

In this appendix, we provide a very brief review of the duality theory. The materials here are based on books [BNO03] and [BV04]. The notation is a little bit different from those in [BNO03] and [BV04], and is chosen to be consistent with the notation used in this thesis.

We consider the following problem:

$$\min : \quad f(\boldsymbol{x}) \tag{.1}$$

$$\text{s.t.} \quad h_j(\boldsymbol{x}) \leq 0, \, \forall \, j, \tag{.2}$$

$$\boldsymbol{x} \in \mathcal{X}. \tag{.3}$$

Here $\mathcal{X}$ is a subset of $\mathbb{R}^n$, $f(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$, $g_j(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$, $\forall \, j$, are given functions. This problem is referred to as the *primal problem*. Note that equality constraints, i.e., $h_j(\boldsymbol{x}) = 0$ can also be included in the problem (.1) - (.3), by writing every such constraint as two inequality constraints, i.e., $h_j(\boldsymbol{x}) \leq 0$ and $-h_j(\boldsymbol{x}) \leq 0$.

To obtain the dual problem of (.1), we associate with each constraint in (.2) a multiplier γ_j, and form the following Lagrangian

$$L(\boldsymbol{x}, \boldsymbol{\gamma}) \triangleq f(\boldsymbol{x}) + \sum_j \gamma_j h_j(\boldsymbol{x}). \tag{.4}$$

The Lagrangian is a function of the primal variable $\boldsymbol{x}$ and the dual variable $\boldsymbol{\gamma} = (\gamma_1, ..., \gamma_r)^T$, which is also called the Lagrange multiplier. The dual function $g(\boldsymbol{\gamma})$ is given by

$$\begin{aligned} g(\boldsymbol{\gamma}) &\triangleq \inf_{\boldsymbol{x} \in \mathcal{X}} L(\boldsymbol{x}, \boldsymbol{\gamma}) \qquad (.5) \\ &= \inf_{\boldsymbol{x} \in \mathcal{X}} \left\{ f(\boldsymbol{x}) + \sum_j \gamma_j h_j(\boldsymbol{x}) \right\}. \end{aligned}$$

The *dual problem* is then given by:

$$\max : \quad g(\boldsymbol{\gamma}) \tag{.6}$$

$$\text{s.t.} \quad \boldsymbol{\gamma} \succeq \mathbf{0}. \tag{.7}$$

The constraint (.7) is due to the fact that the primal constraints are *inequality* constraints. In this appendix, we assume that both the primal problem and the dual problem have finite optimal values, and denote them by f^* and g^*, i.e.,

$$f^* \triangleq \inf_{\boldsymbol{x} \in \mathcal{X}, g_j(\boldsymbol{x}) \leq 0} f(\boldsymbol{x}), \quad g^* \triangleq \sup_{\boldsymbol{\gamma} \succeq 0} g(\boldsymbol{\gamma}). \tag{.8}$$

In many cases, for example, when the functions $f(\cdot)$, $g_j(\cdot)$ are convex, the feasible set $\mathcal{X}$ is convex and certain qualifying conditions such as slackness, are met, we have $g^* = f^*$, in which cases we say that there is no duality gap. However, for any general $f(\cdot)$ and $g_j(\cdot)$ functions and feasible set $\mathcal{X}$, we always have the following theorem, which is more important for the results in this thesis.

Theorem 30. *(Weak Duality) We have $g^* \leq f^*$.*

Proof. Consider any $\boldsymbol{\gamma} \succeq \mathbf{0}$. For any $\boldsymbol{x} \in \mathcal{X}$ with $g_j(\boldsymbol{x}) \leq 0$ for all j, we have

$$g(\boldsymbol{\gamma}) = \inf_{\boldsymbol{x} \in \mathcal{X}} \left\{ f(\boldsymbol{x}) + \sum_j \gamma_j h_j(\boldsymbol{x}) \right\} \leq f(\boldsymbol{x}) + \sum_j \gamma_j h_j(\boldsymbol{x}) \leq f(\boldsymbol{x}).$$

Thus,

$$g^* = \sup_{\boldsymbol{\gamma} \succeq 0} g(\boldsymbol{\gamma}) \leq \inf_{\boldsymbol{x} \in \mathcal{X}, g_j(\boldsymbol{x}) \leq 0} f(\boldsymbol{x}) = f^*,$$

completing the proof. □