2012

Resource allocation in wireless networks with flow level dynamics

Shihuan Liu
Iowa State University

Resource allocation in wireless networks with flow level dynamics

by

Shihuan Liu

A dissertation submitted to the Graduate Faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Iowa State University

Ames, Iowa

2012

DEDICATION

I first would like to dedicate this thesis to my parents. Without their support I would not be able to come to U.S. to pursue my Ph.D. degree and complete it successfully. I also would like to thank my advisor, Prof. Lei Ying, without his outstanding guidance I would not know what is research and how to do research. I finally would like to thank all my friends for their warm friendship and unreserved help in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. Introduction

With the ever increasing popularity of wireless networks, wireless devices are prevalent in our daily life. With cell phones and cellular networks, we can chat with others almost everywhere and anytime; with laptops and Wi-Fi access points, we can easily access a large variety of information online; and with GPS on our vehicles, driving has become much easier than before. Wireless communications have become such an important technology, and is penetrating every aspect of our daily life. Nowadays, wireless networks not only need to support traditional voice communications, but also are carrying more and more Internet traffics such as e-mails, instant messages, and even video streaming. All these applications are demanding wireless networks to support fast and reliable communications.

To build a high-performance wireless network, a key challenge is to allocate the wireless resources dynamically and efficiently among users. This resource allocation problem or scheduling problem is at the core of wireless network design. Compared to traditional wired networks, this scheduling in wireless networks is much harder because of the reasons listed below. First of all, the bandwidth of wireless networks is limited. A typical transmission rate in a wireless network is at the order of tens of megabits per second, while gigabits per second transmission rate can be easily achieved in wired networks. Second, in wireless communications, channel quality may vary significantly across users, locations and time. We therefore need to learn channel conditions and schedule links intelligently based on channel qualities. Finally, nearby transmissions can significantly interfere with each other, so we cannot activate all link at the same time and need to schedule them properly. In a summary, limited bandwidth, rapidly varying channel qualities and wireless interference make it difficult to design high performance scheduling algorithms in wireless networks, which is the key to the success of next-generation wireless networks.

In this thesis, I first looked at the scheduling problem in the downlink of wireless cellular networks. Designing multiuser scheduling algorithms in cellular networks is a very challenging problem because of multi-scale dynamics: channel-level dynamics (channel fading), packet-level dynamics (stochastic packet arrivals) and flow-level dynamics (dynamic flow arrivals/departures). A seminal result in this area is the celebrated MaxWeight scheduling [14], which deals with both channel-level and packet-level dynamics by selecting users based on the product of channel state and queue length. Under the assumption that the set of users/nodes is fixed and all traffic flows are persistent, the MaxWeight scheduling is throughput optimal for general channel and traffic models [4–6]. In other words, it can stabilize any traffic that is stabilizable by any other schemes.

While the results in [3–6] demonstrate the power of MaxWeight-based algorithms, they were obtained under the assumptions that *the number of users in the network is fixed and the traffic flow generated by each user is long-lived, i.e., each user continually injects new bits into the network.* In other words, flow-level dynamics is not considered in these results. However, in practical systems, users dynamically arrive to transmit data and leave the network after the data are fully transmitted. In a recent paper [1], the authors showed that the MaxWeight algorithm is in fact *not throughput optimal* in networks with flow-level dynamics by providing a clever example showing the instability of the MaxWeight scheduling. The intuition is as follows: if a long-lived flow does not receive enough service, its backlog builds up, which forces the MaxWeight scheduler to allocate more service to the flow. This interaction between user backlogs and scheduling guarantees the correctness of the resource allocation. However, if a flow has only a finite number of bits, its backlog does not build up over time and it is possible for the MaxWeight to stop serving such a flow and thus, the flow may stay in the network forever. Thus, in a network where finite-sized flows continue to arrive, the number of flows in the network could increase to infinity. One may wonder why flow-level instability is important since, in real networks, base stations limit the number of simultaneously active flows in the network by rejecting new flows when the number of existing flows reaches a threshold. The reason is that, if a network model without such upper limits is unstable in the sense that the number of flows grows unbounded, then the corresponding real network with an upper limit on the number of flows will experience high flow blocking rates. This fact is demonstrated in the simulation parts of Chapter 2 and Chapter 3 later.

In [1], the authors address this instability issue of MaxWeight-based algorithms, and establish necessary and sufficient conditions for the stability of networks with flow-level dynamics. The authors also propose throughput-optimal scheduling algorithms. However, as the authors mention in [1], the proposed algorithms require prior knowledge of channel distribution and traffic distribution, which is difficult and sometimes impossible to obtain in practical systems, and further, the performance of the proposed algorithms is also not ideal. A delay-driven MaxWeight scheduler has also been proposed to stabilize the network under flow-level dynamics [2]. The algorithm however works only when the maximum achievable rates of the flows are identical.

Since flow arrivals and departures are common in reality, the focus of my Ph.D. study is to develop *practical* scheduling algorithms that are throughput-optimal *under flow-level dynamics*. I first studied the *single-channel* case, where we consider a wireless downlink system with a single base station and multiple users (flows), and the base station uses a single frequency band (channel) for transmission. Based on this model, first, the necessary conditions for flow-level stability of networks were obtain. I have also developed algorithms that are based on the estimated *workload*, the number of time slots required to transmit the remainder of a flow based on the best channel condition seen by the flow so far for scheduling and proved that the algorithm is throughput-optimal under flow-level dynamics.

After that, motivated by current and next generation cellular systems (e.g., WiMax and LTE) implementing the Orthogonal Frequency Division Multiplexing (OFDM), I have investigated *multichannel* wireless cellular networks. These systems have hundreds of sub-carriers, and are grouped into tens of channels for scheduling purposes. In a multichannel network, the base station can transmit to multiple mobile users simultaneously over different channels. Specifically, similar to the single-channel case, we consider a downlink wireless network where a base station is responsible for scheduling downlink transmissions. We assume mobile users dynamically join the network for receiving finite-sized files and leave the network after downloading the files. For such multichannel wireless networks, an important question that should be answered is the following: *Are the algorithms designed for single-channel networks [1, 2, 16] still throughput optimal for multichannel networks in the presence of flow-level dynamics?* The answer to this question is *no.* A counter example will be presented in Chapter 3. In fact in multichannel wireless networks, the base station not only needs to decide which flow to serve on each channel, but also how to split a flow across multiple channels. We call the second problem the channel-assignment problem. Designing the channel-assignment algorithm is a key contribution of this chapter, which makes both the intuition and the analysis fundamentally different from those for single-channel networks.

Based on the ideas stated above, we first derive the necessary conditions for the stability of multi-channel downlink networks in the presence of flow-level dynamics. Then, we develop a throughput optimal algorithm, which we call *joint channel-assignment and workload-based scheduling (CA-WS),* in which the channel assignment algorithm is derived based on an optimization formulation and its Lagrangian dual. We prove that the CA-WS algorithm is throughput optimal in the presence of flow-level dynamics. Although it seems that by now we have solved the problem, a problem comes up, which is that the CA-WS algorithm starts to serve a flow only after the complete file is received at the base-station, so the performance of the CA-WS is worse than the MaxWeight in light or medium traffic regimes. We then propose a hybrid CA-WS algorithm, which schedules those flows who are still injecting packets to the base station using the MaxWeight scheduling; and schedules fully arrived flows (those flows who have been completely transferred to the base-station) using an algorithm called workload-based scheduling. The hybrid CA-WS algorithm seamlessly combines the MaxWeight scheduling and the workload-based scheduling, and we prove that it is also throughput optimal in multichannel downlink networks with flow-level dynamics. Simulation results will be presented in Chapter 3 to validate the performances of the proposed algorithms.

During my Ph.D. study, I also investigated the resource allocation problem in wireless peer-to-peer networks in the presence of flow-level dynamics for real-time traffic, i.e., each packet is associated with a hard deadline. In wireless peer-to-peer networks, each pair of users is allowed to communicate directly with each other. All transmissions are single-hop. This communication pattern is more efficient compared to the transmission pattern in cellular networks in the sense that the delivery of each packet no longer needs to go through two hops - uplink and downlink. However, in wireless peer-to-peer networks, the admission control and medium access control are very important because concurrent transmissions may cause severe interference if they are not arranged wisely. We assume there is a limited-functional central controller in the wireless peer-to-peer network, which schedules transmissions in each time slot, and our objective is designing a joint congestion control and scheduling algorithm which maximizes the network welfare while satisfying the delay constraints of the users.

The well-known MaxWeight scheduler [3, 14] is still throughput optimal here when the user population is fixed, but no longer provides maximum throughput in the presence of flow-level dynamics. We have developed novel solutions to handle this problem in wireless cellular networks. However, none of them are meant to support the traffic with strict delay constraints.

In Chapter 4, we propose an optimization formulation for the problem of service allocation and scheduling of real-time messages under strict per-message deadline constraints in wireless peer-to-peer networks. We show that using the fact that the network is aware of the location of the devices allows us to deal with the difficulty of scheduling small-sized messages, translating the problem of serving message requests into a long-term formulation where messages are grouped by regions where channel and interference conditions are similar. The formulation allows for very general interference constraints and arrival models. Based on this modeling, we design an optimal service controller and scheduler that allocates service such that it maximizes the total network utility in a stochastic sense, while meeting deadline constraints. Through simulations we compare our algorithm against the MaxWeight scheduling algorithm which is an optimal solution for scheduling persistent real-time traffic and show the limitations of the MaxWeight approach to handle real-time messages for providing fairness, and the need to develop a new approach.

After addressing the resource allocation problem in wireless networks with single-hop traffic flows, I started to look at the resource allocation in wireless networks with multihop flows. Although most of existing practical wireless networks include only single-hop data transmissions, multi-hop wireless networks have many important applications and are expected to be deployed in future.

A widely-used algorithm to stabilize multi-hop flows in wireless networks is the back-pressure algorithm proposed in [3], which can stabilize any traffic flows that can be supported by any other routing/scheduling algorithm. We refer to [9, 10] for a comprehensive survey on the back-pressure algorithm and its variations. A key idea of the back-pressure algorithm is to use largest queue difference as link weight, and schedule the links with largest aggregated weights. Therefore, the back-pressure algorithm requires constant exchange of queue-length information among neighboring nodes. Furthermore, under the back pressure algorithm, the sum of the queue lengths along a route increases quadratically as the route length [18], which leads to poor delay performance. The most important thing is, the back-pressure algorithm is optimal only for the network without flow-level dynamics. When the system has flow-level dynamics, it is no longer throughput optimal as pointed out in [1].

To address the scheduling problem in the presence of flow-level dynamics, we can think of the nodes in wireless ad-hoc networks as "access points" (AP). Each user who wants to transmit data should first associate with a particular AP and transmit data to it, and then the associated AP will forward the data to the destination AP through wireless links, who will finally dump the data to the destination user. By doing this we "translate" the flow-level dynamics to packet-level dynamics. Now the question is, considering the drawbacks of the back-pressure algorithm, *can the network be stabilized without using back pressure?* We address this question in a multi-hop wireless network with fixed routing. We note that a multi-hop flow with a fixed route can be broken into multiple single-hop flows, one for each link on the route. A scheduling policy that stabilizes the collection of single-hop flows also provides sufficient service for supporting the set of multi-hop flows. Therefore, assuming each link knows the aggregated rate it needs to carry, an alternative scheduling approach is to let each link generate virtual packets according to the aggregated rate and then let the network schedule the links according to the virtual queues. When a virtual queue is scheduled, real packets are served according to the allocated link rate. This approach is throughput optimal under the fixed routing assumption, but again requires information exchange in network. A source needs to estimate the arrival rate of the associated flow and communicate the rate to all nodes along the route of the flow. A directly following question is whether and under what conditions, it is possible to stabilize a network without explicitly exchanging any information among nodes in the network.

In Chapter 5, we present an algorithm that can achieve this goal for networks where *(i)* the arrival rates of flows follow some statistical property, and *(ii)* routes of flows are fixed. We propose a self-regulated MaxWeight scheduling algorithm where each node estimates the aggregated link rate locally, i.e., by taking average over the past arrivals on that link. We would like to emphasize that the accuracy of link-rate estimates relies on the stability of the network because if one queue builds up, it blocks packets to down-stream nodes so that those nodes cannot accurately estimate the link rates. On the other hand, the stability of the network relies on the accuracy of the link-rate estimates. This is an interesting paradox which makes the stability of the self-regulated MaxWeight scheduling a non-trivial problem. In Chapter 5, we prove that the self-regulated MaxWeight scheduling is throughput optimal when both (i) and (ii) are satisfied. The self-regulated MaxWeight scheduling combined with distributed scheduling algorithms such as the CSMA-based scheduling [19–21] provides a scheduling algorithm for multi-hop wireless networks, which does not require any information exchange in the network. Finally, we would like to comment that the proposed algorithm is motivated by the idea of regulators, which was first proposed for re-entrant lines in [22] and later used for scheduling in wireless networks [23]. Our algorithm however does not require any information exchange in network, while the algorithm in [23] requires the mean arrival rates to be communicated to regulators in the network.

The rest of the report is organized as follows. In Chapter 2 and 3, I present our works on wireless cellular networks, for the single-channel case and the multi-channel case respectively.

In Chapter 4, our work on the resource allocation problem in wireless peer-to-peer networks is presented. The solution to the scheduling problem in wireless multi-hop networks is introduced in Chapter 5. In Chapter 6, I conclude my Ph.D. research.

CHAPTER 2. Scheduling in Single-channel Wireless Cellular Networks

In this chapter, I present the scheduling algorithm for wireless single-channel system with flow-level dynamics, which will be extended to the multi-channel case in Chapter 3. We assume the network contains both long-lived flows, which keep injecting bits into the network, and short-lived flows, which have a finite number of bits to transmit. A short-lived flow joins the network to download some packets, and leaves the system when all its packets are transmitted.

The terminology of long-lived and short-lived flows above has to be interpreted carefully in practical situations. In practice, each flow has a finite size and thus, all flows eventually will leave the system if they receive sufficient service. Thus, all flows are short-lived flows in reality. Our results suggest that transmitting to users who are individually in their best estimated channel state so far is thus, throughput optimal. On the other hand, it is also well known that real network traffic consists of many flows with only a few packets and a few flows with a huge number of packets. If one considers the time scales required to serve the small-sized flows, the large-sized flows will appear to be long-lived (i.e., persistent forever) in the terminology above. Thus, if one is interested in performance over short time-scales, an algorithm which considers flows with a very large number of packets as being long-lived may lead to better performance and hence, we consider the more general model which consists of both short-lived flows and long-lived flows. Our simulations in Section 2.6 confirm the fact that the algorithm which treats some flows are being long-lived leads to better performance although through-optimality does not require such a model. In addition, long-lived flows partially capture the scenario where all bits from a flow do not arrive at the base station all at once.

2.1 Basic Model

Network Model: We consider a discrete-time wireless downlink network with a single base station and many flows, each flow associates with a distinct mobile user. The base station can serve only one flow at a time.

Traffic Model: The network consists of the following two types of flows:

- **Long-lived flows:** Long-lived flows are traffic streams that are always in the network and continually generate bits to be transmitted.

- **Short-lived flows:** Short-lived flows are flows that have a finite number of bits to transmit. A short-lived flow enters the network at a certain time, and leaves the system after all bits are transmitted.

We assume that the set of long-lived flows is fixed, and short-lived flows arrive and depart. We let l be the index for long-lived flows, $\mathcal{L}$ be the set of long-lived flows, and L be the number of long-lived flows, i.e., $L = |\mathcal{L}|$. Furthermore, we let $X_l(t)$ be the number of new bits injected by long-lived flow l in time slot t, where $X_l(t)$ is a discrete random variable with finite support, and independently and identically distributed (i.i.d.) across time slots. We also assume $\mathbf{E}[X_l(t)] = x_l$ and $X_l(t) \leq X^{\max}$ for all l and t.

Similarly, we let i be the index for short-lived flows, $\mathcal{I}(t)$ be the set of short-lived flows in the network at time t, and $I(t)$ be the number of short-lived flows at time t, i.e., $I(t) = |\mathcal{I}(t)|$. We denote by f_i the size (total number of bits) of short-lived flow i, and assume $f_i \leq F^{\max}$ for all i.

It is important to note that we allow different short-lived flows to have different maximum link rates. A careful consideration of our proofs will show the reader that the learning algorithm is not necessary if all users have the same maximum rate and that one can simply transmit to the user with the best channel state if it is assumed that all users have the same maximum rate. However, we do not believe that this is a very realistic scenario since SNR variations will dictate different maximum rates for different users.

Residual Size and Queue Length: For a short-lived flow i, let $Q_i(t)$, which we call the residual size, denote the number of bits still remaining in the system at time t. For a long-lived flow l, let $Q_l(t)$ denote the number of bits stored at the queue at the base station.

Channel Model: There is a wireless link between each user and the base station. Denote by $R_i(t)$ the state of the link between short-lived flow i and the base station at time t (i.e., the maximum rate at which the base station can transmit to short-lived flow i at time t), and $R_l(t)$ the state of the link between long-lived flow l and the base station at time t. We assume that $R_i(t)$ and $R_l(t)$ are discrete random variables with finite support. Define $R_i^{\max}$ and $R_l^{\max}$ to be the largest values that these random variables can take, i.e., $\Pr(R_j(t) > R_j^{\max}) = 0$ for each $j \in \mathcal{L} \bigcup \left(\bigcup_t \mathcal{I}(t)\right)$. Choose $p_s^{\max} > 0$ and $R^{\max} > 0$ such that

$$\Pr(R_i(t) = R_i^{\max}) \geq p_s^{\max} \qquad \forall i, t$$

$$\max\left\{\max_i R_i^{\max}, \max_l R_l^{\max}\right\} \leq R^{\max}.$$

The states of wireless links are assumed to be independent across flows and time slots (but not necessarily identically distributed across flows). The independence assumption across time slots can be relaxed easily but at the cost of more complicated proofs.

2.2 Workload-based Scheduling with Learning

In this section, we introduce a new scheduling algorithm called Workload-based Scheduling with Learning (WSL).

Workload-based Scheduling with Learning: For a short-lived flow i, we define

$$\tilde{R}_i^{\max}(t) = \max_{\max\{t-D, b_i\} \leq s \leq t} R_i(s),$$

where b_i is the time short-lived flow i joins the network and $D > 0$ is called the learning period. A key component of this algorithm is to use $R_i^{\max}$ to evaluate the workload of short-lived flows (the reason will be explained in a detail in Section 2.4). However, $R_i^{\max}$ is in general unknown, so the scheduling algorithm uses $\tilde{R}_i^{\max}(t)$ as an estimate of $R_i^{\max}$.

During each time slot, the base station first checks the following inequality:

$$\alpha \sum_{i \in \mathcal{I}(t)} \left\lceil \frac{Q_i(t)}{\tilde{R}_i^{\max}(t)} \right\rceil > \max_{l \in \mathcal{L}} Q_l(t) R_l(t), \tag{2.1}$$

where $\alpha > 0$.

- If inequality (2.1) holds, then the base station serves a short-lived flow as follows: if at least one short-lived flow (say flow i) satisfies $R_i(t) \geq Q_i(t)$ or $R_i(t) = \tilde{R}_i^{\max}(t)$, then the base station selects such a flow for transmission (ties are broken according to a *good* tie-breaking rule, which is defined at the end of this algorithm); otherwise, the base station picks an arbitrary short-lived flow to serve.

- If inequality (2.1) does not hold, then the base station serves a long-lived flow l^* such that

$$l^* \in \arg\max_{l \in \mathcal{L}} Q_l(t) R_l(t)$$

(ties are broken arbitrarily).

"Good" tie-breaking rule: Assume that the tie-breaking rule is applied to pick a short-lived flow every time slot (but the flow is served only if $\alpha \sum_{i \in \mathcal{I}(t)} \left\lceil \frac{Q_i(t)}{\tilde{R}_i^{\max}(t)} \right\rceil > \max_{l \in \mathcal{L}} Q_l(t) R_l(t)$). We define $\mathcal{E}_{miss}(t)$ to be the event that the tie-breaking rule selects a short-lived flow with $\tilde{R}_i^{\max}(t) \neq R_i^{\max}$. Define

$$W_s(t) = \sum_{i \in \mathcal{I}(t)} \left\lceil \frac{Q_i(t)}{R_i^{\max}} \right\rceil,$$

which is the total workload of the system at time t. A tie-breaking rule is said to be *good* if the following condition holds: Consider the WSL with the given tie-breaking rule and learning period D. Given any $\epsilon_{miss} > 0$, there exist $N_{\epsilon_{miss}}$ and $D_{\epsilon_{miss}}$ such that

$$\Pr\left(\mathcal{E}_{miss}(t)\right) \leq \epsilon_{miss}$$

if $D \geq D_{\epsilon_{miss}}$ and $W_s(t-D) \geq N_{\epsilon_{miss}}$.

Remark 1: While all WSL scheduling algorithms with good tie-breaking rules are throughput optimal, their performances in terms of other metrics could be different depending upon the tie-breaking rules. We consider two tie-breaking rules in this chapter:

- **Uniform Tie-breaking:** Among all short-lived flows satisfying $R_i(t) = \tilde{R}_i^{\max}(t)$ or $R_i(t) \geq Q_i(t)$, the base-station uniformly and randomly selects one to serve.

- **Oldest-first Tie-breaking:** Let β_i denote the number of time slots a short-lived flow has been in the network. The base station keeps track $\tau_i = \min\{\bar{\tau}, \beta_i\}$ for every short-lived flow, where $\bar{\tau}$ is some fixed positive integer. Among all short-lived flows satisfying $R_i(t) = \tilde{R}_i^{\max}(t)$ or $R_i(t) \geq Q_i(t)$, the tie-breaking rule selects the one with the largest τ_i, and the ties are broken uniformly and randomly.[1]

The "goodness" of these two tie-breaking rules are proved in Appendix C and D of our technical report [16], and the impact of the tie-breaking rules on performance is studied in Section 2.6 using simulations.

Remark 2: The α in inequality (2.1) is a parameter balancing the performance of long-lived flows and short-lived flows. A large α leads to a small number of short-lived flows but large queue-lengths of long-lived flows, and vice versa.

Remark 3: In Theorem 3, we will prove that WSL is throughput optimal when D is sufficiently large. From purely throughput-optimality considerations, it is then natural to choose $D = \infty$. However, in practical systems, if we choose D too large, such as ∞, then it is possible that a flow may stay in the system for a very long time if its best channel condition occurs extremely rarely. Thus, it is perhaps best to choose a finite D to tradeoff between performance and throughput.

Remark 4: If all flows are short-lived, then the algorithm simplifies as follows: If at least one short-lived flow (say flow i) satisfies $R_i(t) \geq Q_i(t)$ or $R_i(t) = \tilde{R}_i^{\max}(t)$, then the base station selects such a flow for transmission according to a "good" tie-breaking rule; otherwise, the base station picks an arbitrary short-lived flow to serve. Simply stated, the algorithm serves one of the flows which can be completely transmitted or sees its best channel state, where the best channel state is an estimate based on past observations. If no such flow exists, any flow can be served. We do not separately prove the throughput optimality of this scenario since it is

[1]We set a upper bound $\bar{\tau}$ on β for technical reasons that facilitate the throughput-optimality proof. Since $\bar{\tau}$ can be arbitrarily large, we conjecture that this upper bound is only for analysis purpose, and not required in practical systems.

a special case of the scenario considered here. But it is useful to note that, in the case of short-lived flows only, the algorithm does not consider backlogs at all in making scheduling decisions.

We will prove that WSL (with any $\alpha > 0$) is throughput-optimal in the following sections, i.e., the scheduling policy can support any set of traffic flows that are supportable by any other algorithm. In the next section, we first present the necessary conditions for the stability, which also define the network throughput region.

2.3 Necessary Conditions for Stability

In this section, we establish the necessary conditions for the stability of networks with flow-level dynamics. To get the necessary condition, we need to classify the short-lived flows into different classes.

- A short-lived flow class is defined by a pair of random variables $(\hat{R}, \hat{F})$. Class-k is associated with random variables $\hat{R}_k$ and $\hat{F}_k$.[2] A short-lived flow i belongs to class k if $R_i(t)$ has the same distribution as $\hat{R}_k$ and the size of flow i (f_i) is a realization of $\hat{F}_k$. We let $\Lambda_k(t)$ denote the number of class-k flows joining the network at time t, where $\Lambda_k(t)$ are i.i.d. across time slots and independent but not necessarily identical across classes, and $\mathbf{E}[\Lambda_k(t)] = \lambda_k$. Denote by $\mathcal{K}$ the set of distinct classes. We assume that $\mathcal{K}$ is finite, $|\mathcal{K}| = K$, and $\Lambda_k[t] \leq \lambda^{\max}$ for all t and $k \in \mathcal{K}$.

- Let $\mathbf{c}$ denote an L-dimensional vector describing the state of the channels of the long-lived flows. In state $\mathbf{c}$, $R_{\mathbf{c},l}$ is the service rate that long-lived flow l can receive if it is scheduled. We denote by $\mathcal{C}$ the set of all possible states.

- Let $\mathbf{C}(t)$ denote the state of the long-lived flows at time t, and $\pi_{\mathbf{c}}$ denote the probability that $\mathbf{C}(t)$ is in state $\mathbf{c}$.

[2]We use $\hat{}$ to indicate that the notation is associated with a class of short-lived flows instead of an individual short-lived flow.

- Let $p_{\mathbf{c},l}$ be the probability that the base station serves flow l when the network is in state $\mathbf{c}$. Clearly, for any $\mathbf{c}$, we have
$$\sum_{l\in\mathcal{L}} p_{\mathbf{c},l} \leq 1.$$
Note that the sum could be less than 1 if the base station schedules a short-lived flow in this state.

- Let $\mu_{\mathbf{c},s}$ be the probability that the base station serves a short-lived flow when the network is in state $\mathbf{c}$.

- Let $\Theta_{k,\beta}(t)$ denote the number of short-lived flows that belong to class-k and have residual size $Q(t) = \beta$. Note that β can only take on a finite number of values.

Theorem 1. *Consider traffic parameters $\{x_l\}$ and $\{\lambda_k\}$, and suppose that there exists a scheduling policy guaranteeing*

$$\lim_{t\to\infty} \mathbf{E}\left[\sum_{l\in\mathcal{L}} Q_l(t) + \sum_{k\in\mathcal{K}}\sum_{\beta=1}^{F^{\max}} \Theta_{k,\beta}(t)\right] < \infty.$$

Then there exist $p_{\mathbf{c},l}$ and $\mu_{\mathbf{c},s}$ such that the following inequalities hold:

$$x_l \leq \sum_{\mathbf{c}\in\mathcal{C}} \pi_{\mathbf{c}} p_{\mathbf{c},l} R_{\mathbf{c},l} \quad \forall l \in \mathcal{L}. \tag{2.2}$$

$$\sum_{k\in\mathcal{K}} \lambda_k \mathbf{E}\left[\left\lceil \frac{\hat{F}_k}{\hat{R}_k^{\max}} \right\rceil\right] \leq \sum_{\mathbf{c}\in\mathcal{C}} \mu_{\mathbf{c},s}\pi_{\mathbf{c}}. \tag{2.3}$$

$$\left(\sum_{l\in\mathcal{L}} p_{\mathbf{c},l}\right) + \mu_{\mathbf{c},s} \leq 1 \ \forall c \in \mathcal{C}. \tag{2.4}$$

□

Inequality (2.2) and (2.3) state that the service allocated should be no less than the user requests if the flows are supportable. Inequality (2.4) states that the overall time used to serve long-lived and short-lived flows should be no more than the time available. To prove this theorem, it can be shown that for any traffic for which we cannot find $p_{\mathbf{c},l}$ and $\mu_{\mathbf{c},s}$ satisfying the three inequalities in the theorem, a Lyapunov function can be constructed such that the expected drift of the Lyapunov function is larger than some positive constant under any scheduling algorithm, which implies the instability of the network. The complete proof is based on the Strict Separation Theorem and is along the lines of a similar proof in [5], and is omitted in this chapter.

2.4 Throughput Optimality of WSL

First, we provide some intuition into how one can derive the WSL algorithm from optimization decomposition considerations. Then, we will present our main throughput optimality results. Given traffic parameters $\{x_l\}$ and $\{\lambda_k\}$, the necessary conditions for the supportability of the traffic is equivalent to the feasibility of the following constraints:

$$\begin{aligned} x_l &\le \textstyle\sum_{\mathbf{c}\in\mathcal{C}} \pi_{\mathbf{c}} p_{\mathbf{c},l} R_{\mathbf{c},l} && \forall l \\ \textstyle\sum_{k\in\mathcal{K}} \lambda_k \mathbf{E}\left[\left\lceil \frac{\hat{F}_k}{\hat{R}_k^{\max}} \right\rceil\right] &\le \textstyle\sum_{\mathbf{c}\in\mathcal{C}} \mu_{\mathbf{c},s}\pi_{\mathbf{c}} && \\ \textstyle\sum_{l\in\mathcal{L}} p_{\mathbf{c},l} + \mu_{\mathbf{c},s} &\le 1 && \forall \mathbf{c}. \end{aligned} \tag{2.5}$$

For convenience, we view the feasibility problem as an optimization problem with the objective $\max A$, where A is some constant. While we have not explicitly stated that the x's and μ's are non-negative, this is assumed throughout.

Partially augmenting the objective using Lagrange multipliers, we get

$$\begin{aligned} &\max A - \textstyle\sum_{l\in\mathcal{L}} q_l (x_l - \sum_c \pi_{\mathbf{c}} p_{\mathbf{c},l} R_{\mathbf{c},l}) - \\ &q_s \left(\textstyle\sum_{k\in\mathcal{K}} \lambda_k \mathbf{E}\left[\left\lceil \frac{\hat{F}_k}{\hat{R}_k^{\max}} \right\rceil\right] - \sum_{\mathbf{c}\in\mathcal{C}} \mu_{\mathbf{c},s}\pi_{\mathbf{c}} \right) \\ s.t. \quad &\textstyle\sum_{l\in\mathcal{L}} p_{\mathbf{c},l} + \mu_{\mathbf{c},s} \le 1 \ \forall \mathbf{c}. \end{aligned}$$

For the moment, let us assume Lagrange multipliers q_l and q_s are given. Then the maximization problem above can be decomposed into a collection of optimization problems, one for each $\mathbf{c}$:

$$\begin{aligned} &\max_{p_{\mathbf{c},l},\mu_{\mathbf{c},s}} \sum_{l\in\mathcal{L}} q_l R_{\mathbf{c},l} p_{\mathbf{c},l} + q_s \mu_{\mathbf{c},s} \\ s.t. \quad &\textstyle\sum_{l\in\mathcal{L}} p_{\mathbf{c},l} + \mu_{\mathbf{c},s} \le 1. \end{aligned}$$

It is easy to verify that one optimal solution to the optimization problem above is:

- if $q_s > \max_{l\in\mathcal{L}} q_l R_{\mathbf{c},l}$, then $\mu_{\mathbf{c},s} = 1$ and $p_{\mathbf{c},l} = 0 (\forall l)$;
- otherwise, $\mu_{\mathbf{c},s} = 0$, and $p_{\mathbf{c},l^*} = 1$ for some $l^* \in \arg\max q_l R_{\mathbf{c},l}$ and $p_{\mathbf{c},l} = 0$ for other l.

The complementary slackness conditions give

$$q_l \left(x_l - \sum_{\mathbf{c}\in\mathcal{C}} \pi_{\mathbf{c}} p_{\mathbf{c},l} R_{\mathbf{c},l} \right) = 0.$$

Since x_l is the mean arrival rate of long-lived flow l and $\sum_{\mathbf{c}\in\mathcal{C}} \pi_{\mathbf{c}} p_{\mathbf{c},l} R_{\mathbf{c},l}$ is the mean service rate, the condition on q_l says that if the mean arrival rate is less than the mean service rate, q_l is equal to zero. Along with the non-negativity condition on q_l, this suggests that perhaps q_l behaves likes a queue with these arrival and service rates. Indeed, it turns out that the mean of the queue lengths are proportional to Lagrange multipliers (see the surveys in [9–11]). For long-lived flow l, we can treat the queue-length $Q_l(t)$ as a time-varying estimate of Lagrange multiplier q_l. Similarly q_s can be associated with a queue whose arrival rate is $\sum_{k\in\mathcal{K}} \lambda_k \mathbf{E}\left[\left\lceil \frac{\hat{F}_k}{R_k^{\max}} \right\rceil\right]$, which is the mean rate at which workload arrives where workload is measured by the number of slots needed to serve a short-lived flow if it is served when its channel condition is the best. The service rate is $\sum_{\mathbf{c}\in\mathcal{C}} \mu_{\mathbf{c},s}\pi_{\mathbf{c}}$ which is the rate at which the workload can potentially decrease when a short-lived flow is picked for scheduling by the base station. Thus, the workload in the system can serve as a dynamic estimate of q_s.

Letting $\alpha W_s(t)$ $(\alpha > 0)$ be an estimate of q_s, the observations above suggest the following workload-based scheduling algorithm if $R_i^{\max}$ are known.

Workload-based Scheduling (WS): During each time slot, the base station checks the following inequality:

$$\alpha W_s(t) > \max_{l\in\mathcal{L}} Q_l(t) R_l(t). \tag{2.6}$$

- If inequality (2.6) holds, then the base station serves a short-lived flow as follows: if at least one short-lived flow (say flow i) satisfies $R_i(t) \geq Q_i(t)$ or $R_i(t) = R_i^{\max}$, then such a flow is selected for transmission (ties are broken arbitrarily); otherwise, the base station picks an arbitrary short-lived flow to serve.

- If inequality (2.6) does not hold, then the base station serves a long-lived flow l^* such that $l^* \in \arg\max_{l\in\mathcal{L}} Q_l(t)R_l(t)$ (ties are broken arbitrarily).
- The factor α can be obtained from the optimization formulation by multiplying constraint (2.5) by α on both sides

□

However, this algorithm which was directly derived from dual decomposition considerations is not implementable since $R_i^{\max}$'s are unknown. So WSL uses $\tilde{R}_i^{\max}(t)$ to approximate $R_i^{\max}$. Note that an inaccurate estimate of $R_i^{\max}$ not only affects the base station's decision on whether $R_i(t) = R_i^{\max}$, but also on its computation of $\left\lceil \frac{Q_i(t)}{R_i^{\max}} \right\rceil$. However, it is not difficult to see that the error in the estimate of the total workload is a small fraction of the total workload when the total workload is large: when the workload is very large, the total number of short-lived flows is large since their file sizes are bounded. Since the arrival rate of short-lived flows is also bounded, this further implies that the majority of short-lived flows must have arrived a long time ago which means that with high probability, their estimate of their best channel condition must be correct.

Next we will prove that both WS and WSL can stabilize any traffic x_l and λ_k such that $(1+\epsilon)x_l$ and $(1+\epsilon)\lambda_k$ are *supportable,* i.e., satisfying the conditions presented in Theorem 1. In other words, the number of short-lived flows in the network and the queues for long-lived flows are all bounded. Even though WS is not practical, we study it first since the proof of its throughput optimality is easier and provides insight into the proof of throughput-optimality of WSL.

Let

$$\mathbf{M}(t) = \left(\{Q_l(t)\}_{l\in\mathcal{L}}, \{\Theta_{k,\beta}(t)\}_{k\in\mathcal{K}, 1\leq\beta\leq F^{\max}}\right).$$

Since the base station makes decisions on $\mathbf{M}(t)$ and $\mathbf{R}(t) = \{\{R_i(t)\}_{i\in\mathcal{I}(t)}, \{R_l(t)\}_{l\in\mathcal{L}}\}$ under WS. It is easy to verify that $\mathbf{M}(t)$ is a *finite-dimensional* Markov chain under WS. Assume that $\Lambda_k(t)$, $\hat{F}_k$ and $X_l(t)$ are such that the Markov chain $\mathbf{M}$ is *irreducible* and *aperiodic.*

Theorem 2. *Given any traffic x_l and λ_k such that $(1+\epsilon)x_l$ and $(1+\epsilon)\lambda_k$ are supportable,* *the Markov chain* $\mathbf{M}(t)$ *is* positive-recurrent *under WS, and*

$$\lim_{t\to\infty} \mathbf{E}\left[\sum_{l\in\mathcal{L}} Q_l(t) + \sum_{i\in\mathcal{I}(t)} Q_i(t)\right] < \infty.$$

Proof. We consider the following Lyapunov function:

$$V(t) = \alpha\left(W_s(t)\right)^2 + \sum_{l\in\mathcal{L}}(Q_l(t))^2, \tag{2.7}$$

and prove that

$$\begin{aligned}\mathbf{E}[V(t+1)-V(t)|\mathbf{M}(t)] \leq U_d 1_{\mathbf{M}(t)\in\Upsilon} - \frac{\epsilon}{2}\Bigg[\alpha\bar{\lambda}W_s(t) \\ + \sum_{l\in\mathcal{L}} Q_l(t)x_l\Bigg] 1_{\mathbf{M}(t)\notin\Upsilon}\end{aligned}$$

for some $U_d > 0$, $\epsilon > 0$, $\bar{\lambda} > 0$, and a finite set Υ. Positive recurrence of $\mathbf{M}$ then follows from Foster's Criterion for Markov chains [12], and the boundedness of the first moment follows from [13]. The detailed proof is presented in Section 2.5.

□

We next study WSL, where $R_i^{\max}$ is estimated from the history. We define $\Theta_{k,\beta,r}(t)$ to be the number of short-lived flows that belong to class-k, have a residual size of β, and have $\tilde{R}_i^{\max}(t) = r$. Furthermore, we define

$$\tilde{\mathbf{M}}(n) = \left(\{Q_l(t)\}_{l\in\mathcal{L}}, \{\Theta_{k,\beta,r}(t)\}_{\substack{k\in\mathcal{K} \\ 1\leq\beta\leq F^{\max} \\ 1\leq r\leq \hat{R}_k^{\max}}}\right)_{(n-1)T+1\leq t\leq nT}$$

from some $T \geq D$. It is easy to see that $\tilde{\mathbf{M}}(n)$ is a *finite-dimensional* Markov chain under WSL.[3]

Theorem 3. *Consider traffic* x_l *and* λ_k *such that* $(1+\epsilon)x_l$ *and* $(1+\epsilon)\lambda_k$ *are supportable. Given WSL with a* good *tie-breaking rule, there exists* D_ϵ *such that the Markov chain* $\tilde{\mathbf{M}}(n)$

[3]This Markov chain is well-defined under the uniform tie-breaking rule. For other good tie-breaking rules, we may need to first slightly change the definition of $\tilde{M}(n)$ to include the information required for tie-breaking, and then use the analysis in Section 2.5 to prove the positive recurrence.

is positive-recurrent *under the WSL with learning period $D \geq D_\epsilon$ and the given tie-breaking rule. Further,*

$$\lim_{t\to\infty} \mathbf{E}\left[\sum_{l\in\mathcal{L}} Q_l(t) + \sum_{i\in\mathcal{I}(t)} Q_i(t)\right] < \infty.$$

Proof. The proof of this theorem is built upon the following two facts:

- When the number of short-lived flows is large, the majority of short-lived flows must have been in the network for a long time and have obtained the correct estimate of the best channel condition, which implies that
$$\sum_{i\in\mathcal{I}(t)} \left\lceil \frac{Q_i(t)}{R_i^{\max}} \right\rceil \approx \sum_{i\in\mathcal{I}(t)} \left\lceil \frac{Q_i(t)}{\tilde{R}_i^{\max}(t)} \right\rceil .$$

- When the number of short-lived flows is large, the short-lived flow selected by the base station (say flow i) has a high probability to satisfy $R_i(t) = R_i^{\max}$ or $R_i(t) \geq Q_i(t)$.

From these two facts, we can prove that with a high probability, the scheduling decisions of WSL are the same as those of WS, which leads to the throughput optimality of WSL. The detailed proof is presented in Section 2.5.

□

2.5 Proofs

2.5.1 Proof of Theorem 2

Recall that $W_s(t) = \sum_{i\in\mathcal{I}(t)} \left\lceil \frac{Q_i(t)}{R_i^{\max}} \right\rceil$. We define $\hat{R}_k^{\max}$ to be the largest achievable link rate of class-k short-lived flows, and $A_s(t) = \sum_{k\in\mathcal{K}} \sum_{i=1}^{\Lambda_k(t)} \left\lceil \frac{f_i}{\hat{R}_k^{\max}} \right\rceil$, which is the amount of new workload (from short-lived flows) injected in the network at time t, and $\mu_s(t)$ to be the decrease of the workload at time t, i.e., $\mu_s(t) = 1$ if the workload of short-lived flows is reduced by one and $\mu_s(t) = 0$ otherwise. Based on the notations above, the evolution of short-lived flows can be described as:

$$W_s(t+1) = W_s(t) + A_s(t) - \mu_s(t).$$

Further, the evolution of $Q_l(t)$ can be described as

$$Q_l(t+1) = Q_l(t) + X_l(t) - \mu_l(t) + u_l(t),$$

where $\mu_l(t)$ is the decrease of $Q_l(t)$ due to the service long-lived flow l receives at time t, and $u_l(t)$ is the unused service due to the lack of data in the queue.

We consider the following Lyapunov function

$$V(t) = \alpha\left(W_s(t)\right)^2 + \sum_{l\in\mathcal{L}} (Q_l(t))^2. \tag{2.8}$$

We will prove that the drift of the Lyapunov function satisfies

$$\begin{aligned}\mathbf{E}[V(t+1) - V(t)|\mathbf{M}(t)] \leq U_d 1_{\mathbf{M}(t)\in\Upsilon} - \frac{\epsilon}{2}\Bigg[\alpha\bar{\lambda}W_s(t)\\ + \sum_{l\in\mathcal{L}} Q_l(t)x_l\Bigg] 1_{\mathbf{M}(t)\notin\Upsilon}\end{aligned}$$

for some $U_d > 0$, $\bar{\lambda} > 0$ and a finite set Υ (the values of these parameters will be defined in the following analysis). Positive recurrence of $\mathbf{M}$ then follows from Foster's Criterion for Markov chains [12].

First, since the number of arrivals, the sizes of short-lived flows and channel rates are all bounded, it can be verified that there exists U, independent of $\mathbf{M}(t)$, such that

$$\begin{aligned}&\mathbf{E}[V(t+1) - V(t)|\mathbf{M}(t)]\\ =&\mathbf{E}\Bigg[\alpha\left(W_s(t+1)\right)^2 - \alpha\left(W_s(t)\right)^2 +\\ &\quad \sum_{l\in\mathcal{L}}(Q_l(t+1))^2 - \sum_{l\in\mathcal{L}}(Q_l(t))^2 \Bigg| \mathbf{M}(t)\Bigg]\\ \leq& U + 2\alpha W_s(t)\mathbf{E}\left[A_s(t) - \mu_s(t)|\,\mathbf{M}(t)\right] +\\ &\quad 2\sum_{l\in\mathcal{L}} Q_l(t)\mathbf{E}\left[X_l(t) - \mu_l(t)|\,\mathbf{M}(t)\right]\\ \leq& U + 2\alpha W_s(t)\left(\left(\sum_{k\in\mathcal{K}}\lambda_k\mathbf{E}\left[\left\lceil\frac{\hat{F}_k}{\hat{R}_k^{\max}}\right\rceil\right]\right)\right.\\ &\quad\left. - \mathbf{E}\left[\mu_s(t)|\,\mathbf{M}(t)\right]\right)\\ &+ 2\sum_{l\in\mathcal{L}} Q_l(t)\left(x_l - \mathbf{E}\left[\mu_l(t)|\,\mathbf{M}(t)\right]\right).\end{aligned}$$

Recall that we assume that $(1+\epsilon)x_l$ and $(1+\epsilon)\lambda_k$ satisfy the supportability conditions of Theorem 1. By adding and subtracting corresponding $p_{\mathbf{c},l}R_{\mathbf{c},l}$ and $\mu_{\mathbf{c},s}$, we obtain that

$$\begin{aligned}
& \mathbf{E}[V(t+1)-V(t)|\mathbf{M}(t)] - U \\
\leq\; & 2\alpha W_s(t)\mathbf{E}\left[\mathbf{E}\left[\mu_{\mathbf{c},s}-\mu_s(t)\right|\mathbf{C}(t)=\mathbf{c}]\right|\mathbf{M}(t)] \\
& +2\sum_{l\in\mathcal{L}} Q_l(t)\mathbf{E}\left[\mathbf{E}\left[p_{\mathbf{c},l}R_{\mathbf{c},l}-\mu_l(t)\right|\mathbf{C}(t)=\mathbf{c}]\right|\mathbf{M}(t)] \\
& -2\epsilon\alpha W_s(t)\bar{\lambda} - 2\epsilon\sum_{l\in\mathcal{L}} Q_l(t)x_l,
\end{aligned}$$

where

$$\bar{\lambda} = \left(\sum_{k\in\mathcal{K}} \lambda_k \mathbf{E}\left[\left\lceil \frac{\hat{F}_k}{\hat{R}_k^{\max}} \right\rceil\right]\right).$$

Next we assume $\mathbf{C}(t)=\mathbf{c}$ and analyze the following quantity

$$\alpha W_s(t)\left(\mu_{\mathbf{c},s}-\mu_s(t)\right) + \sum_{l\in\mathcal{L}} Q_l(t)\left(p_{\mathbf{c},l}R_{\mathbf{c},l}-\mu_l(t)\right). \tag{2.9}$$

We have the following facts:

- **Fact 1:** *Assume that there exists a short-lived flow i such that $R_i(t)=R_i^{\max}$ or $R_i(t)\geq Q_i(t)$.* If a short-lived flow is selected to be served, then the workload of the selected flow is reduced by one and $\mu_s(t)=1$. If long-lived flow l is selected, the rate flow l receives is $R_{\mathbf{c},l}$. Thus, we have that

$$\begin{aligned}
& \alpha W_s(t)\mu_s(t) + \sum_{l\in\mathcal{L}} Q_l(t)\mu_l(t) \\
=\; & \max\left\{\alpha W_s(t), \max_l Q_l(t)R_{\mathbf{c},l}\right\} \\
\geq\; & \alpha W_s(t)\mu_{\mathbf{c},s} + \sum_{l\in\mathcal{L}} Q_l(t)p_{\mathbf{c},l}R_{\mathbf{c},l},
\end{aligned}$$

  where the last inequality holds because $\sum_l p_{\mathbf{c},l} + \mu_{\mathbf{c},s} \leq 1$. Therefore, we have (2.9) ≤ 0 in this case.

- **Fact 2:** *Assume that there does not exist a short-lived flow i such that $R_i(t)=R_i^{\max}$ or $R_i(t)\geq Q_i(t)$.* In this case, we have

$$\begin{aligned}
(2.9) \;\leq\; & \alpha W_s(t) + \max_{l\in\mathcal{L}} Q_l(t)R_{\mathbf{c},l} \\
\leq\; & \alpha W_s(t) + R^{\max}\max_{l\in\mathcal{L}} Q_l(t).
\end{aligned}$$

$\square$

Now we define a set Υ such that

$$\Upsilon = \left\{\mathbf{M} : W_s \leq U_W \text{ and } Q_l \leq U_Q \ \forall l\right\},$$

where U_W is a positive integer satisfying that

$$(1 - p_s^{\max})^{\frac{U_W}{F^{\max}}} \leq \tfrac{\epsilon}{2} \min\left\{\bar{\lambda}, \tfrac{\min_{l \in \mathcal{L}} x_l}{R^{\max}}\right\} \triangleq \epsilon_1 \tag{2.10}$$

$$U_W \geq \tfrac{2U}{\epsilon \alpha \bar{\lambda}}, \tag{2.11}$$

and U_Q is a positive integer satisfying

$$U_Q \geq \frac{4\alpha U_W + U}{\epsilon \min_{l \in \mathcal{L}} x_l}. \tag{2.12}$$

We next compute the drift of the Lyapunov function according to the value of $\mathbf{M}(t)$.

- **Case I:** Assume $\mathbf{M}(t) \in \Upsilon$. According to the definition of Υ, we have

$$\mathbf{E}[V(t+1) - V(t)|\mathbf{M}(t)] \leq U + 2\alpha U_W + 2R^{\max} L U_Q.$$

- **Case II:** Assume $W_s(t) > U_W$. Since the size of a short-lived flow is upper bounded by $F^{\max}$, $W_s(t) > U_W$ implies that at least $\frac{U_W}{F^{\max}}$ short-lived flows are in the network at time t. Define $\mathcal{S}(t)$ to be the following event: no short-lived flow satisfies $R_i(t) = R_i^{\max}$ or $R_i(t) \geq Q_i(t)$.

  Recall that

$$\min_i \Pr(R_i(t) = R_i^{\max}) \geq p_s^{\max}.$$

  Given at least $\frac{U_W}{F^{\max}}$ short-lived flows are in the network, we have that

$$\Pr(1_{\mathcal{S}(t)} = 1) \leq (1 - p_s^{\max})^{\frac{U_W}{F^{\max}}} \leq \epsilon_1.$$

  According to facts 1 and 2, (2.9) is positive only if $\mathcal{S}(t)$ occurs and the value of (2.9) is bounded by $\alpha W_s(t) + R^{\max} \max_{l \in \mathcal{L}} Q_l(t)$. Therefore, we can conclude that in this case

(Case II),

$$\begin{aligned}
&\mathbf{E}[V(t+1)-V(t)|\mathbf{M}(t)] \\
\leq\ & U + 2\epsilon_1 \left(\alpha W_s(t) + R^{\max} \max_{l\in\mathcal{L}} Q_l(t)\right) \\
& -2\epsilon\alpha W_s(t)\bar{\lambda} - 2\epsilon \sum_{l\in\mathcal{L}} Q_l(t)x_l \\
\leq\ & U - \epsilon\alpha W_s(t)\bar{\lambda} - \epsilon \sum_{l\in\mathcal{L}} Q_l(t)x_l \qquad (2.13) \\
\leq\ & -\frac{\epsilon}{2}\left[\alpha\bar{\lambda}W_s(t) + \sum_{l\in\mathcal{L}} Q_l(t)x_l\right] \qquad (2.14)
\end{aligned}$$

where inequality (2.13) holds due to the definition of ϵ_1 (2.10), and inequality (2.14) holds due to inequality (2.11).

- **Case III:** Assume that $W_s(t) \leq U_W$ and $Q_l(t) > U_Q$ for some l. In this case, if a long-lived flow is selected for a given $\mathbf{c}$, we have

$$(2.9) \quad \leq \quad \alpha W_s(t)\mu_{\mathbf{c},s} \leq \alpha W_s(t).$$

Otherwise, if a short-lived flow is selected, it means for the given $\mathbf{c}$, we have $\max_l Q_l(t)R_{\mathbf{c},l} \leq \alpha W_s(t)$, and

$$(2.9) \leq 2\alpha W_s(t).$$

Therefore, we can conclude that in this case,

$$\begin{aligned}
&\mathbf{E}[V(t+1)-V(t)|\mathbf{M}(t)] \\
&\leq U + 4\alpha W_s(t) - 2\epsilon\alpha W_s(t)\bar{\lambda} - 2\epsilon \sum_{l\in\mathcal{L}} Q_l(t)x_l \qquad (2.15) \\
&\leq U + 4\alpha U_W - 2\epsilon\alpha W_s(t)\bar{\lambda} - 2\epsilon \sum_{l\in\mathcal{L}} Q_l(t)x_l \\
&\leq -\frac{\epsilon}{2}\left[\alpha\bar{\lambda}W_s(t) + \sum_{l\in\mathcal{L}} Q_l(t)x_l\right] \qquad (2.16)
\end{aligned}$$

where the last inequality yields from the definition of U_Q (2.12).

From the analysis above, we can conclude that

$$\mathbf{E}[V(t+1) - V(t)|\mathbf{M}(t)] \leq U_d 1_{\mathbf{M}(t)\in\Upsilon} - \frac{\epsilon}{2}\left[\alpha\bar{\lambda}W_s(t) + \sum_{l\in\mathcal{L}} Q_l(t)x_l\right] 1_{\mathbf{M}(t)\notin\Upsilon},$$

where $U_d = U + 2\alpha U_W + 2R^{\max}LU_Q$ and Υ is a set with a finite number of elements. Since $V(t) \geq 0$ for all t, the Lyapunov function is always lower bounded. Further the drift of the Lyapunov is upper bounded when $\mathbf{M}(t)$ belongs to a finite set Υ, and is negative otherwise. So invoking Foster's criterion, the Markov chain $\mathbf{M}(t)$ is positive recurrent and the boundedness of the first moment follows from [13].

2.5.2 Proof of Theorem 3

Consider the network that is operated under WSL, and define $\mathcal{H}(t)$ to be

$$\mathcal{H}(t) \triangleq \left\{Q_l(t), R_l(t), Q_i(t), R_i(t), \tilde{R}_i^{\max}(t)\right\}.$$

Now *given* $\mathcal{H}(t)$, we define the following notations:

- Define $\mu_{2;l}(t) = R_l(t)$ if flow l is selected by WSL, and $\mu_{2;l}(t) = 0$ otherwise.
- Define $\mu_{2;i}(t) = 1$ if flow i is selected by WSL and the workload of flow i can be reduced by one, and $\mu_{2;i}(t) = 0$ otherwise.
- Define $\mu_{1;l}(t) = R_l(t)$ if flow l is selected by WS, and $\mu_{1;l}(t) = 0$ otherwise.
- Define $\mu_{1;i}(t) = 1$ if flow i is selected by WS and the workload of flow i can be reduced by one, and $\mu_{1;i}(t) = 0$ otherwise.

We remark that $\mu_{2;j}(t)$ is the action selected by the base station at time t under WSL and $\mu_{1;j}(t)$ is the action selected by the base station at time t under WS, assuming the same history $\mathcal{H}(t)$.

We define the Lyapunov function to be

$$V(n) = \alpha\left(W_s(nT)\right)^2 + \sum_{l\in\mathcal{L}}(Q_l(nT))^2. \qquad (2.17)$$

This Lyapunov function is similar to the one used in the proof of Theorem 2, and we will show that this is a valid Lyapunov function for workload-based scheduling with learning. Then, it is easy to verify that there exists U_1 independent of $\tilde{\mathbf{M}}(n)$ such that

$$\begin{aligned}
&\mathbf{E}[V(n+1) - V(n)|\tilde{\mathbf{M}}(n)] \\
<&U_1 + 2\alpha \mathbf{E}\left[W_s(nT) \sum_{t=nT}^{(n+1)T-1} (A_s(t) - \mu_{2;s}(t)) \middle| \tilde{\mathbf{M}}(n) \right] \\
&+ \sum_{l\in\mathcal{L}} 2\mathbf{E}\left[Q_l(nT) \sum_{t=nT}^{(n+1)T-1} (X_l(t) - \mu_{2;l}(t)) \middle| \tilde{\mathbf{M}}(n) \right].
\end{aligned}$$

Dividing the time into two segments $[nT, nT+D-1]$ and $[nT+D, (n+1)T-1]$, we obtain

$$\begin{aligned}
&\mathbf{E}[V(n+1) - V(n)|\tilde{\mathbf{M}}(n)] \\
<&U_1 + 2\alpha W_s(nT)\bar{\lambda}D + 2\sum_{l\in\mathcal{L}} Q_l(nT)x_l D \\
&+ 2\alpha \mathbf{E}\left[W_s(nT) \sum_{t=nT+D}^{(n+1)T-1} (A_s(t) - \mu_{2;s}(t)) \middle| \tilde{\mathbf{M}}(n) \right] \\
&+ \sum_{l\in\mathcal{L}} 2\mathbf{E}\left[Q_l(nT) \sum_{t=nT+D}^{(n+1)T-1} (X_l(t) - \mu_{2;l}(t)) \middle| \tilde{\mathbf{M}}(n) \right].
\end{aligned}$$

Note that $|Q_l(t_1) - Q_l(t_2)|$ and $|W_k(t_1) - W_k(t_2)|$ are both bounded by some constants independent of $\tilde{\mathbf{M}}(n)$, so there exists $\tilde{U}$ such that

$$\begin{aligned}
&\mathbf{E}[V(n+1) - V(n)|\tilde{\mathbf{M}}(n)] \\
<&\tilde{U} + 2\alpha W_s(nT)\bar{\lambda}D + 2\sum_{l\in\mathcal{L}} Q_l(nT)x_l D \\
&+ 2\mathbf{E}\left[\alpha \sum_{t=nT+D}^{(n+1)T-1} W_s(t)\left(A_s(t) - \mu_{2;s}(t)\right) \middle| \tilde{\mathbf{M}}(n) \right] \\
&+ \sum_{l\in\mathcal{L}} 2\mathbf{E}\left[\sum_{t=nT+D}^{(n+1)T-1} Q_l(t)\left(X_l(t) - \mu_{2;l}(t)\right) \middle| \tilde{\mathbf{M}}(n) \right].
\end{aligned}$$

Now, by adding and subtracting $\mu_{1;\cdot}(t)$, we obtain

$$\begin{aligned}
&\mathbf{E}[V(n+1) - V(n)|\tilde{\mathbf{M}}(n)] \\
\le&\tilde{U} + 2\alpha W_s(nT)\bar{\lambda}D + 2\sum_{l\in\mathcal{L}} Q_l(nT)x_l D + \sum_{t=nT+D}^{(n+1)T-1} \text{Drift}(t),
\end{aligned}$$

where

$$
\begin{aligned}
&\text{Drift}(t)\\
=&2\mathbf{E}\left[\alpha W_s(t)A_s(t)+\sum_{l\in\mathcal{L}}Q_l(t)X_l(t)\middle|\tilde{\mathbf{M}}(n)\right] && (2.18)\\
&-2\mathbf{E}[\alpha W_s(t)\mu_{1;s}(t)+\sum_{l\in\mathcal{L}}Q_l(t)\mu_{1;l}(t)|\tilde{\mathbf{M}}(n)] && (2.19)\\
&+\sum_{l\in\mathcal{L}}2\mathbf{E}[Q_l(t)\left(\mu_{1;l}(t)-\mu_{2;l}(t)\right)|\tilde{\mathbf{M}}(n)] && (2.20)\\
&+2\mathbf{E}\left[\alpha W_s(t)\left(\mu_{1;s}(t)-\mu_{2;s}(t)\right)|\tilde{\mathbf{M}}(n)\right]. && (2.21)
\end{aligned}
$$

Note that (2.20)+(2.21) is the difference between WS and WSL. In the following analysis, we will prove that this difference is small compared to the absolute value of (2.18)+(2.19).

We define

$$
\begin{aligned}
\text{Diff}(t)=&\alpha W_s(t)\left(\mu_{1;s}(t)-\mu_{2;s}(t)\right)\\
&+\sum_{l\in\mathcal{L}}Q_l(t)\left(\mu_{1;l}(t)-\mu_{2;l}(t)\right),
\end{aligned}
$$

and

$$\tilde{W}_s(t)=\sum_{i\in\mathcal{I}(t)}\left\lceil\frac{Q_i(t)}{\tilde{R}_i^{\max}(t)}\right\rceil.$$

Next, we compute its value in three different situations:

- **Situ-A:** *Consider the situation in which* $\alpha\tilde{W}_s(t)\le\max_{l\in\mathcal{L}}Q_l(t)R_l(t)$. We note that $\tilde{W}_s(t)\ge W_s(t)$ since $\tilde{R}_i^{\max}(t)\le R_i^{\max}$ for all t and i. Therefore, given $\alpha\tilde{W}_s(t)\le\sum_{l\in\mathcal{L}}Q_l(t)$, both WS and WSL will select a long-lived flow. In this case, we can conclude that

  $$\mu_{1;l}(t)=\mu_{2;l}(t)\text{ and }\mu_{1;s}(t)=\mu_{2;s}(t)=0,$$

  and $\text{Diff}(t)=0$.

- **Situ-B:** *Consider the situation in which* $\alpha W_s(t)>\max_{l\in\mathcal{L}}Q_l(t)R_l(t)$. In this case, both WS and WSL will select a short-lived flow, which implies that

  $$\mu_{1;l}(t)=\mu_{2;l}(t)=0,$$

and

$$\begin{aligned}\text{Diff}(t) =&\alpha W_s(t)\left(\mu_{1;s}(t) - \mu_{2;s}(t)\right)\\ &\le \alpha W_s(t)\left(1-\mu_{2;s}(t)\right).\end{aligned}$$

- **Situ-C:** *Consider the situation in which* $\alpha\tilde{W}_s(t) > \max_{l\in\mathcal{L}} Q_l(t)R_l(t) \ge \alpha W_s(t)$. In this case, WS will select a long-lived flow and WSL will select a short-lived flow. We hence have

  $$\mu_{1;l}(t) > 0 \text{ and } \mu_{1;s}(t) = \mu_{2;l}(t) = 0,$$

  and

  $$\begin{aligned}\text{Diff}(t) &= \max_{l\in\mathcal{L}} Q_l(t)R_l(t) - \alpha W_s(t)\mu_{2;s}(t)\\ &\le \alpha\tilde{W}_s(t) - \alpha W_s(t)\mu_{2;s}(t)\end{aligned}$$

□

According to the analysis above, we have that

$$\begin{aligned}&\mathbf{E}[\text{Diff}(t)|\tilde{\mathbf{M}}(n)]\\ \le&\mathbf{E}\left[\alpha W_s(t)|\text{Situ-B}, \mu_{2;s}=0, \tilde{\mathbf{M}}(n)\right]\times\\ &\Pr\left(\text{Situ-B}, \mu_{2;s}=0|\tilde{\mathbf{M}}(n)\right)\\ +&\mathbf{E}\left[\alpha\tilde{W}_s(t)|\text{Situ-C}, \mu_{2;s}=0, \tilde{\mathbf{M}}(n)\right]\times\\ &\Pr\left(\text{Situ-C}, \mu_{2;s}=0|\tilde{\mathbf{M}}(n)\right)\\ +&\mathbf{E}\left[\alpha\tilde{W}_s(t)-\alpha W_s(t)|\text{Situ-C}, \mu_{2;s}=1, \tilde{\mathbf{M}}(n)\right]\times\\ &\Pr\left(\text{Situ-C}, \mu_{2;s}=1|\tilde{\mathbf{M}}(n)\right).\end{aligned}$$

Next we define a finite set $\tilde{\Upsilon}$. We first introduce some constants:

- $\epsilon_1 = \min\left\{\frac{\bar{\lambda}\epsilon}{32}, \frac{\epsilon\min_l x_l}{8R^{\max}}\right\}$.
- $\epsilon_2 = \frac{\bar{\lambda}\epsilon}{32R^{\max}}$, and D_{ϵ_2} and N_{ϵ_2} are the numbers that guarantee $\Pr\left(\mathcal{E}_{miss}(t)\right) \le \epsilon_2$, which are defined by the goodness of the tie-breaking rule.

- $\lambda_W^{\max} = K\lambda^{\max}F^{\max}$, which is the maximum number of bits of short-lived flows injected in one time slot, and also the upper bound on the new workload injected in the network in one time slot.

We define a set $\tilde{\Upsilon}$ such that

$$\tilde{\Upsilon} = \left\{ \tilde{\mathbf{M}}(n) : \begin{array}{c} W_s(nT) \leq \tilde{U}_W + 2T + \frac{2\sum_l x_l R^{\max} T}{\alpha\bar{\lambda}} \\ Q_l(nT) \leq \tilde{U}_Q + \frac{2\alpha\bar{\lambda}T}{\min_l x_l} + \frac{2TR^{\max}\sum_l x_l}{\min_l x_l} \quad \forall l \end{array} \right\}.$$

In this definition, $\tilde{U}_W$ is a positive integer satisfying that

$$(1 - p_s^{\max})^{\frac{\tilde{U}_W}{F^{\max}}} \leq \epsilon_1, \tag{2.22}$$

$$\tilde{U}_W \geq \frac{\frac{8\tilde{U}}{T-D} + 16\epsilon_2\alpha\lambda_W^{\max}T + 8\alpha DR^{\max} + 16\epsilon_2\alpha R^{\max}T + 8\lambda_W^{\max}D}{\epsilon\alpha\bar{\lambda}} \tag{2.23}$$

$$\frac{\tilde{U}_W}{F^{\max}} \geq N_{\epsilon_2}, \tag{2.24}$$

and $\tilde{U}_Q$ is a positive integer satisfying

$$\tilde{U}_Q \geq \frac{8\tilde{U} + 12\alpha R^{\max}(\tilde{U}_W + \frac{2\sum_l x_l R^{\max}T}{\alpha\bar{\lambda}} + (\lambda_W^{\max} + 2)T)}{\epsilon \min_l x_l}. \tag{2.25}$$

Since the changes of $W_s(t)$ and $Q_l(t)$ during each time slot is bounded by some constants independent of $\tilde{\mathbf{M}}(n)$, it is easy to verify that $\tilde{\Upsilon}$ is a set of a finite number of elements.

Next, we analyze the drift of Lyapunov function case by case assuming that

$$D > \left\lceil \frac{\log\bar{\lambda}\epsilon - \log 16 - \log R^{\max}}{\log(1 - p_s^{\max})} \right\rceil \tag{2.26}$$

and $T > \left\lceil \frac{(4+\epsilon)D}{\epsilon} \right\rceil$.

- **Case I:** Assume that $\tilde{M}(n) \in \tilde{\Upsilon}$. In this case, it is easy to verify that $\mathbf{E}[V(n+1) - V(n) | \tilde{\mathbf{M}}(n)]$ is bounded by some constant $\tilde{U}_d$.

- **Case II:** Assume that

$$W_s(nT) > \tilde{U}_W + 2T + \frac{2\sum_l x_l R^{\max}T}{\alpha\bar{\lambda}} \geq \tilde{U}_W + T.$$

Recall that $\mathcal{E}_{miss}(t)$ is the event such that the tie-breaking rule selects a short-lived flow with $\tilde{R}_i^{\max}(t) \neq R_i^{\max}$. Note that $\mu_{2;s}(t) = 0$ implies that $\mathcal{E}_{miss}(t)$ occurs. Also note the following facts:

- For any $nT \leq t \leq (n+1)T$, we have $W(t) \leq W(nT) + \lambda_W^{\max} T$,
- Given $W_s(nT) \geq \tilde{U}_W + T$, we have $W_s(t) \geq \tilde{U}_W$ for all $nT \leq t \leq (n+1)T - 1$. Then according to the definition of ϵ_2 and $\tilde{U}_W$ and assumption that the tie-breaking rule is good, we have

$$\Pr\left(\mathcal{E}_{miss}(t)\right) \leq \epsilon_2$$

for all $nT + D \leq t \leq (n+1)T - 1$.

- Given any $\tilde{\mathbf{M}}(n)$ and any $nT + D \leq t \leq (n+1)T - 1$, we have

$$\begin{aligned}
&\mathbf{E}\left[\alpha \tilde{W}_s(t) - \alpha W_s(t) | \text{Situ-C}, \mu_{2;s} = 1, \tilde{\mathbf{M}}(n)\right] \times \\
&\Pr\left(\text{Situ-C}, \mu_{2;s} = 1 | \tilde{\mathbf{M}}(n)\right) \\
\leq & \mathbf{E}\left[\alpha \tilde{W}_s(t) - \alpha W_s(t) | \tilde{\mathbf{M}}(n)\right] \\
= & \mathbf{E}\left[\mathbf{E}\left[\alpha \tilde{W}_s(t) - \alpha W_s(t) \middle| W_s(t-D)\right] \middle| \tilde{\mathbf{M}}(n)\right] \\
\leq & \mathbf{E}\left[\alpha (1 - p_s^{\max})^D W_s(t-D) R^{\max} + \alpha \lambda_W^{\max} D | \tilde{\mathbf{M}}(n)\right] \qquad (2.27) \\
\leq & \mathbf{E}\left[\alpha (1 - p_s^{\max})^D (W_s(t) + D) R^{\max} + \alpha \lambda_W^{\max} D | \tilde{\mathbf{M}}(n)\right],
\end{aligned}$$

where the inequality (2.27) holds because at most $\lambda_W^{\max} D$ bits belonging to short-lived flows are in the network for less than D time slots at time t, and a flow having been in the network for at least D time slots can estimate correctly its workload with a probability at least $1 - (1 - p_s^{\max})^D$.

Now according to the observations above, we can obtain that

$$\begin{aligned}
&\mathbf{E}[\text{Diff}(t) | \tilde{\mathbf{M}}(n)] \\
\leq & \epsilon_2 \alpha \left(W_s(nT) + \lambda_W^{\max} T\right) + \epsilon_2 \alpha \left(R^{\max} W_s(nT) + \lambda_W^{\max} T\right) \\
&+ \mathbf{E}\left[\alpha (1 - p_s^{\max})^D (W_s(t) + D) R^{\max} + \alpha \lambda_W^{\max} D | \tilde{\mathbf{M}}(n)\right].
\end{aligned}$$

Combining with the analysis leading to (2.13) in the proof of Theorem 2, we conclude that

$$\begin{aligned}
&\mathrm{Drift}(t)\\
\le & 2\mathbf{E}\Bigg[\epsilon_1\left(\alpha W_s(t) + R^{\max}\max_{l\in\mathcal{L}} Q_l(t)\right)\\
&-\epsilon\alpha W_s(t)\bar{\lambda} - \epsilon\sum_{l\in\mathcal{L}} Q_l(t)x_l\\
&+\epsilon_2\alpha\left(W_s(nT) + \lambda_W^{\max}T\right)\\
&+\epsilon_2\alpha\left(R^{\max}W_s(nT) + \lambda_W^{\max}T\right)\\
&+\alpha(1-p_s^{\max})^D(W_s(t)+D)R^{\max} + \alpha\lambda_W^{\max}D|\tilde{\mathbf{M}}(n)\Bigg]\\
\le & \mathbf{E}\left[-\epsilon\left(\alpha\bar{\lambda}W_s(t) + \sum_{l\in\mathcal{L}} x_l Q_l(t)\right)\Bigg|\tilde{\mathbf{M}}(n)\right],
\end{aligned}$$

where the last inequality holds due to (2.23).

- **Case III:** Assume that

$$W_s(nT) < \tilde{U}_W + 2T + \frac{2\sum_l x_l R^{\max}T}{\alpha\bar{\lambda}}$$

and

$$Q_l(nT) > \tilde{U}_Q + \frac{2\alpha\bar{\lambda}T}{\min_l x_l} + \frac{2TR^{\max}\sum_l x_l}{\min_l x_l} > \tilde{U}_Q$$

for some l. In this case, we have

$$\mathrm{Diff}(t) \le \alpha\tilde{W}_s(t) \le \alpha R^{\max}W_s(t).$$

Combining with the analysis leading to (2.15) in the proof of Theorem 2, we have that

$$\begin{aligned}
&\mathrm{Drift}(t)\\
\le & 2\mathbf{E}\Bigg[\alpha R^{\max}W_s(t) + 2\alpha W_s(t)\\
&-\epsilon\left(\alpha\bar{\lambda}W_s(t) + \sum_{l\in\mathcal{L}} x_l Q_l(t)\right)\Bigg|\tilde{\mathbf{M}}(n)\Bigg]\\
\le & \mathbf{E}\left[-\epsilon\left(\alpha\bar{\lambda}W_s(t) + \sum_{l\in\mathcal{L}} x_l Q_l(t)\right)\Bigg|\tilde{\mathbf{M}}(n)\right],
\end{aligned}$$

where the last inequality holds due to (2.25).

□

Now, combining case II and case III, we can obtain that

$$
\begin{aligned}
&\mathbf{E}[V(n+1)-V(n)|\tilde{\mathbf{M}}(n)]\\
\leq&\tilde{U}+2\alpha W_s(nT)\bar{\lambda}D+2\sum_{l\in\mathcal{L}}Q_l(nT)x_lD\\
&+\sum_{t=nT+D}^{(n+1)T-1}\mathbf{E}\left[-\epsilon\left(\alpha\bar{\lambda}W_s(t)+\sum_{l\in\mathcal{L}}x_lQ_l(t)\right)\middle|\tilde{\mathbf{M}}(n)\right]\\
\leq&\tilde{U}+2\alpha W_s(nT)\bar{\lambda}D+2\sum_{l\in\mathcal{L}}Q_l(nT)x_lD\\
&-\epsilon(T-D)\left(\alpha\bar{\lambda}W_s(nT)+\sum_{l\in\mathcal{L}}x_lQ_l(nT)\right)\\
&+\epsilon(T-D)(\alpha\bar{\lambda}T+\sum_{l\in\mathcal{L}}x_lR^{\max}T)\\
\leq&-\tilde{U}-\sum_{t=nT+D}^{(n+1)T-1}\mathbf{E}\left[\frac{\epsilon}{2}\left(\alpha\bar{\lambda}W_s(t)+\sum_{l\in\mathcal{L}}x_lQ_l(t)\right)\middle|\tilde{\mathbf{M}}(n)\right],
\end{aligned}
$$

where the last inequality yields from the definition of $\tilde{U}_W$ and $\tilde{U}_Q$. Finally, we can conclude the theorem from [12, 13].

2.6 Simulations

In this section, we use simulations to evaluate the performance of different variants of WSL and compare it to other scheduling policies. There are three types of flows used in the simulations:

- **S-flow:** An S-flow has a finite size, generated from a truncated exponential distribution with mean value 30 (before truncation) and maximum value 150. Non-integer values are rounded to integers.
- **M-flow:** An M-flow keeps injecting bits into the network for 10,000 time slots and stops. The number of bits generated at each time slot follows a Poisson distribution with mean value 1.

- **L-flow:** An L-flow keeps injecting bits into the network and never leaves the network. The number of bits generated at each time slot follows a truncated Poisson distribution with mean value 1 (before truncation) and maximum value 10.

Here S-flows represent short-lived flows that have finite sizes and whose bits arrive all at once; L-flows represent long-lived flows that continuously inject bits and never leave the network; and M-flows represent flows of finite size but whose arrival rate is controlled at their sources so that they do not arrive instantaneously into the network. Our simulation will demonstrate the importance of modeling very large, but finite-sized flows as long-lived flows.

We assume that the channel between each user and the base station is distributed according to one of the following three distributions:

- **G-link:** A G-link has five possible link rates $\{10, 20, 30, 40, 50\}$, and each of the states happens with probability 20%.
- **P-link:** A P-link has five possible link rates $\{5, 10, 15, 20, 25\}$, and each of the states happens with probability 20%.
- **R-link:** An R-link has five possible link rates $\{10, 20, 30, 40, 100\}$, and the probabilities associated with these link states are $\{0.5, 0.2, 0.2, 0.09, 0.01\}$.

The G, P and R stand for Good, Poor and Rare, respectively. We include these three different distributions to model the SNR variations among the users, where G-links represent links with high SNR (e.g., those users close to the base station), P-links represent links with low SNR (e.g., those users far away from the base station), and R-links represent links whose best state happens rarely. The R-links will be used to study the impact of learning period D on the network performance.

We name the WSL with the uniform tie-breaking rule WSLU, and the WSL with the oldest-first tie-breaking rule WSLO. In the following simulations, we will first demonstrate that the WSLU performs significantly better than previously suggested algorithms, and then show that the performance can be further improved by choosing a good tie-breaking policy (e.g., WSLO). We set α to be 50 in all the following simulations.

Simulation I: Short-lived Flow or Long-lived Flow?

We first use the simulation to demonstrate the importance of considering a flow with a large number of packets as being long-lived. We consider a network consisting of multiple S-flows and three M-flows, where the arrival of S-flows follows a truncated Poisson process with maximum value 100 and mean value λ. All the links are assumed to be G-links. We evaluate the following two schemes:

- **Scheme-1:** Both S-flows and M-flows are considered to be short-lived flows.
- **Scheme-2:** An M-flow is considered to be long-lived before its last packet arrives, and to be short-lived after that.

The performance of these two schemes are shown in Figure 2.1, where WS with Uniform Tie-breaking Rule is used as the scheduling algorithm. We can see that the performances are substantially different (note that the network is stable under both schemes). The number of queued bits of M-flows under Scheme-1 is larger than that under Scheme-2 by *two orders of magnitude.* This is because even an M-flow contains a huge number of bits ($10,000$ on average), it can be served only when the link rate is 50 under Scheme-1. This simulation suggests that when the performance we are interested is at a small scale (e.g. acceptable queue-length being less than or equal to 100) compared with the size of the flow (e.g., 10^4 in this simulation), the flow should be viewed as a long-lived flow for performance purpose.

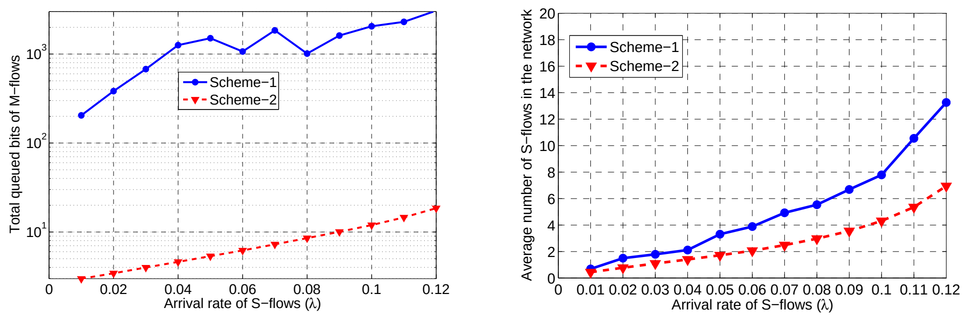

Figure 2.1 Scheme-1 treats M-flows as short-lived flows, and Scheme-2 treats M-flows as long-lived flows

Simulation II: The Impact of Learning Period D

In this simulation, we investigate the impact of D on the performance of WSLU. Recall that it is nature to choose $D = \infty$ for purely throughput-optimality considerations, but the disadvantage is that a flow may stay in the network for a very long time if the best link state occurs very rarely. We consider a network consisting of S-flows, which arrive according to a truncated Poisson process with maximum value 100 and mean λ, and three L-flows. All links are assumed to be R-links. Figure 2.2 depicts the mean and standard deviation of the file-transfer delays with $D = 16$ and $D = \infty$ when the traffic load is light or medium. As we expected, the standard deviation under WSLU with $D = \infty$ is significantly larger than that under WSLU with $D = 16$ when λ is large. This occurs because the best link rate 100 occurs with a probability 0.01. This simulation confirms that in practical systems, we may want to choose a finite D to get desired performance.

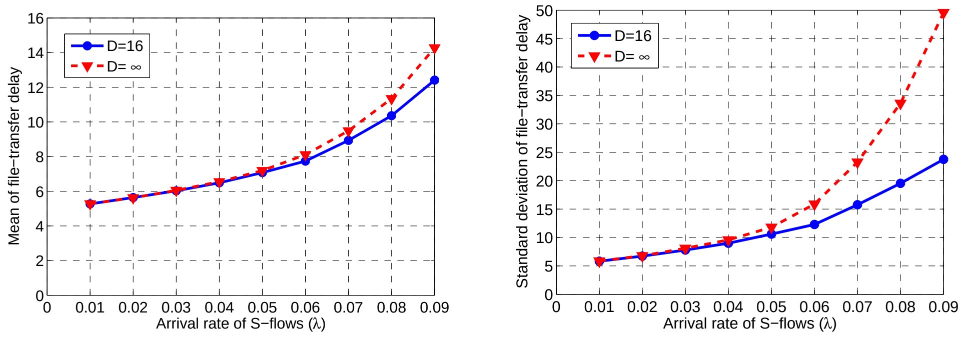

Figure 2.2 The performance of WSLU with $D = 16$ and $D = \infty$ when the traffic load is light or medium

Further we would like to comment that while the WSLU algorithm with a small D has a better performance in light or medium traffic regimes, throughput optimality is only guaranteed when D is sufficiently large. So there is a clear tradeoff in choosing D: A small D reduces the file-transfer delay in light or medium traffic regimes, but a large D guarantees stability in heavy traffic regime. More discussion can be found in [16].

Simulation III: Performance comparison of various algorithms

In the following simulations, we choose $D = 16$. In the introduction, we have pointed out that the MaxWeight is not throughput optimal under flow-level dynamics because the backlog of a short-lived queue does not build up even when it has not been served for a while. To overcome this, one could try to use the delay of the head-of-line packet, instead of queue-length, as the weight because the head-of-line delay will keep increasing if no service is received. In the case of long-lived flows only, this algorithm is known to be throughput-optimal [5]. We will show that this Delay-based scheduling does not solve the instability problem when there are short-lived flows.

Delay-based Scheduling: At each time slot, the base station selects a flow i such that $i \in \arg\max_j D_j(t)R_j(t)$, where $D_j(t)$ is the delay experienced so far by the head-of-line packet of flow j.

We first consider the case where all flows are S-flows, which arrive according to a truncated Poisson process with maximum value 100 and mean λ. An S-flow is assigned with a G-link or a P-link equally likely.

Figure 2.3 shows the average file-transfer delay and average number of S-flows under different values of λ. We can see that WSLU performs significantly better than the MaxWeight and Delay-based algorithms. Specifically, under MaxWeight and Delay-based algorithms, both the number of S-flows and file-transfer delay explode when $\lambda \geq 0.102$. WSLU, on the other hand, performs well even when $\lambda = 0.12$.

Next, we consider the same scenario with three L-flows in the network. Two of the L-flows have G-links and one has a P-link. Figure 2.4 shows the average number of short-lived flows and average file-transfer delay under different values of λ. We can see that the MaxWeight becomes unstable even when *the arrival rate of S-flows is very small.* This is because the MaxWeight stops serving S-flows when the backlogs of L-flows are large, so S-flows stay in the network forever. The delay-based scheduling performs better than the MaxWeight, but significantly worse than WSLU.

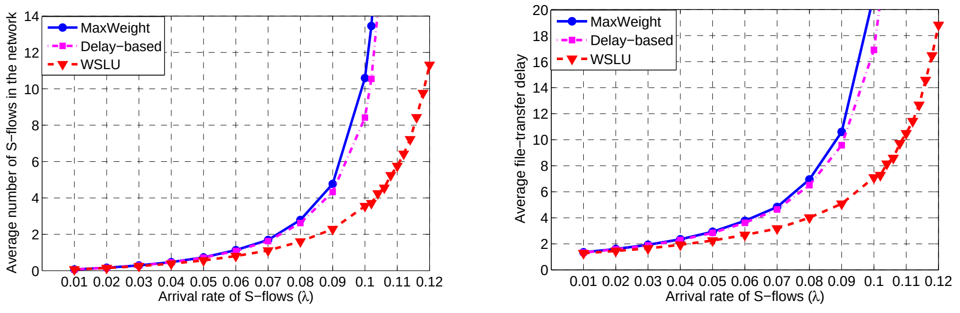

Figure 2.3 The performance of the Delay-based, MaxWeight, and WSLU algorithms in a network without L-flows

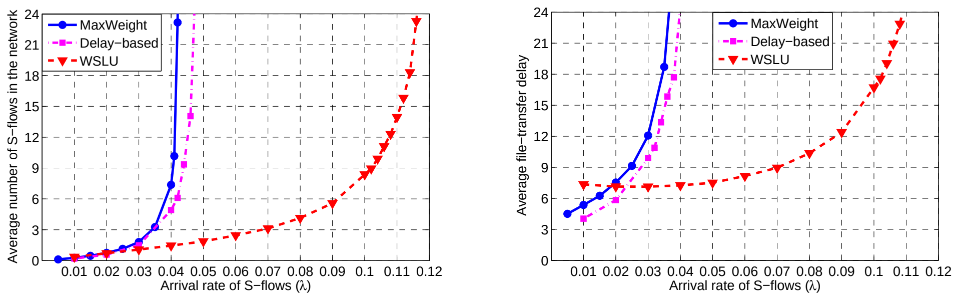

Figure 2.4 The performance of the Delay-based, MaxWeight, and WSLU algorithms in a network with both S-flows and L-flows

Simulation IV: Blocking probability of various algorithms

While our theory assumes that the number of flows in the network can be infinite, in reality, base stations limit the number of simultaneously active flows, and reject new flows when the number of existing flows above some threshold. In this simulation, we assume that the base station can support at most 20 S-flows. A new S-flow will be blocked if 20 S-flows are already in the network. In this setting, the number of flows in the network is finite, so we compute the blocking probability, i.e., the fraction of S-flows rejected by the base station.

We consider the case where no long-lived flow is in the network and the case where both short-lived and long-lived flows are present in the network. The flows and channels are selected as in Simulation III. The results are shown in Figure 2.5 and 2.6. We can see that the blocking probability under WSLU is substantially smaller than that under the MaxWeight or the delay-based scheduling. Thus, this simulation demonstrates that instability under the assumption when the number of flows is allowed to unbounded implies high blocking probabilities for the practical scenario when the base station limits the number of flows in the network.

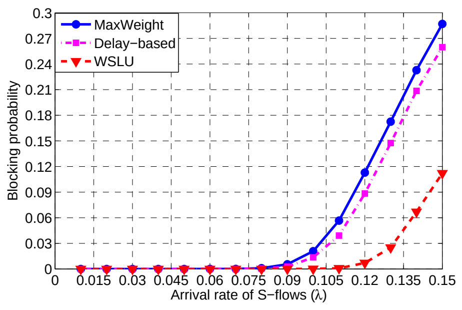

Figure 2.5 The blocking probabilities of the Delay-based, MaxWeight, and WSLU in a network without L-flows

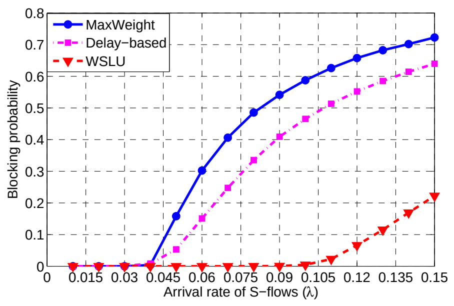

Figure 2.6 The blocking probabilities of the Delay-based, MaxWeight, and WSLU in a network with L-flows

Simulation V: WSLU versus WSLO

In this simulation, we study the impact of tie-breaking rules on performance. We compare the performance of the WSLU and WSLO. We first study the case where the base station does not limit the number of simultaneously active flows and there is no long-lived flow in the network. The simulation setting is the same as that in Simulation III. Figure 2.7 shows the average file-transfer delay and average number of S-flows under different values of λ. We can see that the WSLO reduces the file-transfer delay and number of S-flows by nearly 75% when

$\lambda = 0.13$, which indicates the importance of selecting a good tie-breaking rule for improving the network performance.

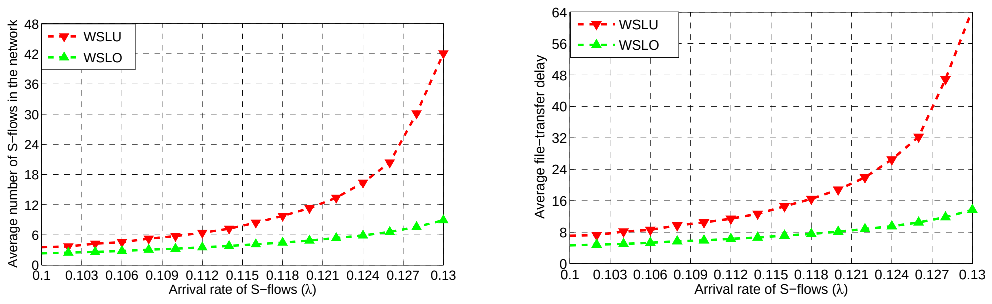

Figure 2.7 The performance of the WSLU and WSLO algorithms in a network without L-flows

Next, we study the case where the base station does not limit the number of simultaneously active flows and there are three L-flows in the network. Figure 2.8 shows the average number of short-lived flows and average file-transfer delay under different values of λ. We can see again that the WSLO algorithm has a much better performance than the WSLU, especially when λ is large.

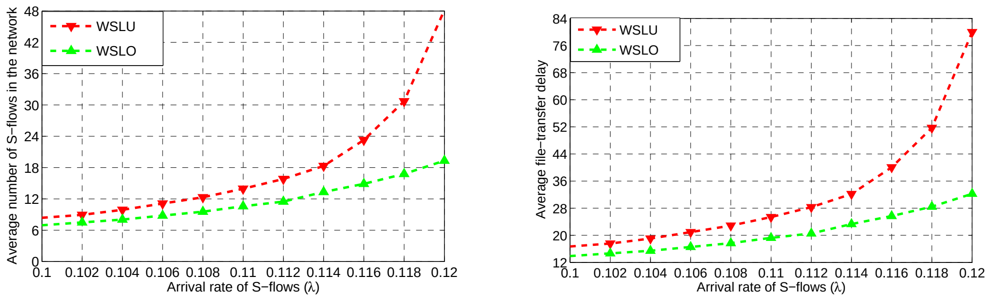

Figure 2.8 The performance of the WSLU and WSLO algorithms in a network with both S-flows and L-flows

Finally we consider the situation where the base station can support at most 20 S-flows. A new S-flow will be blocked if 20 S-flows are already in the network. The simulation setting is the same as that in Simulation IV. We calculate the blocking probabilities, and the results are shown in Figure 2.9 and 2.10. We can see that the blocking probability under the WSLO is much smaller than that under the WSLU policy when λ is large.

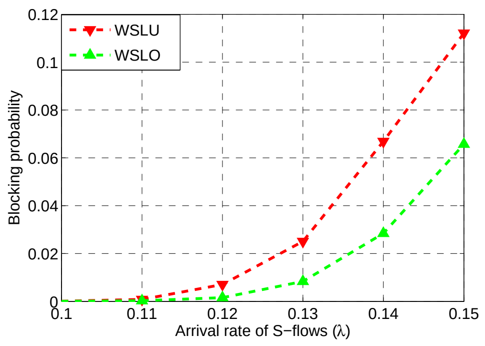

Figure 2.9 The blocking probabilities of the WSLU and WSLO in a network without L-flows

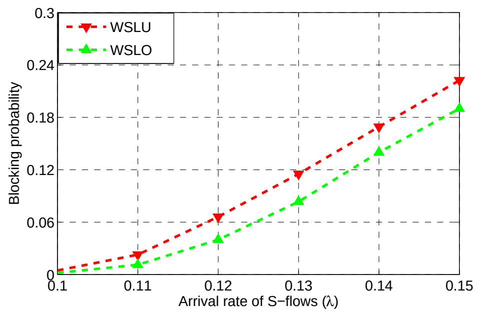

Figure 2.10 The blocking probabilities of the WSLU and WSLO in a network with L-flows

2.7 Conclusions and Discussions

In this chapter, we studied multiuser scheduling in networks with flow-level dynamics. We first obtained necessary conditions for flow-level stability of networks with both long-lived flows and short-lived flows. Then based on an optimization framework, we proposed the workload-based scheduling with learning that is throughput-optimal under flow-level dynamics and requires no prior knowledge about channels and traffic. In the simulations, we evaluated the performance of the proposed scheduling algorithms, and demonstrated that the proposed algorithm performs significantly better than the MaxWeight algorithm and the Delay-based algorithm in various settings. Next we discuss the limitations of our model and possible extensions.

2.7.1 The Choice of D

According to Theorem 3, the learning period D should be sufficiently large to guarantee throughput-optimality. Our simulation results on the other hand suggested that a small D may result in better performance. Therefore, there is clear trade-off in choosing D. The study of the choice for D is one potential topic for future work.

2.7.2 Unbounded File Arrivals and File Sizes

One limitation of our model is that the random variables associated with the number of file arrivals and file sizes are assumed to be upper bounded. One interesting future research problem is to extend the results to unbounded number of file arrivals and file sizes.

CHAPTER 3. Scheduling in OFDM-based Wireless Cellular Networks

In this chapter, I present the results on scheduling in OFDM-based wireless cellular networks with flow-level dynamics. Note that in an OFDM-based system, the base station has multiple frequency bands (channels) for the downlink transmission. The problem is more complicated than the single-channel case because in multichannel scheduling, we need to handle two problems. The first one is "channel assignment" for each flow, i.e., which channels should be used to serve the flow. The second problem is "scheduling", i.e., for each channel, which flow to serve at each time slot. For the second problem, the single-channel scheduling algorithm (i.e., the workload-based scheduling) provides a good intuition: exploit good channel rates as much as possible. Based on this intuition, we propose a joint channel-assignment and scheduling algorithm, which solves the multichannel scheduling problem. Note that the model we use in this part is slightly different from the single-channel case, and contains two types of flows, transient flows and resident flows. A flow is called a transient flow if the last packet of the file has not arrived at the base station; and otherwise, we call the flow a resident flow. Despite this minor difference, the model is consistent with the one in the single-channel case in the sense that a transient flow corresponds to a long-lived flow, and a resident flow corresponds to a short-lived flow.

3.1 Basic Model

In this section, we define the network, channel and traffic models that will be used in this chapter.

Network model: We consider a wireless downlink network with multiple channels (frequency bands). We let $\mathcal{M}$ denote the set of channels and let $M = |\mathcal{M}|$. The network consists of a single base station and multiple flows (mobile users). The flows join the network for the purpose of receiving files from some remote source which is not modeled in our framework, and leave the network after downloading the complete file. The remote source transmits the file to the base-station, and then the base-station transmits to the mobile user. The base station can communicate with a mobile user using any of the M channels. We assume time is slotted, and that at each time slot, only one flow can be served over a given channel (frequency band) but a flow can be served by multiple channels simultaneously. A two-channel, three-mobile downlink network is demonstrated in Figure 3.1.

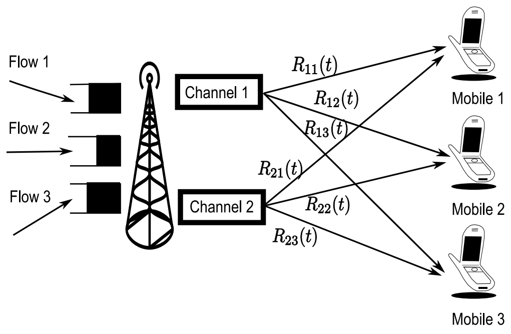

Figure 3.1 A two-channel, three-mobile downlink network

Channel model: We denote by $R_{if}(t)$ the state of channel i seen by flow f in time slot t, i.e., $R_{if}(t)$ denotes the number of packets that can be served by the channel at time instant t. We assume that $R_{if}(t)$ are a sequence of independent random variables (across time slots and across users), each distributed like some random variable $\mathfrak{R}_{if}$, where $\mathfrak{R}_{if}$ has a finite support. We denote by $R_{if}^{\max}$ the largest possible value of $\mathfrak{R}_{if}$ and $\mathbf{R}_f^{\max} = \left(R_{1f}^{\max}, \ldots, R_{Mf}^{\max}\right)$. We assume that there exists $p^{\max} > 0$ such that

$$\Pr(R_{if}(t) = R_{if}^{\max}) \geq p^{\max}$$

for all i, f and t.

Traffic model: We denote by $\tilde{F}_f$ the size of the file associated with flow f and assume $\tilde{F}_f$ are a sequence of independent random variables (across flows), each distributed like a random variable $\mathfrak{F}$. Thus, $\tilde{F}_f$ is the number of packets in flow f's file. We classify flows into different classes according to the maximum-rate vector $\mathbf{R}_f^{\max}$ seen by them. So flows f_1 and f_2 belong to the same class if $R_{if_1}^{\max} = R_{if_2}^{\max}$ for all i. We let $\mathcal{K}$ denote the set of classes, and assume $K = |\mathcal{K}|$. We further denote by k_f the class of flow f and $\Lambda_{kF}(t)$ the number of class-k flows that have a size of F and join the network at time t. We assume $\Lambda_{kF}(t)$ are a sequence of independent random variables (across time slots), each distributed like Λ_{kF}, and $\lambda_{kF} = \mathbf{E}[\Lambda_{kF}]$. We further assume that the size of a file is upper bounded by $F^{\max}$ and

$$\sum_{k\in\mathcal{K}, F\leq F^{\max}} \Lambda_{kF}(t) \leq \lambda^{\max}$$

for any t. Finally, we denote by $F_f(t)$ the number of packets of flow f queued at the base station at time t, and $\mathcal{F}(t)$ the set of flows in the network at time t.

A flow is called a *transient flow* if the last packet of the file has not arrived at the base station; and otherwise, we call the flow a *resident flow.* In this chapter, we assume that the base station knows when a file is completely transferred to the base station (e.g., the base station can figure out if a flow is a resident flow by looking for a special end-of-file packet). We let b_f denote the time flow f joins the network, and s_f the time flow f becomes a resident flow. We further denote by $\mathcal{L}(t)$ the set of transient flows at time t, and $\mathcal{S}(t)$ the set of resident flows at time t.

3.2 Joint Channel Assignment and Workload Based Scheduling

For single-channel networks in the presence of flow-level dynamics, throughput-optimal scheduling algorithms have been proposed in [1, 2, 16]. The key idea of these algorithms is to minimize the number of time-slots used to serve all traffic flows. Note that the minimum number of time slots required to fully transmit a file f is $\left\lceil \tilde{F}_f/R_f^{\max} \right\rceil$, where $R_f^{\max}$ is the best channel state seen by flow f and $\left\lceil \tilde{F}_f/R_f^{\max} \right\rceil$ is called the workload of flow f. So the idea is then to serve a flow f only when $R_f(t) = R_f^{\max}$, in other words, serve a flow only if the workload of that flow can be reduced by one. Since the average workload injected into the network in one time slot should be less than one given the traffic load is within the throughput region, scheduling algorithms that reduce workload by one (with a high probability) during each time slot stabilize the network.

The reader may wonder whether we can directly use this workload-based approach to multichannel networks? *For example to be throughput-optimal, is it sufficient to serve on each channel i a flow such that $R_{if}(t) = R_{if}^{\max}$?* The answer unfortunately is negative, as shown in the following example.

Example: Consider a network with two channels with constant service rates: $R_{1f} = B+1$ and $R_{2f} = 2B$ for all f, and two types of flows in the network: the file size of a type 1 flow is $2B + 2$ and the file size of a type 2 flow is $4B$. We assume $B \geq 4$ and both types of flows arrive with a constant rate $1/2$, i.e., one new arrival every two time slots.

Under this setting, consider a channel assignment that serves type 1 flows on channel 1 and type 2 flows on channel 2. Since each flow consumes two channel uses under this channel assignment, the network is stable.

However, we will now show that throughput optimality is not guaranteed by serving on each channel i a flow with $R_{if}(t) = R_{if}^{\max}$. For this purpose, consider *a scheduling policy which gives priority to type 2 flows on channel 1 and priority to type 1 flows on channel 2.*

Note that each type 1 flow requires two channel uses, irrespective of the channels assigned to it, so channel 2 is fully occupied by type 1 flows with arrival rate $1/2$. Each type 2 flow requires four channel uses on channel 1, so channel 1 alone cannot support type 2 flows with arrival rate $1/2$. However, since channel 2 is fully occupied by type 1 flows, the number of type 2 flows will build up and the network is unstable. While this example considers deterministic arrivals for simplicity, it is not difficult to construct an example with stochastic arrivals to demonstrate the lack of throughput optimality of a policy which schedules a user with the best channel state on each channel.

From this example, we can see the direct adoption of the workload-based algorithm for single channel networks may not be throughput-optimal for multichannel networks. This is because, in a multichannel network, a flow can be served by more than one channel, and the channels may have different best channel states. Therefore, to achieve the maximum throughput, we need to intelligently split a flow among the M channels. In the previous example, the optimal solution is to assign all type-1 flows to channel 1 and all type-2 flows to channel 2. Now to develop efficient channel-assignment algorithms, our first step is to understand the throughput region of a multichannel network.

3.2.1 Necessary Conditions for Stability

To describe necessary conditions for supportability, we introduce the concept of a channel assignment vector $\mathbf{h}$. Associated with each flow is a channel assignment vector $\mathbf{h} = (h_1, h_2, ..., h_M)$, where h_i denotes the number of time slots allocated to the flow on channel i. The parameter h_i can be viewed as the workload imposed by the flow on channel i. For example, in a network with three channels, $\mathbf{h}_f(t) = (0, 1, 1)$ means that after time slot t, the base station is allowed to serve flow f once (one time slot) over channel 2 and 3, but not allowed to serve the flow over channel 1. Next we define $Q_i(t) = \sum_{f \in \mathcal{S}(t)} h_{if}(t)$, where $h_{if}(t)$ is the i^{th} element of vector $\mathbf{h}_f(t)$. Now given arrival rates $\{\lambda_{kF}\}$, we say that $\{\lambda_{kF}\}$ is *supportable* if there exists a scheduling algorithm, under which

$$\lim_{t\to\infty} \mathbf{E}\left[\sum_i Q_i(t)\right] < \infty$$

holds.

Further, if a flow has F packets, then we only need to consider channel assignment vectors such that $\sum_{i \in \mathcal{M}} h_i \leq F$. If a traffic load is supportable, then the average rate at which workload arrives on each channel should be less than one, so we obtain the following necessary conditions for supportability.

Lemma 4. *If arrival rates $\{\lambda_{kF}\}$ are supportable, then there exist $Z^*_{\mathbf{h}|kF} \geq 0$ such that*

$$\begin{aligned} \sum_{k,F,\mathbf{h}} \lambda_{kF} Z^*_{\mathbf{h}|kF} h_i &\leq 1, i = 1, 2, ..., M &(3.1)\\ \sum_{\mathbf{h}} Z^*_{\mathbf{h}|kF} &= 1, \forall k, F &(3.2)\\ Z^*_{\mathbf{h}|kF} &= 0 \textit{ if } F > \sum_{i\in\mathcal{M}} h_i R^{\max}_{ik} &(3.3)\end{aligned}$$

□

We comment that $Z^*_{\mathbf{h}|kF}$ can be viewed as the fraction of class k flows of size F that are assigned the channel assignment vector $\mathbf{h}$. Inequality (3.1) is the capacity constraint which says that the average workload assigned to a channel should be less than one. Inequality (3.2) states that every file should be associated with a channel assignment vector, and (3.3) states that considering a flow f, the channel assignment vector should guarantee sufficient service for transmitting the complete file. We do not provide a proof of Lemma 4. The proof follows along the lines of similar proofs in [14] or [5].

3.2.2 Joint Channel-Assignment and Workload-based Scheduling

Now based on the necessary conditions (Lemma 4), we derive an *on-line* channel-assignment algorithm using an optimization based approach. Consider the following optimization problem (feasibility problem):

$$\begin{aligned} &\min_{\mathbf{Z}} 0 \\ \sum_{k,F,\mathbf{h}} \lambda_{kF} Z_{\mathbf{h}|kF} h_i &\le 1, i = 1, 2, ..., M \\ \sum_{\mathbf{h}} Z_{\mathbf{h}|kF} &= 1, \forall k, F \\ Z_{\mathbf{h}|kF} &= 0 \text{ if } F > \sum_{i \in \mathcal{M}} h_i R_{ik}^{\max} \\ Z_{\mathbf{h}|kF} &\ge 0, \ \forall \mathbf{h}, k, F \end{aligned}$$

By appending some of the constraints to the objective using Lagrangian multipliers, we get:

$$\begin{aligned} &\min_{\mathbf{Z}} \textstyle\sum_i Q_i \left(\sum_{k,F,\mathbf{h}} \lambda_{kF} Z^*_{\mathbf{h}|kF} h_i - 1 \right) \\ \text{subject to:} \quad & \textstyle\sum_{\mathbf{h}} Z_{\mathbf{h}|kF} = 1, \forall k, F \\ & Z_{\mathbf{h}|kF} = 0 \text{ if } F > \textstyle\sum_{i \in \mathcal{M}} h_i R_{ik}^{\max} \\ & Z_{\mathbf{h}|kF} \ge 0, \forall \mathbf{h}, k, F, \end{aligned}$$

where Q_i is the Lagrangian multiplier associated with constraint $\sum_{k,F,\mathbf{h}} \lambda_{kF} Z_{\mathbf{h}|kF} h_i \le 1$.

The partially augmented problem can be decomposed into subproblems associated with each pair of k and F :

$$\min_{\mathbf{Z}} \sum_{\mathbf{h}} \sum_i Q_i Z_{\mathbf{h}|kF} h_i$$

$$\text{subject to:} \qquad \sum_{\mathbf{h}} Z_{\mathbf{h}|kF} = 1,$$

$$Z_{\mathbf{h}|kF} = 0 \text{ if } F > \sum_{i \in \mathcal{M}} h_i R_{ik}^{\max}$$

$$Z_{\mathbf{h}|kF} \geq 0, \ \forall \mathbf{h}, k, F.$$

Since the objective function is linear, the subproblem (for fixed k and F) can be further written as:

$$\min_{\mathbf{h}} \sum_i Q_i h_i$$

$$\text{subject to:} \quad F \leq \sum_{i \in \mathcal{M}} h_i R_{ik}^{\max}.$$

Therefore for each flow, the channel-assignment problem can be written as:

$$\min_{\mathbf{h}} \sum_i Q_i h_i$$

$$\text{subject to:} \quad \tilde{F}_f \leq \sum_{i \in \mathcal{M}} h_i R_{if}^{\max}.$$

Recall that Lagrangian multipliers can be viewed as the price for using a given resource. Thus, if the Lagrangian multipliers are given, the channel assignment problem becomes a load balancing problem in which channel assignment is performed to minimize a weighted sum of channel prices. To compute the channel prices, we use the well-known intuition that the Lagrange multipliers are proportional to queue lengths. Note that the Lagrangian multiplier Q_i is associated with the constraint $\sum_{k,F,\mathbf{h}} \lambda_{kF} Z_{\mathbf{h}|kF} h_i \leq 1$, where $\sum_{k,F,\mathbf{h}} \lambda_{kF} Z_{\mathbf{h}|kF} h_i$ is the average incoming workload during each time slot. Thus, the natural queue to consider here is the overall workload assigned to channel i that has not yet been served by the network. We describe it more precisely next.

For each flow f, we define a channel assignment vector at time slot t

$$\mathbf{h}_f(t) = (h_{1f}(t), h_{2f}(t), ..., h_{Mf}(t)),$$

where $h_{if}(t)$ is the number of remaining time slots assigned to serve flow f over channel i at time slot t.

We then use $Q_i(t)$ as an estimate of the Lagrangian multiplier Q_i and propose the following algorithm.

Joint Channel-Assignment and Workload-based Scheduling (CA-WS):

(i) **Channel-assignment:** When the last packet of flow f is received at the base station (at time slot s_f),[1] the base station computes $\mathbf{h}_f(s_f)$ by solving the following optimization problem:

$$\begin{aligned} \mathrm{OPT}_f = \quad & \min \textstyle\sum_{i \in \mathcal{M}} Q_i(b_f) h_{if} \\ \text{subject to:} \quad & \tilde{F}_f \leq \textstyle\sum_{i \in \mathcal{M}} h_{if} R_{if}^{\max}, \end{aligned}$$

where h_{if} are non-negative integers. Clearly $\mathbf{h}_f(t) = 0$ for $t < s_f$.

(ii) **Workload-based scheduling:** At time slot t, the base station selects a file f for channel i such that

$$R_{if}(t) = R_{if}^{\max} \text{ and } h_{if}(t) > 0, \tag{3.4}$$

and transmits $R_{if}(t)$ packets to mobile user f. Then, the base station reduces $h_{if}(t)$ by one. If no flow satisfies (3.4), the base station randomly selects a flow, say flow f, and transmits $R_{if}(t)$ packets to mobile f (in this case, $h_{if}(t)$ is not updated). Ties are broken arbitrarily. When the file f has been completely transmitted to mobile f, the base station sets $h_{if}(t) = 0$ for all i.

□

Theorem 5. *Assume that $s_f - b_f \leq T_{in}$ for all f. Given arrival rates $\{\lambda_{kF}\}$ such that $\{\frac{1}{1-\epsilon}\lambda_{kF}\}$ are supportable, the CS-WS algorithm guarantees*

$$\lim_{t \to \infty} \mathbf{E}\left[\sum_i Q_i(t)\right] < \infty.$$

[1]Here for simplicity we only consider the case where a flow can be served only after its last packet arrives at the system. Later we will consider a more general case where a flow may be served before the arrival of its last packet.

Proof. The proof of this theorem follows from the proof of Theorem 8 to be presented in the next section. Therefore, we omit the details here. □

Remark: The theorem assumes $s_f - b_f \leq T_{in}$, which means that the injection period of a flow (the time duration from the first packet arrives at the base station to the last packet arrives at the base station) is bounded by T_{in}. For example, if the flow is a constant-bit-rate flow with rate r, then the injection period is upper bounded by $F^{\max}/r$; and if the flow is an elastic flow whose rate is controlled by congestion control algorithm, then the injection period is also bounded when the injection rate is lower bounded as in the TCP congestion control algorithm (e.g., at least one packet over a fixed number of time slots).

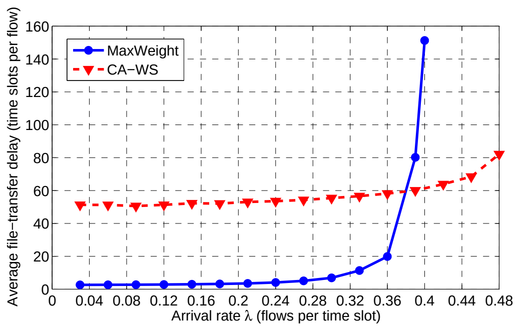

Figure 3.2 Average file-transfer delay of the CA-WS and MaxWeight algorithms

Theorem 5 shows that the CA-WS algorithm is throughput-optimal for multichannel downlink networks, but the algorithm has two weaknesses:

- The performance of the algorithm can be poor in light to moderate traffic regimes. This is because *(i)* the base station serves a file only after the complete file is received at the base station, which results in large waiting times for large files, and *(ii)* the scheduling algorithm is independent of queue-sizes even in a light traffic regime, which again may result in large file-transfer delays for large files. Figure 3.2 shows a simulation result where we compare the MaxWeight algorithm and the CA-WS algorithm with uniform tie-breaking rule (CA-WSU) (the simulation setting will be described in Section 3.4). We can see while the CA-WSU has a smaller file-transfer delay than the MaxWeight in heavy traffic regime, in light and medium traffic regimes, the performance of the CA-WSU algorithm is much worse than the MaxWeight algorithm.

In fact, it has been observed in [16] that from the performance perspective, we may need to serve the files with large sizes using the MaxWeight algorithm. The authors in [16] suggest that flows be classified as long-lived flows and short-lived flows, and use different scheduling algorithms for different types of flows. However, they do not provide any criterion for the classification. Further, in practice, the base station may not even know the size of a file before the file fully arrives at the base station.

- The algorithm assumes that $\mathbf{R}_f^{\max}$ is known a priori, which is unrealistic in practice.

To overcome these two weaknesses, we introduce a hybrid CA-WS algorithm in the next section.

3.3 A Throughput-Optimal Hybrid CA-WS Algorithm

The key idea behind our hybrid algorithm is as follows: any flow whose last packet has not arrived at the base station (recall that these are called transient flows in the terminology of Section II) is treated as a persistent flow as in the traditional MaxWeight algorithm. The MaxWeight algorithm is then used to decide schedules among these flows. Flows that have fully arrived at the base station (called resident flows in Section II) are scheduled using the CA-WS algorithm. However, we have to further decide whether to schedule transient flows or resident flows over each channel. This is one of the key elements of the hybrid algorithm to be described later.

To tackle the issue of $\mathbf{R}_f^{\max}$, we adopt the learning idea introduced in [16]. We define a $\tilde{R}_{if}^{\max}(t)$ to be the best state of channel i seen by flow f from b_f to $\min\{t, b_f + D\}$, and use $\tilde{R}_{if}^{\max}(t)$ to approximate $R_{if}^{\max}$. The parameter D is called the learning period.

Before we present the hybrid CS-WS algorithm, we first define the sequence of events that take place within a slot. We assume the new flows (mobile users) arrive at the beginning of the time slot t (denoted by t_b) and the channel state of time t is also measured at t_b. Then we assume that any computation or recomputation of $\mathbf{h}_f(t)$ occurs at time t_m. Finally, the packets are served at the end of each time slot (denoted by t_e). The sequence of these events is demonstrated in Figure 3.3.

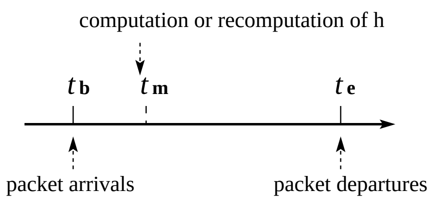

Figure 3.3 The sequence of flow/packet arrivals, computation or recomputation of $\mathbf{h}(t)$ and packet departures within a time slot

Hybrid Channel Assignment and Workload-based Scheduling (Hybrid CA-WS): At time slot t, the flows are served as follows:

(i) When a new flow (say flow f) joins the network, it records $Q_i(b_f)$ for all channels i.

(ii) **Channel learning:** The base station measures $R_{if}(t)$ for all i and f. Consider a flow f. If $t \leq b_f + D$ and $R_{if}(t) > \tilde{R}_{if}^{\max}(t-1)$ for some i, then flow f updates the $\tilde{\mathbf{R}}^{\max}$ based on the new channel state, i.e.,

$$\tilde{R}_{if}^{\max}(t) = \max\left\{\tilde{R}_{if}^{\max}(t-1), R_{if}(t)\right\}.$$

(iii) **Channel-assignment:** Consider a resident flow f. If $t \leq b_f + D$ and $\tilde{R}_{if}^{\max}(t) \neq \tilde{R}_{if}^{\max}(t-1)$ for some i, the base station recomputes $\mathbf{h}_f(t)$ by solving the following optimization problem:

$$\begin{aligned} \text{OPT}_f(t) = \quad & \min \textstyle\sum_{i\in\mathcal{M}} Q_i(b_f)h_{if}(t) \\ \text{subject to:} \quad & F_f(t) \leq \textstyle\sum_{i\in\mathcal{M}} h_{if}(t)\tilde{R}_{if}^{\max}(t), \end{aligned}$$

where $h_{if}(t)$ are non-negative integers. Note that the channel assignment for flow f is recomputed every time we have a better estimate of $R_{if}^{\max}$ for any channel i up to time $b_f + D$. This is necessary because the channel assignment algorithm is derived assuming $\mathbf{R}_f^{\max}$ is known.

(iv) Recall $\mathcal{L}(t)$ denotes the set of transient flows at time t and $\mathcal{S}(t)$ denotes the set of resident flows at time t. The base station first checks:

$$\sum_{f\in\mathcal{L}(t)} F_f(t) \leq \sum_{f\in\mathcal{S}(t)} F_f(t). \tag{3.5}$$

- **Workload-based scheduling:** If inequality (3.5) holds, the base station selects a resident file f for channel i such that

$$\tilde{R}_{if}^{\max}(t) \leq R_{if}(t) \text{ and } h_{if}(t) \neq 0, \tag{3.6}$$

and transmits $R_{if}(t)$ packets to mobile user f. Then the base station reduces $h_{if}(t)$ by one. If no resident flow satisfies (3.6), the base station randomly selects a flow, say flow f, and transmits $R_{if}(t)$ packets to mobile f. Ties are broken uniformly or according to the arrival time b_f (giving priority to flows with small b_f may improve delay performance in practice although it has no effect on stability). When the file of flow f is completely transferred to the mobile user, the base station sets $h_{if}(t) = 0$ for all i.

- **MaxWeight scheduling:** If inequality (3.5) does not hold, then the base station selects a transient file f^* for channel i such that

$$f^* \in \arg\max_{f \in \mathcal{L}(t)} F_f(t) R_{if}(t), \tag{3.7}$$

and transmits $\min\{F_f(t), R_{if^*}(t)\}$ packets to mobile user f^* over channel i.

$\square$

Remark 1: While each resident flow is associated with a channel assignment vector $\mathbf{h}_f(t)$, the packets of a flow are stored in the same queue and served in a First-In, First-Out (FIFO) fashion.

Remark 2: The advantages of using the MaxWeight algorithm for large-size flows are two-fold: *(i)* the file with a large size could experience smaller delay because it can be served at any $R_{if}(t)$ not just when the channel reaches the best state, and *(ii)* when only a few large-size flows are in the network, the MaxWeight algorithm can lead to a fair resource allocation. These advantages will be observed in the simulations.

In the next subsection, we will prove that the hybrid CA-WS algorithm is also throughput optimal. We would like to emphasize that because of the channel-assignment algorithm, which is not required for single-channel networks, the analysis is completely different from those in [1, 2, 16].

3.3.1 Throughput Optimality of the Hybrid CA-WS Algorithm

Without loss of generality we assume that $T_{in} = F^{\max}$, i.e., we assume that the injecting rate of any flow is at least one. All of our results apply more generally, but this assumption simplifies a lot of the notation. We first show that the number of transient flows is always bounded.

Lemma 6. *Assume that $s_f - b_f \leq T_{in}$ for all f, then no more than $\lambda^{\max} F^{\max}$ transient flows are in the network during any time slot.*

Proof. Recall that we assume that file sizes are upper bounded by $F^{\max}$, and the injecting rate is at least one. Therefore, the injection period, the time taken for a transient flow to become a resident flow, is upper bounded by $F^{\max}$. Furthermore, the number of new files joining the network at each time slot is bounded by $\lambda^{\max}$, so the number of transient files in the network is upper bounded by $\lambda^{\max} F^{\max}$. □

Since the number of transient flows is upper bounded at any time slot, to prove the stability of the network, we only need to consider the number of resident flows.

Before we proceed, we present a lemma first which is useful in the proof of the stability of hybrid CA-WS algorithm.

Lemma 7. *Consider the hybrid CA-WS with oldest-first or uniform tie-breaking rule and define $\mathcal{R}_i(t)$ to be the event that a resident flow f with $s_f \geq nT$ is served over channel i at time t. Given any $\delta > 0$, there exists Q_δ such that if $Q_i(nT) > Q_\delta$, then for any $nT \leq t \leq (n+1)T - 1$,*

$$\Pr(\mathcal{R}_i(t)) < \delta.$$

Proof. For any $t \in [nT, (n+1)T - 1]$, we denote by $\mathcal{O}_i(t)$ the set of resident flows that arrived before $nT - D$ ($D \geq F^{\max}$) and have $h_{if}(t) > 0$. It can be easily verified that

$$|\mathcal{O}_i(t)| \geq \frac{Q_i(nT)}{F^{\max}} - \lambda^{\max} D - T.$$

Consider the hybrid CA-WS with the oldest-first tie-breaking rule. Only if none of flows in $\mathcal{O}_i(t)$ have $R_{if}(t) = R_{if}^{\max}$, the base station will serve a flow which becomes a resident flow in $[nT, (n+1)T - 1]$ over channel i. Therefore, we have

$$\Pr(\mathcal{R}_i(t)) \leq (1 - p^{\max})^{\frac{Q_i(nT)}{F^{\max}} - \lambda^{\max} D - T},$$

and the lemma holds for the oldest-first tie-breaking rule.

Consider the hybrid CA-WS with the uniform tie-breaking rule. For any t such that $nT \leq t \leq (n+1)T-1$, the number of flows becoming resident flows after nT is no more than $\lambda^{\max}(T+F^{\max})$. Furthermore, according to the Chernoff's bound, we have

$$\begin{aligned} & \Pr\left(\left|\{f : f \in \mathcal{O}_i(t) \text{ and } R_{if}(t) = R_{if}^{\max}\}\right| \geq (1-\delta)\Theta\right) \\ \geq \quad & 1 - \exp\left(-\frac{\delta^2\Theta}{3}\right), \end{aligned}$$

where $\Theta = p^{\max}(\frac{Q_i(nT)}{F^{\max}} - \lambda^{\max}D - T)$.

Therefore, it can be easily shown that

$$\Pr\left(\mathcal{R}_i(t)\right) \leq \frac{\lambda^{\max}(T+F^{\max})}{\lambda^{\max}(T+F^{\max}) + (1-\delta)\Theta} + \exp\left(-\frac{\delta^2\Theta}{3}\right),$$

and the lemma holds for the uniform tie-breaking rule. □

To study the performance of the hybrid CA-WS, we first define a sampled version of the network, sampled once every T time slots, as follows:

$$\begin{aligned} \mathbf{M}(n) = \{ & Y_f(nT), F_f(nT), \tilde{\mathbf{R}}_f^{\max}(nT), \min\{D, nT - b_f\}, \\ & \mathbf{Q}(b_f), \mathbf{h}_f(nT)\}_{f \in \mathcal{L}(t) \cup \mathcal{S}(t)}, \end{aligned}$$

where $Y_f(nT)$ is the number of packets of flow f that have not been transmitted to the base station. It is easy to see that $\mathbf{M}(n)$ is a Markov chain. We also assume that the arrival process is such that the Markov chain is irreducible and aperiodic. The sampling interval T in the definition of $\mathbf{M}(n)$ above will be chosen later. The reason we need this T is that our proof uses the standard drift argument in Foster's criterion (see [12]), but the drift of $\mathbf{M}(n)$ may not be negative over successive time instants. The drift will be negative only after most flows in the network get reasonably accurate estimates of their channel assignment vectors, which may take several recomputations due to updates in the estimate of $\mathbf{R}^{\max}$. The parameter T tries to capture the time interval that it takes for most flows to get sufficiently accurate estimates of their channel assignment vectors.

Theorem 8. *Assume that $s_f - b_f \leq T_{in}$ for all f. Given arrival rates $\{\lambda_{ik}\}$ such that $\{\frac{1}{1-\epsilon}\lambda_{ik}\}$ are supportable, there exists a D_ϵ such that the Markov chain $\mathbf{M}(n)$ is positive-recurrent under the hybrid CA-WS algorithm with $D \geq D_\epsilon$, which implies that* $\lim_{t\to\infty} \mathbf{E}\left[\sum_i Q_i(t)\right] < \infty$.

Proof. We consider the Lyapunov function

$$V(n) = \sum_{i\in\mathcal{M}} Q_i^2(nT),$$

and introduce the following notations:

- $\mathcal{C}(t)$: We define $\mathcal{C}(t)$ to be the set of flows who become *resident flows* at the beginning of time slot t.
- $A_i(t)$: We define $A_i(t) = \sum_{f\in\mathcal{C}(t)} h_{if}(t_m)$, which is the increase in workload for channel i due to new resident flows, i.e., the flows in $\mathcal{C}(t)$.
- $\mu_i(t)$: We define $\mu_i(t) = \sum_{f\in\mathcal{S}(t)} \left(h_{if}(t_m) - h_{if}(t_e)\right)$, which is the decrease in workload for channel i when a resident flow is served over channel i.
- $A_i^r(t)$: We define $A_i^r(t) = \sum\limits_{f\in\mathcal{S}(t)\backslash\mathcal{C}(t)} \left(h_{if}(t_m) - h_{if}(t_b)\right)^+$, which is the increase in workload for channel i due to the adjustment of the channel assignment vectors of existing resident flows (in other words, due to the recomputation of $\mathbf{h}_f$).
- $\mu_i^r(t)$: We define $\mu_i^r(t) = \sum\limits_{f\in\mathcal{S}(t)\backslash\mathcal{C}(t)} \left(h_{if}(t_b) - h_{if}(t_m)\right)^+$, which is the decrease in workload for channel i due to the adjustment of channel assignments of existing resident flows.

Without causing confusion, we let $h_{if}(t) = h_{if}(t_m)$, i.e., $h_{if}(t)$ is the value after recomputation at time t. Now based on the notations above, the dynamics of $Q_i(t)$ can be written as

$$Q_i(t+1) = Q_i(t) + A_i(t) - \mu_i(t) + A_i^r(t) - \mu_i^r(t).$$

Note that the number of flows joining the network at each time slot is bounded by $\lambda^{\max}$, and the file size is also bounded by $F^{\max}$. Further each flow recomputes $\mathbf{h}_f$ for at most D time slots. Therefore $A_i(t)$, $\mu_i(t)$, $A_i^r(t)$, and $\mu_i^r(t)$ are all bounded:

$$\begin{aligned} A_i(t) &\leq \lambda^{\max}F^{\max} \\ \mu_i(t) &\leq F^{\max} \\ A_i^r(t) &\leq \lambda^{\max}F^{\max}D \\ \mu_i^r(t) &\leq \lambda^{\max}F^{\max}D. \end{aligned}$$

Note that in general $\mu_i(t) \leq 1$ since only one channel use is allowed in one time slot. The case $\mu_i(t) > 1$ occurs when the flow is completely transmitted to the mobile user and we set $h_{if}(t) = 0$. Note that when there is no flow having $R_{if}(t) = \tilde{R}^{\max}_{if}(t)$, the base station serves a flow f at rate $R_{if}(t)$ but does not reduce $h_{if}(t)$. So it is possible that even after almost all packets of a flow have been transmitted, we still have $h_{if}(t) > 1$.

Now based on the definitions and notations above, we have

$$|Q_i(nT) - Q_i(s)| \leq T\left(F^{\max} + \lambda^{\max}F^{\max}(2D+1)\right)$$

for all $s \in [nT, (n+1)T-1]$, and

$$\begin{aligned}
&\mathbf{E}[V(n+1) - V(n)|\mathbf{M}(n)] \leq \Phi_1 \\
&+ 2\sum_i Q_i(nT)\mathbf{E}\left[\sum_{t=nT}^{(n+1)T-1} A_i(t) + A_i^r(t) - \mu_i^r(t)\middle|\mathbf{M}(n)\right] \qquad (3.8)\\
&- 2\sum_i Q_i(nT)\mathbf{E}\left[\sum_{t=nT}^{(n+1)T-1} \mu_i(t)\middle|\mathbf{M}(n)\right], \qquad (3.9)
\end{aligned}$$

where $\Phi_1 = M\left(T\left(F^{\max} + \lambda^{\max}F^{\max}(2D+1)\right)\right)^2$.

In the following analysis, we will show that there exists a finite set $\mathcal{W}$ such that when $\mathbf{M}(n) \notin \mathcal{W}$, we have

$$\mathbf{E}[V(n+1) - V(n)|\mathbf{M}(n)] \leq -\frac{\epsilon}{2M}\sum_i Q_i(nT). \qquad (3.10)$$

The theorem then follows from the Foster's criterion [12]. To prove (3.10), we will first analyze (3.8) and (3.9) separately, and then show that

$$\Phi_1 + (3.8) + (3.9) \leq -\frac{\epsilon}{2M}\sum_i Q_i(nT)$$

when $\mathbf{M}(n) \notin \mathcal{W}$.

Analysis of (3.8)

Denote by $\mathcal{G}(n)$ the set of resident flows that are in the network at least in one of the time slots belonging to $[nT, (n+1)T-1]$. We further divide $\mathcal{G}(n)$ into five subsets (see Figure 3.4):

- $\mathcal{G}_A(n)$: The set of resident flows that *(i)* become *resident* during $[nT, (n+1)T-D-1]$, *(ii)* are not served during $[nT, (n+1)T-1]$, and *(iii)* have learned $\mathbf{R}^{\max}$ by time $(n+1)T-1$.
- $\mathcal{G}_B(n)$: The set of resident flows that *(i)* become *resident* during $[nT, (n+1)T-D-1]$, *(ii)* are not served during $[nT, (n+1)T-1]$, and *(iii)* have not learned $\mathbf{R}^{\max}$ by time $(n+1)T-1$.
- $\mathcal{G}_C(n)$: The set of resident flows that become *resident* during $[nT, (n+1)T-D-1]$ and are served at least once during $[nT, (n+1)T-1]$.
- $\mathcal{G}_D(n)$: The set of resident flows that become *resident* during $[(n+1)T-D, (n+1)T-1]$.
- $\mathcal{G}_E(n)$: The set of resident flows that are in the system at nT.

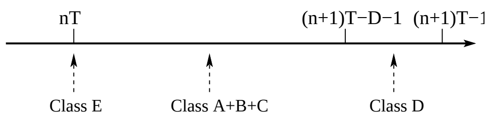

Figure 3.4 Five subsets of $\mathcal{G}(n)$

It is obvious to see that

$$\mathcal{G}(n) = \mathcal{G}_A(n) \cup \mathcal{G}_B(n) \cup \mathcal{G}_C(n) \cup \mathcal{G}_D(n) \cup \mathcal{G}_E(n).$$

Recall that s_f denotes the time flow f becomes a resident flow, so (3.8) can be rewritten as

$$\sum_{t=nT}^{(n+1)T-1} (A_i(t) + A_i^r(t) - \mu_i^r(t)) \tag{3.11}$$

$$= \sum_{f\in\mathcal{G}(n)} \sum_{t=\max\{s_f,nT\}}^{(n+1)T-1} (A_{if}(t) + A_{if}^r(t) - \mu_{if}^r(t)), \tag{3.12}$$

where $A_{if}(t)$, $A_{if}^r(t)$ and $\mu_{if}^r(t)$ are workload adjustments related to flow f. Next we analyze (3.12) case by case. To simplify our notations, we assume $D \geq \max\{F^{\max}, \frac{\lambda^{\max} F^{\max}}{\min_{k,F} \lambda_{kF}}\}$.

In the following analysis, we will show that *the subset of flows that determines the value of (3.8) is* $\mathcal{G}_A(n)$. Since the flows in $\mathcal{G}_A(n)$ learn the correct $\mathbf{R}_f^{\max}$ by $(n+1)T-1$ and are not served during $[nT, (n+1)T-D-1]$, we can compare the channel assignment vector under the hybrid CA-WS with that defined in the necessary conditions, which will lead to (3.10) by combining the analysis of (3.9).

Case 1: We first consider a flow in $\mathcal{G}_A(n)$, and have

$$\begin{aligned} & \sum_{t=s_f}^{(n+1)T-1} A_{if}(t) + A^r_{if}(t) - \mu^r_{if}(t) & (3.13) \\ = \quad & h_{if}(s_f) + \sum_{t=s_f+1}^{(n+1)T-1} (A^r_{if}(t) - \mu^r_{if}(t)). & (3.14) \end{aligned}$$

Since f is not served before $(n+1)T$, according to the definitions of A_{if}, A^r_{if}, and μ^r_{if}, we have

$$h_{if}(t+1) - h_{if}(t) = A^r_{if}(t+1) - \mu^r_{if}(t+1)$$

for any $s_f \leq t \leq (n+1)T - 2$, which implies that

$$h_{if}(s_f) + \sum_{t=s_f+1}^{(n+1)T-1} (A^r_{if}(t) - \mu^r_{if}(t)) = h_{if}((n+1)T-1),$$

and

$$\begin{aligned} & \mathbf{E}\left[\sum_{f\in\mathcal{G}_A(n)} \sum_{t=b_f}^{(n+1)T-1} (A_{if}(t) + A^r_{if}(t) - \mu^r_{if}(t)) \middle| \mathbf{M}(n)\right] \\ & = \mathbf{E}\left[\sum_{f\in\mathcal{G}_A(n)} h_{if}((n+1)T-1) \middle| \mathbf{M}(n)\right]. \end{aligned}$$

Case 2: Following the analysis of Case 1, for any $f \in \mathcal{G}_B(n)$, we obtain

$$\sum_{t=nT}^{(n+1)T-1} (A_{if}(t) + A^r_{if}(t) - \mu^r_{if}(t)) = h_{if}((n+1)T-1).$$

Since $h_{if}((n+1)T-1) \leq F^{\max}$ for any f and any channel i,

$$\begin{aligned} & \mathbf{E}\left[\sum_{f\in\mathcal{G}_B(n)} \sum_{t=s_f}^{(n+1)T-1} (A_{if}(t) + A^r_{if}(t) - \mu^r_{if}(t)) \middle| \mathbf{M}(n)\right] \\ = \quad & \mathbf{E}\left[\sum_{f\in\mathcal{G}_B(n)} h_{if}((n+1)T-1) \,|\mathbf{M}(n)\right] \\ \leq \quad & F^{\max}\mathbf{E}\left[\sum_{f\in\mathcal{G}_B(n)} 1\right]. \end{aligned}$$

Now we study the size of $\mathcal{G}_B(n)$. According to Lemma 6, the network has at most $\lambda^{\max}F^{\max}$ transient flows at time slot $nT-1$, which may become resident flows during $[nT, (n+1)T-1]$.

Also at each time slot, at most $\lambda^{\max}$ flows join the network. For a resident flow with $s_f \leq (n+1)T - D$, the probability that the flow has not learned the $\mathbf{R}^{\max}$ by time $(n+1)T-1$ is less than $M\left(1-p^{\max}\right)^D$. Therefore, we have

$$\begin{aligned} &\mathbf{E}\left[\sum_{f\in\mathcal{G}_B(n)} 1\right] \\ \leq\ & \left(\lambda^{\max}F^{\max} + (T-D)\lambda^{\max}\right) M\left(1-p^{\max}\right)^D, \end{aligned}$$

and

$$\begin{aligned} &\mathbf{E}\left[\sum_{f\in\mathcal{G}_B(n)} \sum_{t=s_f}^{(n+1)T-1} \left(A_{if}(t) + A_{if}^r(t) - \mu_{if}^r(t)\right) \middle| \mathbf{M}(n)\right] \\ &\leq F^{\max}\left(\lambda^{\max}F^{\max} + (T-D)\lambda^{\max}\right) M\left(1-p^{\max}\right)^D \\ &\leq F^{\max}\lambda^{\max}TM\left(1-p^{\max}\right)^D, \end{aligned}$$

where the last inequality holds under the assumption that $D \geq F^{\max}$.

Case 3: We now study the flows in $\mathcal{G}_C(n)$. Since flows f are served before $(n+1)T$, according to the definition of the notations, we have

$$h_{if}(t+1) - h_{if}(t) = A_{if}^r(t+1) - \mu_{if}^r(t+1) - \mu_{if}(t)$$

for any $s_f \leq t \leq (n+1)T-2$, which implies that

$$\begin{aligned} & \sum_{t=s_f}^{(n+1)T-1} A_{if}(t) + A_{if}^r(t) - \mu_{if}^r(t) \\ =\ & h_{if}(s_f) + \sum_{t=s_f+1}^{(n+1)T-1} \left(A_{if}^r(t) - \mu_{if}^r(t)\right) \\ =\ & h_{if}((n+1)T-1) + \sum_{t=s_f+1}^{(n+1)T-1} \mu_{if}(t) \\ \leq\ & F^{\max}, \end{aligned}$$

and

$$\begin{aligned} &\mathbf{E}\left[\sum_{f\in\mathcal{G}_C(n)} \sum_{t=b_f}^{(n+1)T-1} \left(A_{if}(t) + A_{if}^r(t) - \mu_{if}^r(t)\right) \middle| \mathbf{M}(n)\right] \\ &= F^{\max}\mathbf{E}\left[\textstyle\sum_{f\in\mathcal{G}_C(n)} 1 \middle| \mathbf{M}(n)\right]. \end{aligned}$$

Note that the number of flows that become *resident* during $[nT, (n+1)T - D - 1]$ is no more than

$$\lambda^{\max} F^{\max} + \lambda^{\max}(T - D) \leq \lambda^{\max} T$$

since we have at most $\lambda^{\max} F^{\max}$ transient flows at time $nT - 1$ and at most $\lambda^{\max}$ new flows join the network at each time slot t.

Now according to Lemma 7, that given any δ, there exists Q_δ such that if $Q_i(nT) \geq Q_\delta$, then the probability that a flow with $s_f \geq nT$ is served any given time slot in $[nT, (n+1)T - 1]$ is less than δ. Therefore, if $Q_i(nT) \geq Q_\delta$, we have that

$$\mathbf{E}\left[\sum_{f \in \mathcal{G}_C(n)} 1 \middle| \mathbf{M}(n)\right] \leq \lambda^{\max} T^2 \delta;$$

and otherwise

$$\begin{aligned} & Q_i(nT)\mathbf{E}\left[\sum_{f \in \mathcal{G}_C(n)} \sum_{t=s_f}^{(n+1)T-1} A_{if}(t) + A^r_{if}(t) - \mu^r_{if}(t) \middle| \mathbf{M}(n)\right] \\ & \leq Q_\delta \lambda^{\max} F^{\max} T. \end{aligned}$$

We then conclude that

$$\begin{aligned} & \sum_i Q_i(nT)\mathbf{E}\left[\sum_{f \in \mathcal{G}_C(n)} \sum_{t=s_f}^{(n+1)T-1} A_{if}(t) + A^r_{if}(t) - \mu^r_{if}(t) \middle| \mathbf{M}(n)\right] \\ & \leq \sum_i \left(Q_i(nT)\lambda^{\max} F^{\max} T^2 \delta + Q_\delta \lambda^{\max} F^{\max} T\right). \end{aligned}$$

Case 4: We now study the flows in $\mathcal{G}_D(n)$. Following the analysis of Case 3, the size of set $\mathcal{G}_D(n)$ is upper bounded by

$$\lambda^{\max} F^{\max} + \lambda^{\max} D.$$

Therefore,

$$\begin{aligned} & \mathbf{E}\left[\sum_{f \in \mathcal{G}_D(n)} \sum_{t=s_f}^{(n+1)T-1} \left(A_{if}(t) + A^r_{if}(t) - \mu^r_{if}(t)\right) \middle| \mathbf{M}(n)\right] \\ = \ & \mathbf{E}\left[\sum_{f \in \mathcal{G}_D(n)} h_{if}((n+1)T - 1) + \sum_{t=s_f+1}^{(n+1)T-1} \mu_{if}(t) \middle| \mathbf{M}(n)\right] \\ \leq \ & F^{\max}\left(\lambda^{\max} F^{\max} + \lambda^{\max} D\right). \end{aligned}$$

Case 5: We now analyze the last case: the set $\mathcal{G}_E(n)$. For a flow $f \in \mathcal{G}_E(n)$, we have the following facts:

- $A_{if}(t) = 0$,
- $|A^r_{if}(t) - \mu^r_{if}(t)| \leq F^{\max}$ for any $nT \leq t < nT + D$, and
- $A^r_{if}(t) = \mu^r_{if}(t) = 0$ for $t \geq nT + D$.

The last equality holds because a resident flow adjusts its $\mathbf{h}_f(t)$ for at most D time slots after joining the network.

Now note that at most $\lambda^{\max}D$ flows join the network during $[nT - D, nT - 1]$, which are the only flows in set $\mathcal{G}_E(n)$ that recompute $\mathbf{h}_f$ during $[nT, (n+1)T - 1]$. Therefore, we obtain

$$\begin{aligned} &\mathbf{E}\left[\sum_{f \in \mathcal{G}_E(n)} \sum_{t=nT}^{(n+1)T-1} \left(A_{if}(t) + A^r_{if}(t) - \mu^r_{if}(t)\right) \middle| \mathbf{M}(n)\right] \\ \leq\ & \lambda^{\max} D^2 F^{\max}. \end{aligned}$$

□

Summarizing the five cases above, we obtain

$$\begin{aligned} &(3.8) \\ \leq & 2\sum_i Q_i(nT)\mathbf{E}\left[\sum_{f \in \mathcal{G}_A(n)} h_{if}((n+1)T - 1) \middle| \mathbf{M}(n)\right] \\ +& 2\sum_i Q_i(nT)\left(F^{\max}\lambda^{\max}TM\left(1 - p^{\max}\right)^D + \right. \\ & \qquad \left. \lambda^{\max}F^{\max}T^2\delta + 2\lambda^{\max}F^{\max}D + \lambda^{\max}F^{\max}D^2\right) \\ +& MQ_\delta\lambda^{\max}F^{\max}T. \end{aligned} \tag{3.15}$$

Analysis of (3.9)

Next we consider (3.9) under the assumption that

$$\sum_i Q_i(nT) > \left(F^{\max}\right)^2 \lambda^{\max} + TMF^{\max}. \tag{3.16}$$

It can be easily verified that under assumption (3.16), the base station always serves resident flows during $[nT, (n+1)T-1]$ because we have at most $\lambda^{\max}F^{\max}$ transient flows in the network at any given time.

Since $h_{if}(t) \leq F^{\max}$ for any i and f, there are at least $Q_i(t)/F^{\max}$ flows having $h_{if}(t) > 0$ at time t. Therefore, we obtain that

$$\Pr(\mu_i(t) = 1) \geq 1 - (1 - p^{\max})^{\frac{Q_i(t)}{F^{\max}}},$$

and

$$\mathbf{E}\left[\mu_i(t)|\mathbf{M}(n)\right] \geq 1 - (1 - p^{\max})^{\frac{Q_i(nT)-\sqrt{\Phi_1}}{F^{\max}}}. \tag{3.17}$$

Analysis of (3.8)+(3.9)

Recall that the theorem assumes that there exists $Z^*_{\mathbf{h}|kF}$ such that

$$\begin{aligned} \sum_{k,F,\mathbf{h}} \lambda_{kF} Z^*_{\mathbf{h}|kF} h_i &\leq 1-\epsilon, i = 1, 2, ..., M & (3.18)\\ \sum_{\mathbf{h}} Z^*_{\mathbf{h}|kF} &= 1, \forall k, F & (3.19)\\ Z_{\mathbf{h}|kF} &= 0 \text{ if } F > \sum_{i\in\mathcal{M}} h_i R^{\max}_{ik},. & (3.20) \end{aligned}$$

Next we define

$$\mathcal{H}_{kF}(n) = \{f : f \in \mathcal{G}_A(n), k_f = k, \tilde{F}_f = F\},$$

i.e., $\mathcal{H}_{kF}(n)$ is the set of class-k flows that belong to set $\mathcal{G}_A(t)$ and with file length F. For any $f \in \mathcal{H}_{kF}(n)$, since $\mathbf{R}^{\max}_f$ has been correctly learned at time $(n+1)T-1$, we have

$$\sum_i Q_i(b_f) h_{if}((n+1)T-1) \leq \left(\sum_i Q_i(b_f) h_i\right)$$

for any $\mathbf{h}$ such that $\sum_i h_i R^{\max}_{if} \geq F$. Based on (3.19), we further obtain

$$\begin{aligned} \sum_i Q_i(b_f) h_{i,f}((n+1)T-1) \leq & \sum_{\mathbf{h}} Z^*_{\mathbf{h}|kF} \left(\sum_i Q_i(b_f) h_i\right) \\ = & \sum_i Q_i(b_f) \sum_{\mathbf{h}} Z^*_{\mathbf{h}|kF} h_i. & (3.21) \end{aligned}$$

Now based on inequality (3.21) and assume $T > F^{\max}$, we obtain

$$
\begin{aligned}
&\sum_{k,F}\sum_i Q_i(nT)\mathbf{E}\left[\sum_{f\in\mathcal{H}_{kF}(n)} h_{i,f}((n+1)T-1)\middle|\mathbf{M}(n)\right]\\
\le_{(a)}&\Phi_1+\sum_{k,F}\mathbf{E}\left[\sum_{f\in\mathcal{H}_{kF}(n)}\sum_i Q_i(b_f)h_{i,f}((n+1)T-1)\middle|\mathbf{M}(n)\right]\\
\le&\Phi_1+\sum_{k,F}\mathbf{E}\left[\sum_{f\in\mathcal{H}_{kF}(n)}\sum_i Q_i(b_f)\sum_{\mathbf{h}} Z^*_{\mathbf{h}|kF}h_i\middle|\mathbf{M}(n)\right]\\
\le_{(a)}&2\Phi_1+\sum_{k,F}\sum_i\left(Q_i(nT)\left(\sum_{\mathbf{h}}Z^*_{\mathbf{h}|kF}h_i\right)\times\right.\\
&\left.\mathbf{E}\left[\sum_{f\in\mathcal{H}_{kF}(n)}1\middle|\mathbf{M}(n)\right]\right)\\
\le_{(b)}&2\Phi_1+\sum_{k,F}\sum_i\left(Q_i(nT)\left(\sum_{\mathbf{h}}Z^*_{\mathbf{h}|kF}h_i\right)\times\right.\\
&\left.\mathbf{E}\left[|\mathcal{L}(nT)|+\sum_{f:(n+1)T-D-1\ge b_f\ge nT}1\middle|\mathbf{M}(n)\right]\right)\\
\le&2\Phi_1+\sum_{k,F}\sum_i\left(Q_i(nT)\left(\sum_{\mathbf{h}}Z^*_{\mathbf{h}|kF}h_i\right)\times\right.\\
&\left.\left(\lambda^{\max}F^{\max}+\lambda_{kF}(T-D)\right)\right)\\
\le&2\Phi_1+\sum_{k,F}\sum_i Q_i(nT)T\lambda_{kF}\sum_{\mathbf{h}}Z^*_{\mathbf{h}|kF}h_i,
\end{aligned}
\tag{3.22}
$$

where inequality (a) holds because $(n+1)T-1\ge b_f\ge nT-F^{\max}$ for any $f\in\mathcal{H}_{kF}(n)$ and $|Q_i(nT)-Q_i(b_f)|\le T\left(F^{\max}+\lambda^{\max}F^{\max}(2D+1)\right)$, and inequality (b) holds because the flows in $\mathcal{H}_{kF}(n)$ must arrive during $[nT,(n+1)T-1]$ or are transient flows at time nT.

Now by combining inequalities (3.15) and (3.22), we get that

$$
\begin{aligned}
&(3.8)-2\textstyle\sum_i Q_i(nT)\sum_{t=nT}^{(n+1)T-1}\sum_{\mathbf{h},k,F}\lambda_{k,F}Z^*_{\mathbf{h}|k,F}h_i\\
\le\;&4\Phi_1+MQ_\delta\lambda^{\max}F^{\max}T\\
&+2\textstyle\sum_i Q_i(nT)\Big(F^{\max}\lambda^{\max}TM\left(1-p^{\max}\right)^D+\\
&\lambda^{\max}F^{\max}T^2\delta+2\lambda^{\max}F^{\max}D+\lambda^{\max}F^{\max}D^2\Big)
\end{aligned}
\tag{3.23}
$$

Further, based on inequality (3.17), we have

$$\begin{aligned}
&2\sum_i Q_i(nT)\sum_{t=nT}^{(n+1)T-1}\sum_{\mathbf{h},k,F}\lambda_{k,F}Z^*_{\mathbf{h}|k,F}h_i + (3.9)\\
&= 2\sum_i Q_i(nT)\times\\
&\mathbf{E}\left[\sum_{t=nT}^{(n+1)T-1}\left(\sum_{\mathbf{h},k,F}\lambda_{k,F}Z^*_{\mathbf{h}|k,F}h_i - \mu_i(t)\right)\middle|\mathbf{Q}(nT)\right]\\
&\leq 2\sum_i Q_i(nT)\left(-\epsilon T + T(1-p^{\max})^{\frac{Q_i(nT)-\sqrt{\Phi_1}}{F^{\max}}}\right).
\end{aligned} \tag{3.24}$$

Combining inequalities (3.23) and (3.24), we have

$$\begin{aligned}
&\mathbf{E}[V(n+1)-V(n)|\mathbf{M}(n)]\\
\leq\ &\Phi_1 + (3.8) + (3.9)\\
\leq\ &5\Phi_1 + MQ_\delta\lambda^{\max}F^{\max}T\\
&+2\textstyle\sum_i Q_i(nT)\left(F^{\max}\lambda^{\max}TM\left(1-p^{\max}\right)^D +\right.\\
&\left.\lambda^{\max}F^{\max}T^2\delta + 2\lambda^{\max}F^{\max}D + \lambda^{\max}F^{\max}D^2\right)\\
&+2\textstyle\sum_i Q_i(nT)\left(-\epsilon T + T(1-p^{\max})^{\frac{Q_i(nT)-\sqrt{\Phi_1}}{F^{\max}}}\right).
\end{aligned}$$

Now we define a set $\mathcal{W}$ such that if $\mathbf{M}(n)\in\mathcal{W}$, then

$$\begin{aligned}
\sum_i Q_i(nT) \leq\ &\tfrac{2M}{\epsilon T}\left(5\Phi_1 + MQ_\delta\lambda^{\max}F^{\max}T\right.\\
&\left.+2MT\sqrt{\Phi_1} + 2MTF^{\max}\tfrac{\log(\epsilon/4)}{\log(1-p^{\max})}\right),
\end{aligned}$$

where $\delta = \frac{\epsilon}{16T\lambda^{\max}F^{\max}}$ and Q_δ is the constant defined in Lemma 7.

We now choose D and T such that

$$\begin{aligned}
D &\geq \frac{\log\frac{\epsilon}{16M\lambda^{\max}F^{\max}}}{\log(1-p^{\max})}\\
T &\geq \frac{32\lambda^{\max}F^{\max}D^2}{\epsilon}.
\end{aligned}$$

We can see that $\mathcal{W}$ is a set with a finite number of elements, and can verify that if $\mathbf{M}(n)\notin\mathcal{W}$, then

$$\mathbf{E}[V(n+1)-V(n)|\mathbf{M}(n)] < -\frac{\epsilon}{2M}\sum_i Q_i(nT).$$

Now according to the Foster's criterion, the Markov chain is positive recurrent, and further, $\lim_{t\to\infty} \mathbf{E}[\sum_i Q_i(t)] < \infty$ [13]. □

3.4 Simulations

In this section, we use simulations to evaluate the hybrid CA-WS algorithm and compare its performance with the MaxWeight scheduling scheme and the CA-WS scheduling scheme. Both the CA-WS and hybrid CA-WS algorithms in the simulations use learning to estimate the maximum transmission rate in each channel.

We consider a network with a single base station and five channels. We further assume there are three classes of flows (mobile users) in network. Class 1 users represent those close to the base station. The channel conditions of class 1 users therefore are better than those of other classes. Class 3 users represent those who are at the edge of the cell. The channel conditions of class 3 users are the worst. Class 2 users are assumed to be located in the middle of the cell. We assume that users in the same class experience the same channel fading, i.e., have the same channel distributions. We further assume that each channel has two possible states (high and low), and each of them happens with probability 0.5. The channel rate distributions of the five channels for the three classes are shown in Table 3.1.

The flow arrival rates of the three classes follow the same Poisson distribution with rate λ. In the simulation, we vary λ to compare the performances of different scheduling schemes under different traffic loads. The file size of a flow follows the Pareto distribution with minimum possible value $x_m = 50$, and decay factor $\alpha = 2$.[2] A transient flow keeps injecting packets into the base station until the complete file is transferred to the base station. The packet arrival rate of file f is controlled by the following congestion controller [9, 10]:

$$X_f(t) = \min\left\{ \left\lceil \frac{50}{F_f(t)} \right\rceil, 50 \right\}.$$

In the simulations, the learning period D is chosen to be 20. We name the CA-WS algorithm with the uniform tie-breaking rule as CA-WSU.

[2]In the simulation, in order to see the performance of our algorithm under a general setting, we do not set an upper bound for file size distribution and flow arrival rate distribution.

Table 3.1 The distributions of channel rates

Class	Channel	High rate	Low rate
Class 1	Channel 1	50	25
	Channel 2	48	24
	Channel 3	46	23
	Channel 4	44	22
	Channel 5	42	21
Class 2	Channel 1	40	20
	Channel 2	38	19
	Channel 3	36	18
	Channel 4	34	17
	Channel 5	32	16
Class 3	Channel 1	30	15
	Channel 2	28	14
	Channel 3	26	13
	Channel 4	24	12
	Channel 5	22	11

Simulation I: Number of Flows and File-Transfer Delay

We first consider the case where the base station does not limit the number of flows in the network. From the base station's perspective, it wants to minimize the total number of flows to reduce the buffer occupancy and computation complexity. From a user's perspective, the user wants to have small file-transfer delay. Therefore, we use simulations to compare the average numbers of flows in the network and the average file-transfer delays under the three scheduling algorithms.

The results are shown in Figure 3.5 and 3.6. We can see that when traffic load is light (i.e., λ is small), the hybrid CA-WSU algorithm and the MaxWeight have similar performance, while the CA-WSU algorithm has much higher delays. The reason is that the CA-WSU scheme starts to serve a flow only after the complete file is received at the base station, which significantly increases the file-transfer delay. When λ is large, the file-transfer delay of the MaxWeight algorithm becomes very large. This is because the MaxWeight is not throughput optimal.

Interestingly, the hybrid CA-WSU algorithm also performs much better than the CA-WSU algorithm even when λ is large. Specifically, the average number of flows and file-transfer delay of the hybrid CA-WSU algorithm with $\lambda = 0.48$ are smaller than those under the MaxWeight or the CA-WSU algorithms with $\lambda = 0.4$.

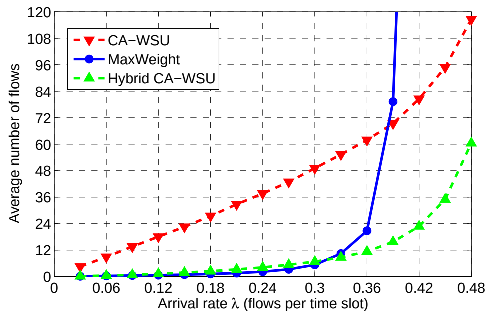

Figure 3.5 The average numbers of flows under the CA-WSU, hybrid CA-WSU and MaxWeight algorithms

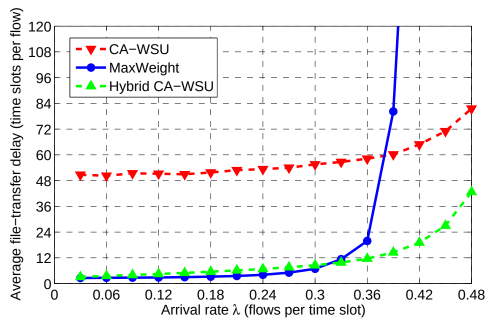

Figure 3.6 The average file-transfer delays under the CA-WSU, hybrid CA-WSU and MaxWeight algorithms

Simulation II: Blocking probability of three algorithms

In practical systems, the base station can only support a finite number of mobiles at any given time slot. In this simulation, we assume the base station can accommodate at most 50 flows simultaneously. New flows are blocked if the number of flows in the network already reaches 50. We use the blocking probability as the performance metric to compare the three scheduling algorithms.

The result is shown in Figure 3.7. We can see under a small λ, all three algorithms have small blocking probabilities. However, when $\lambda = 0.5$, the blocking probability of the hybrid CA-WSU is only 6%, while the blocking probability of the MaxWeight algorithm is around 20% and the blocking probability of the CA-WSU algorithm is around 40%. Thus, our algorithm which was designed for throughput optimality assuming no limit on the number of simultaneous flows in the network also performs well in situations where the number of allowed flows is limited.

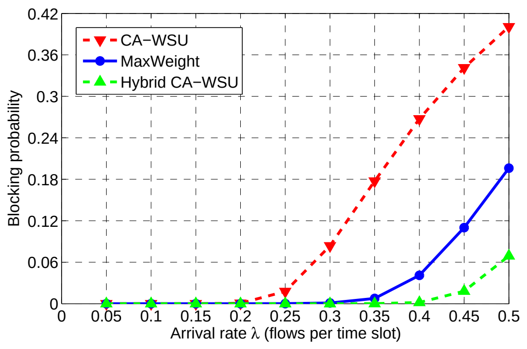

Figure 3.7 The blocking probabilities of the CA-WSU, the hybrid CA-WSU and MaxWeight algorithms

3.5 Conclusion

In this chapter, we have developed a hybrid channel assignment and workload-based scheduling algorithm that is throughput optimal for multichannel downlink wireless networks in the presence of flow-level dynamics. The algorithm has been proved to be throughput optimal and the performance, including delay and blocking probability, has been shown to be much superior to other alternatives.

CHAPTER 4. Joint Congestion Control and Scheduling in Wireless Peer-to-peer Networks

In wireless peer-to-peer networks, a pair of nodes communicate directly with each other. All transmissions are single-hop. This communication pattern is more efficient compared to that in cellular networks because the transmission of each packet no longer needs to go through two hops, i.e., the uplink and downlink. In wireless peer-to-peer networks, the admission control and medium access control are very important because concurrent transmissions can cause severe interference if not arranged wisely. We assume there is a central controller in wireless peer-to-peer networks, which schedules transmissions in each time slot, and our objective is designing a joint congestion control and scheduling algorithm which maximizes the network welfare while satisfying the delay constraints of traffic.

4.1 Network Model

In this section we describe the model we propose for a network that has message requests subject to deadline constraints. The network is located in a bounded region $\mathcal{R}$, where at the beginning of each frame, multiple communication requests occur in the network. A communication request is from one location of the region to another location. The request either gets fulfilled during that frame, or gets dropped due to deadline expiration. In this network, all flows are finite-sized messages with strict deadlines, so resource allocation algorithms designed for persistent flows cannot be used. To effectively schedule these real-time messages, we propose to partition $\mathcal{R}$ into subregions that share similar interference and channel conditions. This will allow us to pose the problem as a long-term optimization problem, where we can maximize the total network utility.

Traffic requests are assumed to originate in a region $\mathcal{R}$ that we divide in M disjoint sub-regions $\{r_i\}_{i\in\mathcal{M}}$, i.e., $r_i \cap r_j = \emptyset$ for all $i \neq j \in \mathcal{M} \overset{def}{=} \{1,\ldots,M\}$ and $\cup_{i\in\mathcal{M}} r_i = \mathcal{R}$. Thus, to specify a flow, we must specify the region where the source node is and the region where the destination is. These regions are also used to define the interference constraints, which we represent by the interference graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges. Formally, $\mathcal{V} \overset{def}{=} \{v = (r_i, r_j) : r_i, r_j \in \mathcal{R}$ for $i, j \in \mathcal{M}\}$ denotes any pair of regions such that the source node is in r_i and the destination is in r_j, and if $(v_1, v_2) \in \mathcal{E}$, where $v_1 = (r_{i1}, r_{j1})$, $v_2 = (r_{i2}, r_{j2})$, then a flow with source in r_{i1} and destination in r_{j1} cannot be scheduled to transmit simultaneously with a flow with source in r_{i2} and destination in r_{j2}.

We assume that time is divided in *slots*, and a set of T consecutive time slots is called a *frame*. Every message is assumed to be comprised of a single, fixed-size packet such that the packet can be transmitted in a time slot and has a deadline of T slots. Furthermore, it is assumed that all packets arrive at the beginning of the frame.

Let $a = (a_{ij})_{i,j\in\mathcal{M}}$ denote the number of real-time messages that arrive at region r_i destined for region r_j at the beginning of a given frame. We assume that a_{ij} is a random variable with mean λ_{ij} and variance σ_{ij}^2, that is independent between different frames, and is such that $Pr(a_{ij} = 0) > 0$ and $Pr(a_{ij} = 1) > 0$. The last assumption is to guarantee that the Markov chain we define later is both irreducible and aperiodic, but it can be substituted by other similar assumptions.

Depending on the wireless technology used, we can have some channel feedback before or after a transmission occurs, either in the form of channel estimation or receiver feedback, respectively. Furthermore, we can define different channel models depending on whether the receivers acknowledge reception of all the packets at the end of the frame, or if acknowledgments are received after each transmission.

In this chapter we assume that the potential number of packets that can be transmitted from region r_i destined for region r_j in a given time slot is denoted by $c = (c_{ij})_{i,j\in\mathcal{M}}$. We assume that the channel state c_{ij} is a Bernoulli random variable, is known at the beginning of the frame, and remains constant for the entire frame. Furthermore, we assume that the channel state is independent between different frames and independent of arrivals. This channel model corresponds to the case when we do channel estimation before transmissions occur. We use this model because it allow us to explain the main ideas behind our algorithm in the simplest way. The other cases have a development similar in nature and are thus omitted. The interested reader is referred to [24] for a related problem where these different channel models are studied in further detail, and it is shown how different channel models affect scheduling.

Denote by $q = (q_{ij})_{i,j \in \mathcal{M}}$ the minimum fraction of packets that need to be served originating in region r_i destined for region r_j, and by $U_{ij}(q_{ij})$ the utility function associated with such fraction. Furthermore, we assume that the function $U_{ij}()$ is concave.

We denote by $s = (s_{ijt})_{i,j \in \mathcal{M}, t \in \mathcal{T}}$ the schedule at any given frame, where s_{ijt} indicates the number of packets scheduled for service from region r_i to region r_j in time slot $t \in \mathcal{T} \overset{def}{=} \{1, \ldots, T\}$. Furthermore, we consider only schedules that fulfill all interference constraints, that is, if $s_{i_1 j_1 t} > 0$ and $s_{i_2 j_2 t} > 0$ for any given time slot t, then it must be the case that $(v_1, v_2) \notin \mathcal{E}$, where $v_1 = (r_{i1}, r_{j1})$, $v_2 = (r_{i2}, r_{j2})$. Since the number of available messages and the channel state determine the maximum number of packets that can be scheduled, we have the following constraints in the schedule:

$$\sum_{t \in \mathcal{T}} s_{ijt} \leq a_{ij} \text{ for all } i, j \in \mathcal{M} \tag{4.1}$$

$$s_{ijt} \leq c_{ij} \text{ for all } i, j \in \mathcal{M},\, t \in \mathcal{T} \tag{4.2}$$

We will denote by $\mathcal{S}(a, c)$ the set of feasible schedules for fixed arrivals and channel state, subject to (4.1), (4.2) and the interference constraints given by graph $\mathcal{G}$.

4.2 Optimization Formulation

We now present a static optimization problem which will be the base to design a dynamic algorithm using a dual decomposition approach.

Our goal is to find a scheduling policy $Pr(s|a, c)$, which is the probability of using schedule $s \in \mathcal{S}(a, c)$ when the arrivals are a and the channel state is c. Thus, the expected service for requests with source in region r_i and destination in r_j, $\mu_{ij}(a,c)$, has the following constraint

$$\mu_{ij}(a,c) \leq \sum_{s\in\mathcal{S}(a,c)} \sum_{t\in\mathcal{T}} s_{ijt} Pr(s|a,c),$$

and the overall expected service is given by

$$\mu_{ij} = \sum_{a,c} \mu_{ij}(a,c) Pr(a) Pr(c).$$

For notational simplicity, define the capacity region for fixed arrivals and channel state as

$$\mathcal{C}(a,c) \overset{def}{=} \left\{ \begin{array}{l} (\bar{\mu}_{ij})_{i,j\in\mathcal{M}} : \text{there exists } \bar{s} \in \mathcal{S}(a,c)_{\mathcal{CH}}, \\ \bar{\mu}_{ij} \leq \sum_{t\in\mathcal{T}} \bar{s}_{ijt} \text{ for all } i,j \in \mathcal{M} \end{array} \right\},$$

where $\mathcal{S}(a,c)_{\mathcal{CH}}$ is the convex hull of $\mathcal{S}(a,c)$. Similarly, if we define the overall capacity of the network as $\mathcal{C} \overset{def}{=}$

$$\left\{ \begin{array}{l} (\mu_{ij})_{i,j\in\mathcal{M}} : \text{there exists } \bar{\mu}(a,c) \in \mathcal{C}(a,c) \\ \text{for all } a,c \text{ and } \mu_{ij} = E[\bar{\mu}_{ij}(a,c)] \text{ for all } i,j \in \mathcal{M} \end{array} \right\}$$

then we have that $\mu \overset{def}{=} (\mu_{ij})_{i,j\in\mathcal{M}} \in \mathcal{C}$.

From the definition of q_{ij} we have the constraint

$$\lambda_{ij} q_{ij} \leq \mu_{ij} \text{ for all } i,j \in \mathcal{M}.$$

In other words, the service rate has to be larger than the minimum number of packets that need to be served.

Since each traffic flow is a real-time message, we cannot define a long-term utility for a flow. However, we can define utility functions for subregions. The goal is therefore to allocate the communication resource fairly to subregions instead of users. The type of fairness is defined by the selected utility function. So, the problem is formulated as follows:

$$\max_{\mu\in\mathcal{C}, 0\leq q_{ij}\leq 1} \sum_{i,j\in\mathcal{M}} U_{ij}(q_{ij}) \tag{4.3}$$

subject to

$$\lambda_{ij} q_{ij} \leq \mu_{ij} \text{ for all } i,j \in \mathcal{M}.$$

We will denote the optimal solution by μ^*, q^*.

4.3 A Duality Theory Approach

Using duality theory, we will show how to solve the optimization problem by solving a set of related subproblems. This problem decomposition will be the basis for the online algorithm that we will present in the next section.

The associated dual function [25] for (4.3) is

$$D(\delta) \overset{def}{=} \max_{\mu \in \mathcal{C}, 0 \le q_{ij} \le 1} \sum_{i,j \in \mathcal{M}} U_{ij}(q_{ij}) - \delta_{ij}(\lambda_{ij} q_{ij} - \mu_{ij}).$$

Since the utility function is concave, and the constraints are affine functions, Slater's condition [26] implies that the duality gap is zero and therefore $D(\delta^*) = \sum_{i,j \in \mathcal{M}} U_{ij}(q^*_{ij})$, where

$$\delta^* \in \arg\min_{\delta_{ij} \ge 0} D(\delta)$$

and q^* is the solution to (4.3). Furthermore, to solve the optimization problem using the dual function, we note that we can simply solve the following subproblems

$$\max_{0 \le q_{ij} \le 1} U_{ij}(q_{ij}) - \delta_{ij} \lambda_{ij} q_{ij}$$

and

$$\max_{\mu \in \mathcal{C}} \sum_{i,j \in \mathcal{M}} \delta_{ij} \mu_{ij}. \tag{4.4}$$

Since $\delta_{ij} \ge 0$ for all $i, j \in \mathcal{M}$, the optimization in (4.4) has a linear objective, and the service rate is a convex combination of the feasible schedules, we can further decompose (4.4) as follows

$$\max_{s \in \mathcal{S}(a,c)} \sum_{i,j \in \mathcal{M}} \delta_{ij} \sum_{t \in \mathcal{T}} s_{ijt}.$$

We can then use the following iterative algorithm to find the solution to our optimization problem, where k is the step index:

$$\tilde{q}^*_{ij}(k) \in \arg\max_{0 \le q_{ij} \le 1} U_{ij}(q_{ij}) - \delta_{ij}(k) \lambda_{ij} q_{ij}$$

and

$$\tilde{s}^*(a, c, k) \in \arg\max_{s \in \mathcal{S}(a,c)} \sum_{i,j \in \mathcal{M}} \delta_{ij}(k) \sum_{t \in \mathcal{T}} s_{ijt},$$

with update equation

$$\delta_{ij}(k+1) = \{\delta_{ij}(k) + \epsilon[\lambda_{ij}\tilde{q}^*_{ij}(k) - \tilde{\mu}^*_{ij}(k)]\}^+,$$

step-size parameter $\epsilon > 0$ and

$$\tilde{\mu}^*_{ij}(k) \stackrel{def}{=} \sum_{a,c}\sum_{t\in\mathcal{T}} \tilde{s}^*_{ijt}(a,c,k)Pr(a)Pr(c).$$

Using the change of variables $\epsilon\hat{d}_{ij}(k) = \delta_{ij}(k)$ we can rewrite the problem as

$$\tilde{q}^*_{ij}(k) \in \arg\max_{0\le q_{ij}\le 1} \frac{1}{\epsilon}U_{ij}(q_{ij}) - \hat{d}_{ij}(k)\lambda_{ij}q_{ij}$$

and

$$\tilde{s}^*(a,c,k) \in \arg\max_{s\in\mathcal{S}(a,c)} \sum_{i,j\in\mathcal{M}} \hat{d}_{ij}(k)\sum_{t\in\mathcal{T}} s_{ijt},$$

with update equation

$$\hat{d}_{ij}(k+1) = [\hat{d}_{ij}(k) + \lambda_{ij}\tilde{q}^*_{ij}(k) - \tilde{\mu}^*_{ij}(k)]^+.$$

It must be noted that given the update equation of $\tilde{q}^*_{ij}(k)$, we can give a controller interpretation to it, where we regulate the minimum service we give to traffic requests. Similarly, $\hat{d}_{ij}(k)$ can have a queue interpretation, with the the number of arrivals given by $\lambda_{ij}\tilde{q}^*_{ij}(k)$ and the departures given by $\tilde{\mu}^*_{ij}(k)$.

4.4 Online Algorithm

In this section we first present our online algorithm and subsequently we present its performance analysis.

4.4.1 Scheduler and Service Controller

We propose to use the following scheduler when the arrivals and channel state at frame k are given by $a(k)$ and $c(k)$, respectively:

$$\tilde{s}^*(a(k),c(k),d(k)) \in \arg\max_{s\in\mathcal{S}(a(k),c(k))} \sum_{i,j\in\mathcal{M}} d_{ij}(k)\sum_{t\in\mathcal{T}} s_{ijt}, \tag{4.5}$$

and the following service controller

$$\tilde{q}^*_{ij}(a(k), d(k)) \in \\ \arg\max_{0 \le q_{ij} \le 1} \frac{1}{\epsilon} U_{ij}(q_{ij}) - d_{ij}(k) a_{ij}(k) q_{ij}. \tag{4.6}$$

In the notation we make explicit the fact that the scheduler and the service controller are a function of the arrivals, the channel state, and the parameter $d(k)$.

We need to translate the minimum service to message requests, which is a fraction, into a minimum number of packets that need to be served. This conversion can be made in different ways: we assume that the minimum number of packets to be served at region r_i destined to region r_j, $\tilde{a}_{ij}(k)$, are a binomial random variable with parameters $a_{ij}(k)$ and $\tilde{q}^*_{ij}(a(k), d(k))$. The quantity $\tilde{a}_{ij}(k)$ can be generated by the network as follows: for every message request, flip a coin with probability of *heads* equal to $\tilde{q}^*_{ij}(a(k), d(k))$, and let $\tilde{a}_{ij}(k)$ be the number of *heads* that we get.

The update equation for $d(k)$ is given by

$$d_{ij}(k+1) = [d_{ij}(k) + \tilde{a}_{ij}(k) - \tilde{I}^*_{ij}(a(k), c(k), d(k))]^+,$$

where

$$\tilde{I}^*_{ij}(a(k), c(k), d(k)) \overset{def}{=} \sum_{t \in \mathcal{T}} \tilde{s}^*_{ijt}(a(k), c(k), d(k)).$$

We interpret the parameter $d_{ij}(k)$ as a virtual queue that keeps track of the deficit in service for traffic requests from region r_i to region r_j, given the minimum service allocated by our controller.

4.4.2 Performance Analysis

We will first bound the expected drift of the Markov chain $d(k)$ for a suitable Lyapunov function. For the sake of readability, we will defer the proofs to Section 4.6.

Lemma 9. *Consider the Lyapunov function $V(d) = \frac{1}{2}\sum_{i,j \in \mathcal{M}} d^2_{ij}$. If there exists $\tilde{\mu} \in \mathcal{C}$ and $0 \le \tilde{q}_{ij} \le 1$ for all $i, j \in \mathcal{M}$ such that*

$$\lambda_{ij}\tilde{q}_{ij} < \tilde{\mu}_{ij} \text{ for all } i, j \in \mathcal{M} \tag{4.7}$$

then

$$E\left[V(d(k+1))|d(k)=d\right]-V(d)\leq B_1-B_2\sum_{i,j\in\mathcal{M}}d_{ij}$$
$$-\frac{1}{\epsilon}\sum_{i,j\in\mathcal{M}}\left\{U_{ij}(\tilde{q}_{ij})-E\left[U_{ij}(\tilde{q}^*_{ij}(a(k),d))\right]\right\}$$

for some positive constants B_1, B_2, any $\epsilon > 0$, where $\tilde{q}^*(a(k), d)$ is the solution to (4.6). ◇

Since $d(k)$ defines an irreducible and aperiodic Markov chain, and the last term of the right hand side of the inequality can be bounded, Lemma 9 implies that $d(k)$ is positive recurrent since the expected drift is negative but for a finite set of values of $d(k)$. Thus, a direct consequence of Lemma 9 is the fact that the total service deficit has an $O(\frac{1}{\epsilon})$ bound.

Corollary 10. *If there exists $\tilde{\mu} \in \mathcal{C}$, $0 \leq \tilde{q}_{ij} \leq 1$ for all $i, j \in \mathcal{M}$ such that* (4.7) *is true, then the total expected service deficit is upper-bounded by*

$$\limsup_{k\to\infty} E\left[\sum_{i,j\in\mathcal{M}}d_{ij}(k)\right]\leq B_3+\frac{1}{\epsilon}B_4,$$

where $B_3 = B_1/B_2$ and

$$B_4\leq\frac{\sum_{i,j\in\mathcal{M}}\max_{0\leq q_{ij}\leq 1}2|U_{ij}(q_{ij})|}{B_2}.$$

◇

Before we can prove that our algorithm can achieve the optimal value of (4.3) in some stochastic sense, we need a related result to Lemma 9.

Lemma 11. *Consider the Lyapunov function $V(d) = \frac{1}{2}\sum_{i,j\in\mathcal{M}} d_{ij}^2$. Then*

$$E\left[V(d(k+1))|d(k)=d\right]-V(d)\leq B_1-B_2\sum_{i,j\in\mathcal{M}}d_{ij}$$
$$-\frac{1}{\epsilon}\sum_{i,j\in\mathcal{M}}\left\{U_{ij}(q^*_{ij})-E\left[U_{ij}(\tilde{q}^*_{ij}(a(k),d))\right]\right\}$$

for $B_1 > 0$, some nonnegative constant B_2, any $\epsilon > 0$, where q^* is the solution to (4.3) *and* $\tilde{q}^*(a(k), d)$ *is the solution to* (4.6)*. Furthermore, if $\lambda_{ij} q^*_{ij} < \mu^*_{ij}$ for all $i, j \in \mathcal{M}$ then $B_2 > 0$.* $\diamond$

The difference between Lemma 9 and Lemma 11 is that Lemma 11 does not guarantee that the Markov chain is positive recurrent, but it allows us to compare the expected drift to the optimal solution. The proof technique is identical to the proof for Lemma 9, so it is omitted. With this result, we can now prove that our algorithm is within $O(\epsilon)$ of the optimal value.

Theorem 12. *For any $\epsilon > 0$ we have that*

$$\limsup_{K\to\infty} \sum_{i,j\in\mathcal{M}} \Big\{ U_{ij}(q^*_{ij}) \\ - U_{ij}\left(E\left[\frac{1}{K} \sum_{k=1}^{K} \tilde{q}^*_{ij}(a(k), d(k)) \right] \right) \Big\} \leq B\epsilon$$

for some $B > 0$, where q^* is the solution to (4.3) *and* $\tilde{q}^*(a(k), d(k))$ *is the solution to* (4.6)*.* $\diamond$

From Corollary 10 and Theorem 12 we observe that there is a tradeoff when choosing ϵ, since the more we approach the optimal solution, the larger the total deficit counters will be.

The statement and proofs of Lemma 9 and Theorem 12 follow the techniques in [27], which are similar to the techniques in [28]. Slightly different results can be derived using the techniques in [29] and [30].

4.5 A Different Utility Function

In the previous sections we assumed that the utility is a function of the minimum *fraction* of packets that need to be served. If we assume instead that the utility is a function of the minimum *rate* of packets that need to be served, the algorithm needs to be slightly modified. Since the analysis is similar, in this section we will only highlight the differences and present the main results without proof. Later in Section 4.7 we will present a simulation-based study to show how the choice of the utility function can considerably affect users' behavior.

Denote by $x = (x_{ij})_{i,j\in\mathcal{M}}$ the minimum rate of packets that need to be served originating in region r_i that are destined for region r_j. The utility function associated with such rate is denoted by $U_{ij}(x_{ij})$, and we assume that $U_{ij}()$ is concave. Thus, we have that the constraint for the overall expected service is now given by

$$x_{ij} \leq \mu_{ij} \text{ for all } i, j \in \mathcal{M}.$$

Given that we want to maximize the total network utility, the problem is formulated as follows:

$$\max_{\mu \in \mathcal{C}, x_{ij} \geq 0} \sum_{i,j \in \mathcal{M}} U_{ij}(x_{ij}) \tag{4.8}$$

subject to

$$x_{ij} \leq \mu_{ij} \text{ for all } i, j \in \mathcal{M}.$$

We will denote the optimal solution by μ^*, x^*.

Using the decomposition approach presented in Section 4.3, we can develop the following online algorithm, where the arrivals and channel state at frame k are given by $a(k)$ and $c(k)$. The scheduler is given by

$$\tilde{s}^*(a(k), c(k), d(k)) \in \operatorname*{arg\,max}_{s \in \mathcal{S}(a(k), c(k))} \sum_{i,j \in \mathcal{M}} d_{ij}(k) \sum_{t \in \mathcal{T}} s_{ijt},$$

and the service controller is

$$\tilde{x}^*_{ij}(d(k)) \in \operatorname*{arg\,max}_{0 \leq x_{ij} \leq T} \frac{1}{\epsilon} U_{ij}(x_{ij}) - d_{ij}(k) x_{ij}. \tag{4.9}$$

To convert $\tilde{x}^*_{ij}(d(k))$ into a minimum number of packets that need to be served, we define the integer-valued random variable $\tilde{a}_{ij}(k)$ such that $E[\tilde{a}_{ij}(k)] = \tilde{x}^*_{ij}(d(k))$, its variance is upper-bounded by σ^2, and is such that $Pr(\tilde{a}_{ij}(k) = 0) > 0$, $Pr(\tilde{a}_{ij}(k) = 1) > 0$. The last two assumptions are used to guarantee that the Markov chain $d(k)$ is both irreducible and aperiodic, but can be substituted by similar assumptions.

The update equation for $d(k)$ is given by

$$d_{ij}(k+1) = [d_{ij}(k) + \tilde{a}_{ij}(k) - \tilde{I}^*_{ij}(a(k), c(k), d(k))]^+,$$

where

$$\tilde{I}^*_{ij}(a(k), c(k), d(k)) \overset{def}{=} \sum_{t \in \mathcal{T}} \tilde{s}^*_{ijt}(a(k), c(k), d(k)).$$

As we did before, let $d_{ij}(k)$ be interpreted as a virtual queue that keeps track of the deficit in service for traffic requests from region r_i to region r_j, given the minimum service allocated by our controller.

For this algorithm we have that the expected drift of the Markov chain $d(k)$ for a suitable Lyapunov function is given by the following lemma.

Lemma 13. *Consider the Lyapunov function* $V(d) = \frac{1}{2}\sum_{i,j\in\mathcal{M}} d_{ij}^2$. *If there exists* $\tilde{\mu} \in \mathcal{C}$, $\tilde{x}_{ij} \geq 0$ *for all* $i,j \in \mathcal{M}$ *such that*

$$\tilde{x}_{ij} < \tilde{\mu}_{ij} \text{ for all } i,j \in \mathcal{M} \tag{4.10}$$

then

$$E\left[V(d(k+1))|d(k)=d\right] - V(d) \leq B_1 - B_2 \sum_{i,j\in\mathcal{M}} d_{ij}$$
$$-\frac{1}{\epsilon}\sum_{i,j\in\mathcal{M}} \left\{U_{ij}(\tilde{x}_{ij}) - E\left[U_{ij}(\tilde{x}^*_{ij}(d))\right]\right\}$$

for some positive constants B_1, B_2, *any* $\epsilon > 0$, *where* $\tilde{x}^*(d)$ *is the solution to* (4.9). ◇

Lemma 13 implies that the total service has an $O(\frac{1}{\epsilon})$ bound, as was proved in Section 4.4.2 for Lemma 9. We can also prove that the algorithm is within $O(\epsilon)$ of the optimal value.

Theorem 14. *For any* $\epsilon > 0$ *we have that*

$$\limsup_{K\to\infty} \sum_{i,j\in\mathcal{M}} \{U_{ij}(x^*_{ij})$$
$$-U_{ij}\left(E\left[\frac{1}{K}\sum_{k=1}^{K}\tilde{a}^*_{ij}(k)\right]\right)\right\} \leq B\epsilon$$

for some $B > 0$, *where* x^* *is the solution to* (4.8) *and* $\tilde{a}^*(k)$ *is given by the solution to* (4.9). ◇

4.6 Proofs

4.6.1 Proof of Lemma 9

We start proving Lemma 9 by presenting the following fact.

Fact 1. *The optimization in (4.5) can be performed over* $\mathcal{S}(a(k), c(k))_{\mathcal{CH}}$*, the convex hull of* $\mathcal{S}(a(k), c(k))$*; that is,*

$$\max_{s \in \mathcal{S}(a(k),c(k))} \sum_{i,j \in \mathcal{M}} d_{ij}(k) \sum_{t \in \mathcal{T}} s_{ijt} =$$

$$\max_{s \in \mathcal{S}(a(k),c(k))_{\mathcal{CH}}} \sum_{i,j \in \mathcal{M}} d_{ij}(k) \sum_{t \in \mathcal{T}} s_{ijt}.$$

The reason for this comes from the fact that the objective function is linear and therefore there must be an optimal point $\tilde{s}^*(a(k), c(k), d(k)) \in \mathcal{S}(a(k), c(k))$*.* $\diamond$

Proof of Lemma 9.

$$
\begin{aligned}
&E\left[V(d(k+1))|d(k)=d\right]-V(d)\\
=&E\left[\frac{1}{2}\sum_{i,j\in\mathcal{M}}\{[d_{ij}+\tilde{a}_{ij}(k)-\tilde{I}^*_{ij}(a(k),c(k),d)]^+\}^2\right]\\
&-\frac{1}{2}\sum_{i,j\in\mathcal{M}}d_{ij}^2\\
\leq&E\left[\frac{1}{2}\sum_{i,j\in\mathcal{M}}[d_{ij}+\tilde{a}_{ij}(k)-\tilde{I}^*_{ij}(a(k),c(k),d)]^2\right]\\
&-\frac{1}{2}\sum_{i,j\in\mathcal{M}}d_{ij}^2\\
=&E\left[\sum_{i,j\in\mathcal{M}}d_{ij}[\tilde{a}_{ij}(k)-\tilde{I}^*_{ij}(a(k),c(k),d)]\right.\\
&\left.+\frac{1}{2}\sum_{i,j\in\mathcal{M}}[\tilde{a}_{ij}(k)-\tilde{I}^*_{ij}(a(k),c(k),d)]^2\right]\\
\leq&E\left[\sum_{i,j\in\mathcal{M}}d_{ij}\tilde{a}_{ij}(k)-d_{ij}\tilde{I}^*_{ij}(a(k),c(k),d)\right.\\
&\left.+\frac{1}{2}\sum_{i,j\in\mathcal{M}}\tilde{a}^2_{ij}(k)+a^2_{ij}(k)\right] \qquad (4.11)\\
\leq&E\left[\sum_{i,j\in\mathcal{M}}d_{ij}\tilde{a}_{ij}(k)-d_{ij}\tilde{I}^*_{ij}(a(k),c(k),d)+a^2_{ij}(k)\right] \qquad (4.12)\\
=&B_1+E\left[\sum_{i,j\in\mathcal{M}}d_{ij}a_{ij}(k)\tilde{q}^*_{ij}(a(k),d)\right.\\
&\left.-d_{ij}\tilde{I}^*_{ij}(a(k),c(k),d)\right]\\
=&B_1-E\left[\sum_{i,j\in\mathcal{M}}\frac{1}{\epsilon}U_{ij}(\tilde{q}^*_{ij}(a(k),d))-d_{ij}a_{ij}(k)\tilde{q}^*_{ij}(a(k),d)\right.\\
&\left.+d_{ij}\tilde{I}^*_{ij}(a(k),c(k),d)-\frac{1}{\epsilon}U_{ij}(\tilde{q}^*_{ij}(a(k),d))\right]
\end{aligned}
$$

where (4.11) and (4.12) follow from the definition of $\tilde{I}^*_{ij}(a(k), c(k), d)$ and $\tilde{a}_{ij}(k)$, respectively, and

$$B_1 = \sum_{i,j \in \mathcal{M}} \lambda_{ij}^2 + \sigma_{ij}^2.$$

From the definition of $\mathcal{C}$, $\tilde{\mu} \in \mathcal{C}$ implies that there exist $\tilde{\mu}(a, c) \in \mathcal{C}(a, c)$ for all a, c and $\tilde{\mu}_{ij} = E[\tilde{\mu}_{ij}(a, c)]$ for all $i, j \in \mathcal{M}$. For the rest of the proof we define $\tilde{\mu}_{ij}(a, c)$ to be such set of values associated to $\tilde{\mu}$. Thus:

$$\begin{aligned}
&E\left[V(d(k+1))|d(k) = d\right] - V(d) \\
\leq &B_1 - E\left[\sum_{i,j \in \mathcal{M}} \frac{1}{\epsilon} U_{ij}(\tilde{q}_{ij}) - d_{ij} a_{ij}(k) \tilde{q}_{ij} \right. \qquad (4.13) \\
&\left. + d_{ij} \tilde{\mu}_{ij}(a(k), c(k)) - \frac{1}{\epsilon} U_{ij}(\tilde{q}^*_{ij}(a(k), d))\right] \\
= &B_1 - \sum_{i,j \in \mathcal{M}} d_{ij} (\tilde{\mu}_{ij} - \lambda_{ij} \tilde{q}_{ij}) \\
&- \frac{1}{\epsilon} \sum_{i,j \in \mathcal{M}} \left\{ U_{ij}(\tilde{q}_{ij}) - E\left[U_{ij}(\tilde{q}^*_{ij}(a(k), d))\right] \right\} \\
\leq &B_1 - B_2 \sum_{i,j \in \mathcal{M}} d_{ij} \\
&- \frac{1}{\epsilon} \sum_{i,j \in \mathcal{M}} \left\{ U_{ij}(\tilde{q}_{ij}) - E\left[U_{ij}(\tilde{q}^*_{ij}(a(k), d))\right] \right\}
\end{aligned}$$

where (4.13) follows from the fact that $\tilde{I}^*_{ij}(a(k), c(k), d)$ and $\tilde{q}^*_{ij}(a(k), d)$ are the solutions to (4.5) and (4.6), respectively, and Fact 1. Furthermore,

$$B_2 = \min_{i,j \in \mathcal{M}} \left\{ \tilde{\mu}_{ij} - \lambda_{ij} \tilde{q}_{ij} \right\}.$$

□

4.6.2 Proof of Theorem 12

From Lemma 11 we have

$$
\begin{aligned}
&\frac{1}{\epsilon}\sum_{i,j\in\mathcal{M}}\left\{U_{ij}(q_{ij}^*) - E\left[U_{ij}(\tilde{q}_{ij}^*(a(k),d))\right]\right\}\\
\leq &B_1 - B_2\sum_{i,j\in\mathcal{M}} d_{ij} - E\left[V(d(k+1))|d(k)=d\right] + V(d)\\
\leq &B_1 - E\left[V(d(k+1))|d(k)=d\right] + V(d).
\end{aligned}
$$

The last inequality follows from the fact that $B_2\sum_{i,j\in\mathcal{M}} d_{ij} \geq 0$. Taking expectations we obtain

$$
\begin{aligned}
&\frac{1}{\epsilon}\sum_{i,j\in\mathcal{M}}\left\{U_{ij}(q_{ij}^*) - E\left[U_{ij}(\tilde{q}_{ij}^*(a(k),d(k)))\right]\right\}\\
\leq &B_1 - E\left[V(d(k+1))\right] + E\left[V(d(k))\right].
\end{aligned}
$$

If we add the terms for $k = \{1, \ldots, K\}$ and divide by K we obtain

$$
\begin{aligned}
&\frac{1}{\epsilon}\sum_{i,j\in\mathcal{M}}\left\{U_{ij}(q_{ij}^*) - E\left[\frac{1}{K}\sum_{k=1}^{K} U_{ij}(\tilde{q}_{ij}^*(a(k),d(k)))\right]\right\}\\
\leq &B_1 - \frac{E\left[V(d(K+1))\right]}{K} + \frac{E\left[V(d(1))\right]}{K}\\
\leq &B_1 + \frac{E\left[V(d(1))\right]}{K},
\end{aligned}
$$

where the last inequality follows from the fact that the Lyapunov function is non-negative.

Using Jensen's inequality [26] we get the following

$$
\begin{aligned}
&\frac{1}{\epsilon}\sum_{i,j\in\mathcal{M}}\left\{U_{ij}(q_{ij}^*) - U_{ij}\left(E\left[\frac{1}{K}\sum_{k=1}^{K} \tilde{q}_{ij}^*(a(k),d(k))\right]\right)\right\}\\
\leq &\frac{1}{\epsilon}\sum_{i,j\in\mathcal{M}}\left\{U_{ij}(q_{ij}^*) - E\left[\frac{1}{K}\sum_{k=1}^{K} U_{ij}(\tilde{q}_{ij}^*(a(k),d(k)))\right]\right\}\\
\leq &B_1 + \frac{E\left[V(d(1))\right]}{K}.
\end{aligned}
$$

Taking the limit as $K \to \infty$, and assuming $E\left[V(d(1))\right] < \infty$, we get

$$\limsup_{K\to\infty} \sum_{i,j\in\mathcal{M}} \Big\{U_{ij}(q^*_{ij}) - U_{ij}\left(E\left[\frac{1}{K}\sum_{k=1}^{K}\tilde{q}^*_{ij}(a(k),d(k))\right]\right)\Big\} \leq B\epsilon,$$

where $B = B_1$. ■

4.7 Simulations

In this section, we first compare our algorithm against the solution proposed in [27] for scheduling persistent real-time traffic and show the limitations of that approach to handle real-time messages for providing fairness. Then we study how the choice of the utility function in our algorithm can affect the behavior of users, giving them incentives either to achieve load-balancing or to create hotspots.

4.7.1 On the Limitations of a Previous Approach

We now compare the algorithm introduced in Section 4.4.1 against the scheduler proposed in [27] for handling persistent real-time traffic. Since the algorithm is an extension to the MaxWeight scheduler for real-time traffic, in this chapter we will call it the *real-time MaxWeight algorithm*, while we will call our algorithm the *fraction-based algorithm*.

4.7.1.1 Simulation settings

We divide the region where traffic is generated in two subregions. In other words, $\mathcal{M} = \{1, 2\}$. There are 80 users in the network, where 40 users are located in subregion 1, and 40 users are in subregion 2. Each frame consists of 4 time slots, i.e., $T = 4$. We assume the channel between any two regions is always on, in other words, $c_{ij} = 1$ for all $i, j \in \mathcal{M}$. At every frame each user generates a message with probability P_m, where the destination is randomly selected. We will denote by (i, j) the set of transmission requests with source in region r_i and destination in r_j, for $i, j \in \mathcal{M}$. Thus, the aggregate number of message requests in (i, j) will determine a_{ij}. The interference graph for our simulations is given by Fig. 4.1, where each

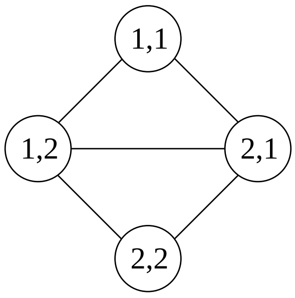

Figure 4.1 Interference graph used in the simulations

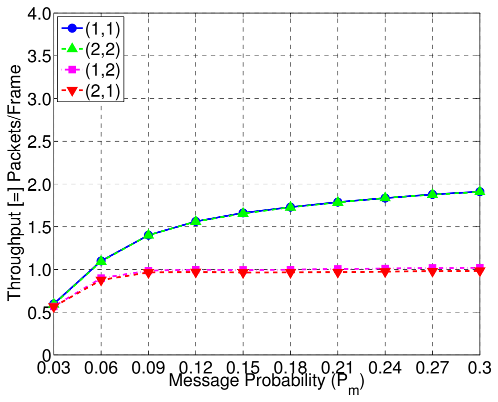

Figure 4.2 Throughput for the Fraction-based algorithm

vertex represents any pair of regions, and an edge joins them if they cannot simultaneously transmit. For example, only transmissions $(1,1)$ and $(2,2)$ can be simultaneously scheduled without interfering with each other. In the simulations, we assume the utility function is α-fair, i.e., $U_{ij}(q_{ij}) = \frac{q_{ij}^{1-\alpha}}{1-\alpha}$. For the fraction-based algorithm, we set $\epsilon = 0.1$ and $\alpha \to 1$, which corresponds to the limit case of proportional fairness.

4.7.1.2 Results

To compare both algorithms, we measure the throughput, which is defined to be the number of packets that are successfully transmitted per frame for every region pair (i,j). In Figs. 4.2 and 4.3 we observe that while the fraction-based algorithm tries to fairly allocate the throughput among different region pairs, the real-time MaxWeight algorithm disproportionally gives preference to intraregion transmissions, starving cross-region transmissions.

To understand this behavior, note that the real-time MaxWeight algorithm uses as weights

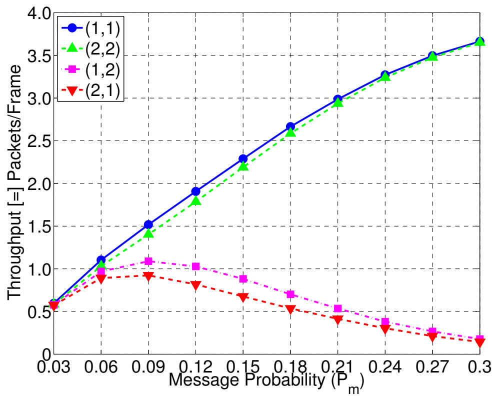

Figure 4.3 Throughput for the Real-time MaxWeight algorithm

the deficit in service for every flow. Since each traffic request consists of a single packet, then no flow has a deficit that allows it to gain priority in a schedule. Thus, for the case of real-time messages, the real-time MaxWeight algorithm becomes the maximal matching algorithm, giving priority to intraregion transmissions since this maximizes the number of links that can be simultaneously scheduled. Therefore, in order to maximize throughput, the real-time MaxWeight algorithm allocates service with no fairness considerations into account.

To explore the tradeoff between maximizing throughput and guaranteeing fairness, we measured the total network throughput for both algorithms, and the results are presented in Fig. 4.4. As it can be seen, for larger arrival rates the difference in both algorithms starts to increase since the real-time MaxWeight algorithm schedules more intraregion transmissions. Hence, in order to achieve proportional fairness we have to pay a price in terms of total network throughput.

Another way to explore the throughput-fairness tradeoff is by increasing the value of α. Note that $\alpha \to \infty$ corresponds to the limit case of max-min fairness, where the algorithm tries to maximize the minimum fraction of service allocated to any given region pair. In Fig. 4.5 we plot the total network throughput when $P_m = 0.2$. As can be seen, the minimum throughput between different region pairs starts to increase with increasing α, but as can be observed in Fig. 4.6 we have to pay a price in terms of a small decrease in total network throughput, since we are sacrificing efficiency for fairness.

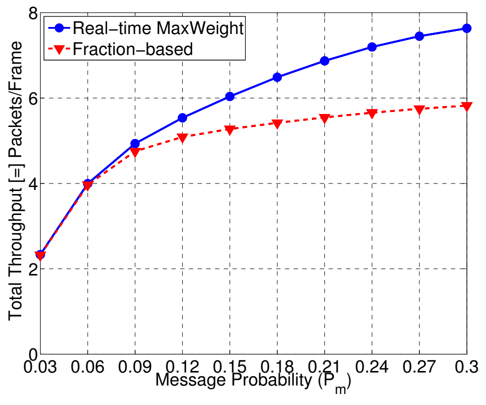

Figure 4.4 Comparison of total network throughput

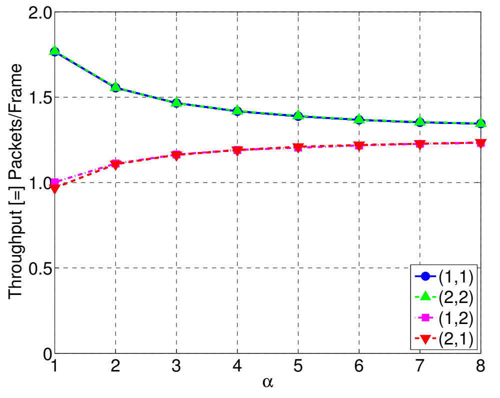

Figure 4.5 Throughput for the Fraction-based algorithm when α varies

4.7.2 The Effect on Users' Behavior of the Utility Function

So far, our work has studied how to schedule real-time messages when the traffic densities in every region are fixed. In this section we complement our theoretical work by studying how the choice of the utility function can affect the behavior of users, giving them incentives either to achieve load-balancing or to create hotspots. To do that, we will study a simulation model where the users are in any given region according to some probability distribution, and we allow users to modify such distribution in order to increase their own throughput. With this simple model we try to understand the decision-making process of users and how they react to the quality of service they receive from the network.

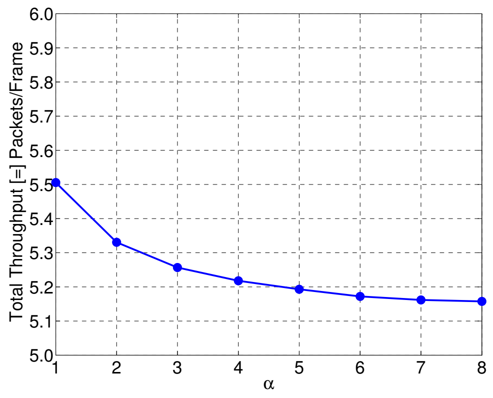

Figure 4.6 Total throughput for the Fraction-based algorithm

4.7.2.1 Simulation Settings

The settings for this section are the same as in Section 4.7.1, with the following changes. We let P_m to be 0.1, and we assume that users are in region 1 with probability 0.8 and in region 2 otherwise. After a random amount of time, each user decides to either continue using the same probability distribution or change it to the inverse, that is, to stay in region 1 with probability 0.2. We assume that the time for our algorithm to converge to the optimal solution is much smaller than the time for an user to decide whether to keep or change its distribution. In other words, we assume there is a time-scale separation between the users and the algorithm. This assumption is reasonable since in practice the behavior of users is much slower than the convergence time of the scheduling algorithm.

4.7.2.2 Results

In the first part of the simulations we set $\alpha = 5$ and compare the fraction-based algorithm, with service controller given by (4.6), against the service controller in (4.9), where the utility is a function of the minimum *rate* of packets that need to be served instead of the minimum *fraction* of packets that need to be served. We will call the second controller the *rate-based algorithm.*

We allow users to modify their probability distribution after a random amount of time, and we let the simulations run until we get the steady-state behavior of the network, which in

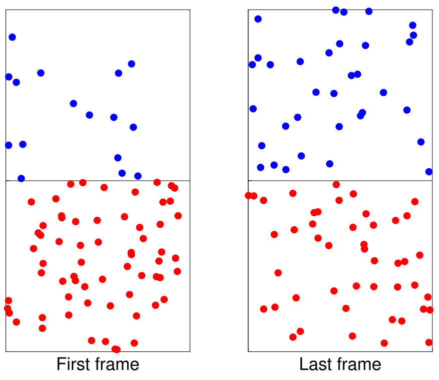

Figure 4.7 Users' location under the Fraction-based algorithm when $\alpha = 5$

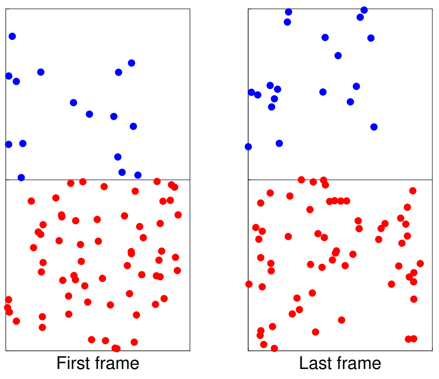

Figure 4.8 Users' location under the Rate-based algorithm when $\alpha = 5$

this case is after 2 million frames. In Figs. 4.7 and 4.8 we plot the geographical location of users in the first and last frame of the simulations. It can be noted that for the fraction-based algorithm roughly half of the users are in each region while for the rate-based algorithm 80% of the users stay in region 1, compared to the distribution of users in the first frame, where 80% of the users stay in region 1. The steady-state distribution of users' location is shown in Table 4.1.

Thus, we observe that the fraction-based algorithm allocates service such that users tend to modify their behavior in such a way that the system achieves load balancing between different regions, and that the rate-based algorithm assigns service in a way that users do not have an incentive to modify their behavior, creating hotspots. This difference in behavior is also

Table 4.1 The steady-state distribution of users' location when $\alpha = 5$

Subregion	Fraction-based	Rate-based
1	50.26%	79.99%
2	49.74%	20.01%

Table 4.2 Total network throughput when $\alpha = 5$

Fraction-based	Rate-based
4.70	4.25

reflected in the total network throughput, as shown in Table 4.2. As expected, when the system achieves load balancing it gets higher throughput compared when it does not.

For the case where $\alpha = 1$, the results are quite different. In Figs. 4.9 and 4.10 we plot the users' location for the first and the last frame of the simulations, while the steady-state distribution of users' location is shown in Table 4.3. As can be seen, both algorithms start with a distribution of users in the first frame such that 80% of the users stay in region 1, but at the end of the simulations they achieve load balancing.

The data implies that when $\alpha = 1$, the performance of the fraction-based algorithm and the rate-based algorithm are equivalent. As we can see in Sections 4.4.1 and 4.5, the only difference between the two algorithms is the service controller. For the fraction-based algorithm, when the utility function is α-fair, we can analytically solve the optimization problem (4.6) such that

$$\tilde{q}^*_{ij}(a(k), d(k)) = \left[\frac{1}{\epsilon a_{ij}(k) d_{ij}(k)}\right]^{\frac{1}{\alpha}}$$

if $\left[\frac{1}{\epsilon a_{ij}(k) d_{ij}(k)}\right]^{\frac{1}{\alpha}} \leq 1$. For the rate-based algorithm, when the utility function is α-fair, we can also solve the optimization problem (4.9) such that

$$\tilde{x}^*_{ij}(d(k)) = \left[\frac{1}{\epsilon d_{ij}(k)}\right]^{\frac{1}{\alpha}}$$

if $\left[\frac{1}{\epsilon d_{ij}(k)}\right]^{\frac{1}{\alpha}} \leq T$. It is easy to see that when $\alpha \rightarrow 1$, both algorithms allocate the same minimum rate of packets to be served.

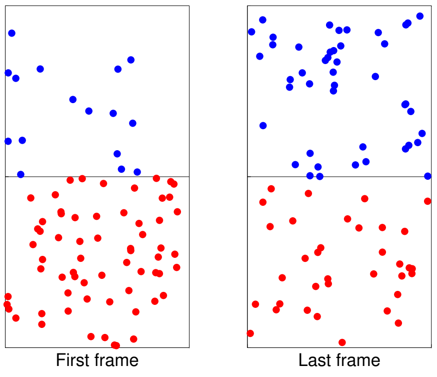

Figure 4.9 Users' location under the Fraction-based algorithm when $\alpha = 1$

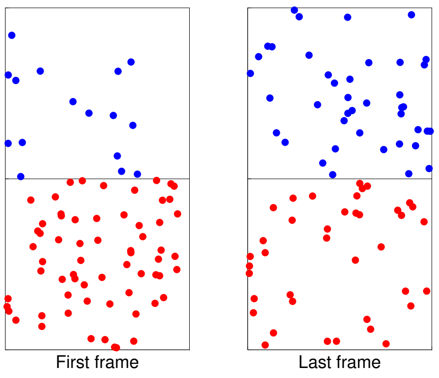

Figure 4.10 Users' location under the Rate-based algorithm when $\alpha = 1$

4.8 Conclusion

In this chapter we have studied the problem of service allocation and scheduling of real-time message requests under strict per-packet deadline constraints. We have presented an optimization framework that groups message requests by regions with similar interference and channel conditions, allowing us to design a solution that optimally allocates resources in the long term while meeting delay constraints. The solution allows for very general interference and arrival models. Using simulations we have showed the limitiations of previous approaches and why there is a need to develop a new solution to the problem we considered.

We have also explored the impact of the choice of the utility function on the mobility patterns of users, and have showed how different utilities lead to incentives that either help

Table 4.3 The steady-state distribution of users' location when $\alpha = 1$

Subregion	Fraction-based	Rate-based
1	51.80%	55.54%
2	48.20%	44.46%

achieve load balancing between regions or that create hotspots where a large number of users concentrate.

CHAPTER 5. Scheduling in Multihop Wireless Networks

In this chapter, I address scheduling in multihop wireless networks with flow-level dynamics. Although most existing practical wireless networks consist of single-hop data transmissions, multihop wireless networks have important applications in various areas.

A widely-used algorithm to stabilize multi-hop flows in wireless networks is the back-pressure algorithm proposed in [3], which can stabilize any traffic flows that can be supported by any other routing/scheduling algorithm. We refer to [9, 10] for a comprehensive survey on the back-pressure algorithm and its variations. A key idea of the back-pressure algorithm is to use largest queue difference as link weight, and schedule the links with largest aggregated weights. Therefore, the back-pressure algorithm requires constant exchange of queue-length information among neighboring nodes. Furthermore, under the back pressure algorithm, the sum of the queue lengths along a route increases quadratically as the route length [18], which leads to poor delay performance. The most important thing is, the back-pressure algorithm is optimal only for the network without flow-level dynamics. When the system has flow-level dynamics, it is no longer throughput optimal as pointed out in [1].

To address the scheduling problem in the presence of flow-level dynamics, we can think of the nodes in wireless ad-hoc networks as "access points" (AP). Each user who wants to transmit data should first associate with a particular AP and transmit data to it, and then the associated AP will forward the data to the destination AP through wireless links, who will finally dump the data to the destination user. By doing this we "translate" the flow-level dynamics to packet-level dynamics. Now the question is, considering the drawbacks of the back-pressure algorithm, *can the network be stabilized without using back pressure?* We address this problem in a multi-hop wireless network with fixed routing in this chapter.

5.1 Basic Model

We consider a network represented by a graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{L}$ is the set of directed links. Let $N = |\mathcal{N}|$ and $L = |\mathcal{L}|$. Denote by (m, n) the link from node m to node n, which implies that node m can communicate with node n. Furthermore, let $\boldsymbol{\mu} = \{\mu_{(m,n)}\}$ denote a link-rate vector such that $\mu_{(m,n)}$ is the transmission rate over link (m, n). A link-rate vector $\boldsymbol{\mu}$ is said to be *admissible* if the link-rates specified by $\boldsymbol{\mu}$ can be achieved *simultaneously*. Define Γ to be the set of all admissible link-rate vectors. It is easy to see that Γ depends on the choice of interference model and might not be a convex set. Furthermore, Γ is time-varying if channels are time-varying. We further assume that there exists $\mu_{\max}$ such that $\mu_{(m,n)} \leq \mu_{\max}$ for all $(m, n) \in \mathcal{L}$ and all admissible $\boldsymbol{\mu}$.

We consider multihop traffic flows with fixed routing in this chapter. Let f denote a flow, and $\mathbf{R}_f$ denote the route associated with flow f. Denote by S_f the source node of flow f, and D_f the destination node of flow f. Further, let $\mu^f_{(m,n)}$ denote the rate at which packets of flow f are served over link (m, n). We use $\mathcal{F}$ to denote the set of all flows in the network, and $F = |\mathcal{F}|$. Assume that time is discretized, and let $X_f(t)$ $(f \in \mathcal{F})$ denote the number of packets injected by flow f at time t. We assume $X_f(t)$ is independent and identical across time, and let $X_f = \mathbf{E}[X_f(t)]$, which is the arrival rate of flow f. We further assume $X_f(t)$ is upper-bounded, i.e., $X_f(t) \leq X^{\max}$ for all f and t. Suppose $A_f(t) = \sum_{\tau=1}^{t} X_f(\tau)$ which is the aggregated arrival rate of flow f at time t, we assume that $A_f(t)$ satisfies the following property:

$$\frac{A_f(s) - A_f(t)}{s - t} \to X_f \tag{5.1}$$

as $s - t \to \infty$. In other words, given any $\epsilon > 0$, we can find a T such that when $s - t > T$, $\left|\frac{A_f(s)-A_f(t)}{s-t} - X_f\right| < \epsilon$ for any f.

Remark: Since $A_f(s) - A_f(t) = \sum_{\tau=t+1}^{s} X_f(\tau)$, according to the Strong Law of Large Numbers, $\Pr\left(\frac{A_f(s)-A_f(t)}{s-t} \to X_f\right) = 1$. However, the sample-path convergence still cannot guarantee deterministic convergence as in (5.1). Therefore, we should consider (5.1) to be the assumption of traffic arrivals, instead of the property of sum of random variables. But we want to point out that when $s - t$ is sufficiently large, (5.1) holds in almost-surely sense.

5.2 Necessary Conditions for Stability

In this section, we study the necessary conditions for network stability. We say a traffic configuration $\{X_f\}_{f\in\mathcal{F}}$ is supportable if there exists $\{\bar{\mu}^f_{(m,n)}\}_{(m,n)\in\mathcal{L}}$ such that the following two conditions hold:

(i) For any flow f and node $n \neq D_f$,

$$X_f \mathbb{I}_{\{S_f=n\}} + \sum_{m:(m,n)\in\mathbf{R}_f} \bar{\mu}^f_{(m,n)} \leq \sum_{b:(n,b)\in\mathbf{R}_f} \bar{\mu}^f_{(n,b)}, \tag{5.2}$$

where $\mathbb{I}_{\{S_f=n\}}$ equals one if $S_f = n$ occurs and equals zero otherwise.

(ii)

$$\left\{\sum_{f\in\mathcal{F}} \bar{\mu}^f_{(m,n)}\right\} \in \mathcal{CH}(\Gamma), \tag{5.3}$$

where $\mathcal{CH}(\Gamma)$ is the convex hull of Γ.

Recall that each flow is associated with a fixed route. It is easy to see the necessary condition (5.2) is equivalent to the following statement: For any flow f and $(m,n) \in \mathbf{R}_f$, we have

$$\bar{\mu}^f_{(m,n)} \geq X_f. \tag{5.4}$$

5.3 Self-Regulated MaxWeight Scheduling for Multihop Wireless Networks

In this section, we introduce the self-regulated MaxWeight scheduling for multhop wireless networks, which is later proved to be throughput optimal.

Two-stage queue architecture: Each node maintains two types of queues: per-flow queues and per-link queues as shown in Figure 5.1. An incoming packet is at first buffered at the corresponding per-flow queue and then moved to the per-link queue where the link is on the route of the flow (the details will be described later). In this chapter, we denote by Q^n_f the length of the queue maintained at node n for flow f, and $Q^n_{(n,b)}$ the length of the queue maintained at

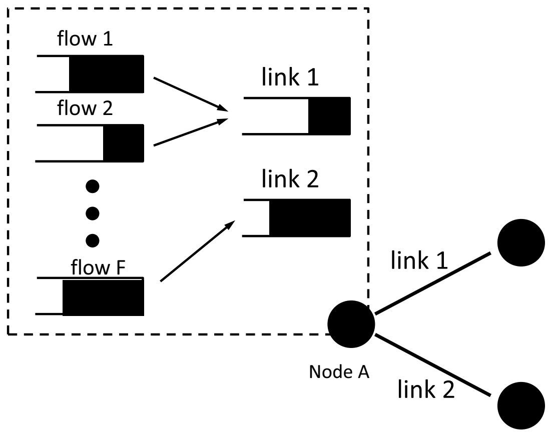

Figure 5.1 The two-stage queue architecture

node n for link (n, b). Clearly, queue for link (n, b) is maintained only at node n, so we simplify $Q^n_{(n,b)}$ to be $Q_{(n,b)}$ without causing any confusion.

Self-Regulated MaxWeight Scheduling

- MaxWeight scheduling: Compute the admissible link-rate vector $\mu^*(t)$ such that

$$\mu^*(t) = \arg \max_{\boldsymbol{\mu}(t)\in\Gamma} \sum_{(n,b)\in\mathcal{L}} \mu_{(n,b)}(t)Q_{(n,b)}(t). \tag{5.5}$$

  Note that here instead of back pressure, we use the queue lengths of per-link queues as link weights to make scheduling decisions.

- Per-link queue transmission: Node n transmits

$$s_{(n,b)}(t) \triangleq \min\{\mu^*_{(n,b)}(t), Q_{(n,b)}(t)\}$$

  packets to node b over link (n, b). The packets are deposited into per-flow queues at node b according to their flows. We let $s^f_{(n,b)}(t)$ denote the number of packets of flow f that are transmitted over link (n, b) at time t. Note that

$$s_{(n,b)}(t) = \sum_f s^f_{(n,b)}(t)$$

  always holds.

- Per-flow queue transmission: Denote by $a_f^n(t)$ the number of packets deposited into queue Q_f^n at time slot t, i.e.,

$$a_f^n(t) = \sum_{(b,n)\in\mathcal{L}} s_{(b,n)}^f(t)$$

for those non-source nodes, and

$$a_f^{S_f}(t) = X_f(t)$$

for the source node of flow f.

For each flow f, node n maintains a rate estimate

$$\tilde{X}_f^n(t) = (1 + \frac{1}{Q_{(n,n_f)}(t)})\frac{\sum_{\tau=1}^t a_f^n(\tau)}{t}, \tag{5.6}$$

where n_f is the next hop of node n on the route of flow f. If $Q_{(n,n_f)}(t) = 0$, then let

$$\tilde{X}_f^n(t) = (1+\gamma)\frac{\sum_{\tau=1}^t a_f^n(\tau)}{t},$$

where $\gamma > 1$ is a positive constant we can set. At time slot t, node n moves

$$s_f^n(t) \triangleq \min\{\tilde{X}_f^n(t), Q_f^n(t)\}$$

packets from queue Q_f^n to queue $Q_{(n,n_f)}$. We further define the packet arrivals of per-link queues

$$a_{(n,b)}(t) \triangleq \sum_{f:(n,b)\in\mathbf{R}_f} s_f^n(t).$$

Remark: Note that the rate estimate (5.6) is self-regulated. Under low traffic load, the queue lengths of per-link queues are small, so the transfer rates from per-flow queues to per-link queues are large to get a good performance on end-to-end packet transmission delay. When the traffic is heavy, i.e., the queue lengths of per-link queues are large, the rate estimate is only slightly larger than the required departure rate of per-flow queue to guarantee the stability of the network. This scheme can stabilize the network without sacrificing any portion of the stability region, which will be proved later.

$\square$

The intuition of this two-stage queue architecture is that, we break each multi-hop flow into multiple single-hop flows, one for each link on the route. A scheduling policy stabilizing the collection of single-hop flows also provides sufficient service for supporting the set of multi-hop flows. Therefore, if each link knows the required rate for it to carry, it can simply generate virtual packets according to the required rate and then let the network makes scheduling decisions according to the virtual queues (per-link queues). When a virtual queue is scheduled, real packets are served according to the allocated link rate. Note that the back pressure scheduling becomes the MaxWeight scheduling here since all flows are single-hop. To totally remove the network overhead, we let each node estimates the required link rate locally by taking average over the past arrivals as in (5.6).

The notations are illustrated in Figure 5.2. From the definition of the self-regulated MaxWeight scheduling algorithm, we can see that the dynamics of queue Q_f^n and $Q_{(n,b)}$ are:

$$Q_f^n(t+1) = \left[Q_f^n(t) - s_f^n(t)\right]^+ + a_f^n(t) \tag{5.7}$$

$$Q_{(n,b)}(t+1) = \left[Q_{(n,b)}(t) - s_{(n,b)}(t)\right]^+ + a_{(n,b)}(t). \tag{5.8}$$

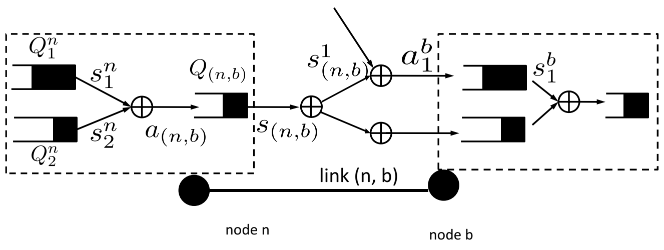

Figure 5.2 The notations and the flows

In the following, we will prove that the self-regulated MaxWeight scheduling algorithm stabilizes any traffic within the stability region of the network. The analysis consists of two steps: we first show that the per-link queues are bounded (Lemma 15), and then using induction to prove that the per-flow queues are bounded as well (Lemma 16).

Lemma 15. *Given a set of traffic flows such that* $\{(1+\epsilon)X_f\}$ *is supportable for some* $\epsilon > 0$, *there exists a positive constant* $C > 0$ *such that the lengths of any per-link queue is no more than* C *for all* $t \geq 0$.

Proof. Since we are considering the asymptotic behavior of the system, we only concern about the queue lengths when t is large. From (5.1) we know that for any given $\delta > 0$, we can find $T(\delta)$, such that when $t > T(\delta)$,

$$\frac{A_f(t)}{t} \leq X_f + \delta, \forall f.$$

Therefore, for any node n,

$$\sum_{\tau=1}^{t} a_f^n(\tau) \leq X_f t + \delta t,$$

which implies that

$$\tilde{X}_f^n(t) \leq (1 + \min\left\{\gamma, \frac{1}{Q_{(n,n_f)}(t)}\right\})\left(X_f + \delta\right), \forall f.$$

Since $\{(1+\epsilon)X_f\}$ is supportable, according to the necessary conditions for stability, there exist $\bar{\mu}$ such that

$$\bar{\mu}_{(m,n)}^f \geq (1 + \epsilon)X_f$$

for all flow f and link (m,n) that is on the route of f. Let $\bar{\mu}_{(m,n)} = \sum_{f:(m,n)\in\mathbf{R}_f} \bar{\mu}_{(m,n)}^f$, then for any link $(m,n) \in \mathcal{L}$, we can conclude that

$$\begin{aligned}
a_{(m,n)}(t) &= \sum_{f:(m,n)\in\mathbf{R}_f} \tilde{X}_f^m(t) \\
&\leq \sum_{f:(m,n)\in\mathbf{R}_f} (1+\min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\})\,(X_f+\delta) \\
&= \sum_{f:(m,n)\in\mathbf{R}_f} (1+\epsilon)X_f \\
&\quad - \sum_{f:(m,n)\in\mathbf{R}_f} (\epsilon-\min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\})X_f \\
&\quad + \sum_{f:(m,n)\in\mathbf{R}_f} (1+\min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\})\delta \\
&\leq \sum_{f:(m,n)\in\mathbf{R}_f} \bar{\mu}^f_{(m,n)} \\
&\quad - \sum_{f:(m,n)\in\mathbf{R}_f} (\epsilon-\min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\})X_f \\
&\quad + \sum_{f:(m,n)\in\mathbf{R}_f} (1+\min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\})\delta \\
&= \bar{\mu}_{(m,n)} \\
&\quad - \sum_{f:(m,n)\in\mathbf{R}_f} (\epsilon-\min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\})X_f \\
&\quad + \sum_{f:(m,n)\in\mathbf{R}_f} (1+\min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\})\delta
\end{aligned} \tag{5.9}$$

The dynamic of per-link queues can be rewritten as

$$Q_{(m,n)}(t+1) = Q_{(m,n)}(t) - s_{(m,n)}(t) + u_{(m,n)}(t) + a_{(m,n)}(t),$$

where $u_{(m,n)}(t)$ is the unused service due to the lack of packets in queue. It is easy to see that when $Q_{(m,n)}(t) > \mu_{\max}$, $u_{(m,n)}(t) = 0$, so $Q_{(m,n)}(t) \times u_{(m,n)}(t) \leq \mu^2_{\max}$.

We construct Lyapunov function as follows:

$$V(t) = \sum_{(m,n)\in\mathcal{L}} Q^2_{(m,n)}(t).$$

Then

$$\begin{aligned}
& V(t+1) - V(t) \\
= & \sum_{(m,n)\in\mathcal{L}} Q^2_{(m,n)}(t+1) - \sum_{(m,n)\in\mathcal{L}} Q^2_{(m,n)}(t) \\
= & \sum_{(m,n)\in\mathcal{L}} \left(Q_{(m,n)}(t+1) + Q_{(m,n)}(t)\right) \times \\
& \left(Q_{(m,n)}(t+1) - Q_{(m,n)}(t)\right) \\
= & \sum_{(m,n)\in\mathcal{L}} \left(2Q_{(m,n)}(t) + a_{(m,n)}(t) + u_{(m,n)}(t) - s_{(m,n)}(t)\right) \\
& \times \left(a_{(m,n)}(t) + u_{(m,n)}(t) - s_{(m,n)}(t)\right) \\
\leq & M_1 + \sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t) \left(a_{(m,n)}(t) - s_{(m,n)}(t)\right) \\
= & M_1 \\
& + \sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t) \left(a_{(m,n)}(t) - \bar{\mu}_{(m,n)}\right) \qquad (5.10) \\
& + \sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t) \left(\bar{\mu}_{(m,n)} - s_{(m,n)}(t)\right), \qquad (5.11)
\end{aligned}$$

where $M_1 = 2L\mu^2_{\max} + L\left[(\max_f X_f + \delta) \times F + \mu_{\max}\right]^2$.

From inequality (5.9), we can get

$$\begin{aligned}
& \sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t) \times \left(a_{(m,n)}(t) - \bar{\mu}_{(m,n)}\right) \\
\leq & \sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t) \left(\sum_{f:(m,n)\in\mathbf{R}_f} (1 + \min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\})\delta \right. \\
& \left. - \sum_{f:(m,n)\in\mathbf{R}_f} (\epsilon - \min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\})X_f \right) \\
\leq & \sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t) \left(\min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\} \left(\sum_{f\in\mathcal{F}} X_f + F\delta\right) \right. \\
& \left. + F\delta - \sum_{f:(m,n)\in\mathbf{R}_f} \epsilon X_f \right) \\
\leq & M_2 + \sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t) \left(F\delta - \sum_{f:(m,n)\in\mathbf{R}_f} \epsilon X_f \right)
\end{aligned}$$

where $M_2 = 2L\left(\sum_{f\in\mathcal{F}} X_f + F\delta\right)$.

Furthermore, from the MaxWeight scheduler (5.5) and necessary condition (5.3),

$$\begin{aligned}
&\sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t) \times \left(\bar{\mu}_{(m,n)} - s_{(m,n)}(t)\right) \\
\leq &\sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t) \times \left(\bar{\mu}_{(m,n)} - \mu^*_{(m,n)}(t)\right) + M_3 \\
\leq &\; M_3,
\end{aligned}$$

where $M_3 = 2L\mu_{\max}^2$.

Therefore, we conclude that

$$\begin{aligned}
&V(t+1) - V(t) \\
\leq &\; M_1 + M_2 + M_3 \\
+ &\sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t) \left(F\delta - \sum_{f:(m,n)\in\mathbf{R}_f} \epsilon X_f\right).
\end{aligned}$$

It is easy to see that when $\{f \in \mathcal{F} : (m,n) \in \mathbf{R}_f\} = \emptyset$, $Q_{(m,n)}(t) = 0$. So

$$\begin{aligned}
&\sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t) \left(F\delta - \sum_{f:(m,n)\in\mathbf{R}_f} \epsilon X_f\right) \\
\leq &\sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t) \left(F\delta - \epsilon \min_f X_f\right).
\end{aligned}$$

If we further choose δ such that $\delta \leq \frac{\epsilon \min_f X_f}{2F}$, we have

$$\begin{aligned}
&\sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t) \left(F\delta - \sum_{f:(m,n)\in\mathbf{R}_f} \epsilon X_f\right) \\
\leq &\sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t) \left(-\frac{1}{2}\epsilon \min_f X_f\right).
\end{aligned}$$

Now note that $\max_{(m,n)\in\mathcal{L}} Q_{(m,n)}(t) \geq \sqrt{\frac{V(t)}{L}}$. Suppose $M > 0$ is a positive constant, then when $V(t) > L\left(\frac{M_1+M_2+M_3+M}{\epsilon \min_f X_f}\right)^2$,

$$\max_{(m,n)\in\mathcal{L}} Q_{(m,n)}(t) > \frac{M_1 + M_2 + M_3 + M}{\epsilon \min_f X_f}.$$

So we have

$$\begin{aligned}
& V(t+1) - V(t) \\
\leq\ & M_1 + M_2 + M_3 \\
& + \sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t)\left(-\frac{1}{2}\epsilon \min_f X_f\right) \\
\leq\ & M_1 + M_2 + M_3 - \max_{(m,n)\in\mathcal{L}} Q_{(m,n)}(t)(\epsilon \min_f X_f) \\
\leq\ & -M
\end{aligned}$$

From the above analysis we can see, when $t > T(\delta)$, if $V(t) > L\left(\frac{M_1+M_2+M_3+M}{\epsilon \min_f X_f}\right)^2$, the drift of the Lyapunov function is negative. Moreover, when $t \leq T(\delta)$, $V(t) \leq L\,(FX^{\max}T(\delta))^2$. Therefore, there exists a constant C_V such that $V(t) \leq C_V$ for all t. So $Q_{(m,n)}(t) \leq \sqrt{C_V} \triangleq C$ for any $(m,n) \in \mathcal{F}$ and t.

□

Next, based on Lemma 15, we will prove that all per-flow queues are bounded as well.

Lemma 16. *Given a set of traffic flows* $\{X_f\}$ *such that* $\{(1+\epsilon)X_f\}$ *is supportable for some* $\epsilon > 0$, *under the self-regulated MaxWeight scheduling, there exists a constant* $\tilde{C}$ *such that, the lengths of the per-flow queues are no more than* $\tilde{C}$ *for all* t.

Proof. First let's focus on the source node S_f of flow f. Suppose n_f^2 is the second hop on the route of flow f. For node S_f we have the rate estimate

$$\tilde{X}_f^{S_f}(t) = (1 + \min\left\{\gamma, \frac{1}{Q_{(S_f,n_f^2)}(t)}\right\})\frac{A_f(t)}{t}.$$

From property (5.1) we know, $\frac{A_f(t)}{t} \to X_f$ as $t \to \infty$. Furthermore, according to Lemma 15, $Q_{(S_f,n_f^2)}(t) < C$ for all t. Combining these two facts, we can always find t_s such that when $t > t_s$, $\tilde{X}_f^{S_f}(t) > (1 + \frac{1}{2C})X_f$.

We define a "super time slot" for node S_f, which consists of M_s time slots, where M_s is a positive number. We index the super time slot using T. Suppose $a_f^{S_f}(T)$ is the arrival packets of flow f at node S_f during super time slot T, then due to (5.1), $\frac{a_f^{S_f}(T)}{M_s} \to X_f$ when $M_s \to \infty$.

In other words, for a constant η_s such that $0 < \eta_s < \frac{X_f}{2C}$, we can always find an M_s such that $\frac{a_f^{S_f}(T)}{M_s} < X_f + \eta_s$. In other words,

$$a_f^{S_f}(T) < X_f M_s + \eta_s M_s.$$

On the other hand, for $M_s T > t_s$, during super time slot $T+1$, $\tilde{X}_f^{S_f}(t) > (1+\frac{1}{2C})X_f$. In other words, the aggregated service rate of $Q_f^{S_f}$ during $T+1$ is at least $(1+\frac{1}{2C})M_s X_f$. Since $\eta_s < \frac{X_f}{2C}$, we have

$$a_f^{S_f}(T) < X_f M_s + \eta_s M_s < (1+\frac{1}{2C})M_s X_f.$$

From above analysis we can see, the packets arriving at node S_f during super time slot T can be completely served during super time slot $T+1$, if $M_s T > t_s$. Since the arrival packets to node S_f before time t_s is at most $X^{\max} t_s$, it is easy to see that there exists a constant C_s such that $Q_f^{S_f}(t) < C_s$ for all t.

Next we use induction to prove all per-flow queues are bounded by a constant. Denote by n_f^i the i^{th} node on the route of flow f. Now assume that

$$Q_f^{n_f^j}(t) \le C_i \text{ for all } t \text{ and } j \le i. \text{ (induction assumption)}$$

We next define

$$C_\delta = i(C_i + C).$$

First consider the departures of the per-flow queue $Q_f^{n_f^{i+1}}$ at node n_f^{i+1}. It is easy to see that

$$\sum_{\tau=1}^{t} a_f^{S_f}(\tau) \ge \sum_{\tau=1}^{t} a_f^{n_f^{i+1}}(\tau) \ge \sum_{\tau=1}^{t} a_f^{S_f}(\tau) - C_\delta$$

because C_δ is an upper bound on the number of packets belonging to flow f and queued at nodes n_f^1 to node n_f^i (the up-streaming nodes of node $i+1$). So according to (5.1) we have,

$$\frac{\sum_{\tau=1}^{t} a_f^{n_f^{i+1}}(\tau)}{t} \to X_f$$

as $t \to \infty$. Moreover, based on Lemma 15, $Q_{(n_f^{i+1}, n_f^{i+2})}(t) < C$ for any t. Therefore, there exists some t_i, such that when $t > t_i$,

$$\tilde{X}_f^{n_f^{i+1}}(t) > (1+\frac{1}{2C})X_f.$$

Then let's see the arrivals of the per-flow queue $Q_f^{n_f^{i+1}}$ at node n_f^{i+1}. We define a "super time slot" for node n_f^{i+1}, each super time slot consists of M_i time slots, where M_i is a positive number. We index the super time slots using T. According to (5.1), for any $\eta_i > 0$, there exists some large enough M_i such that the total arrival packets at source node S_f during super time slot T satisfies

$$a_f^{S_f}(T) \leq M_i(X_f + \eta_i).$$

Thus the total arrival packets at node n_f^{i+1} during super time slot T satisfies

$$a_f^{n_f^{i+1}}(T) \leq M_i(X_f + \eta_i) + C_\delta.$$

Now we can consider the dynamic of $Q_f^{n_f^{i+1}}$. We select large enough M_i and small enough $\eta_i > 0$ such that $M_i(\frac{1}{2C}X_f - \eta_i) > C_\delta$. For the arrival part,

$$a_f^{n_f^{i+1}}(T) \leq M_i(X_f + \eta_i) + C_\delta.$$

On the other hand, for $M_i T > t_i$, during super time slot $T+1$, the rate estimate

$$\tilde{X}_f^{n_f^{i+1}}(t) > (1 + \frac{1}{2C})X_f.$$

In other words, the aggregated service rate of $Q_f^{n_f^{i+1}}$ during $T+1$ is at least $(1+\frac{1}{2C})M_i X_f$. So we have

$$a_f^{n_f^{i+1}}(T) \leq M_i(X_f + \eta_i) + C_\delta < (1 + \frac{1}{2C})M_i X_f.$$

From above analysis we know, the available service rate for the per-flow queue $Q_f^{n_f^{i+1}}$ during super time slot $T+1$ is always greater than the arrivals at super time slot T if $M_i T > t_i$. In other words, under the proposed scheme, the arrival packets at super time slot T can always be completely served by the end of super time slot $T+1$. Since the arrival packets to $Q_f^{n_f^{i+1}}$ before time t_i is at most $X^{\max}t_i$, there exists a C_{i+1} value such that $Q_f^{n_f^{i+1}}(t) \leq C_{i+1}$ for all t. Then the lemma follows from the induction principle.

□

Based on Lemma 15 and Lemma 16, we directly have the following theorem.

Theorem 17. *Given a set of flows with arrival rates $\{X_f\}$ such that $\{(1+\epsilon)X_f\}$ is supportable for some $\epsilon > 0$, all queues under the self-regulated MaxWeight algorithm are bounded.*

5.4 Simulations

In this section, we use simulations to see the performance of the self-regulated MaxWeight algorithm, compared to the well-known back pressure algorithm.

5.4.1 Simulation Settings

The network we are simulating is shown in Figure 5.3. Basically we are simulating a grid network with 8×8 nodes and some random long links. All links are bi-directional links, and each link has capacity 1, i.e., in each time slot each link can transmit at most one packet at each direction. We further assume that all links can be activated simultaneously. All nodes in the network are full-duplex. In other words, they can transmit and receive packets concurrently. The nodes are indexed regularly, as can be seen on the figure.

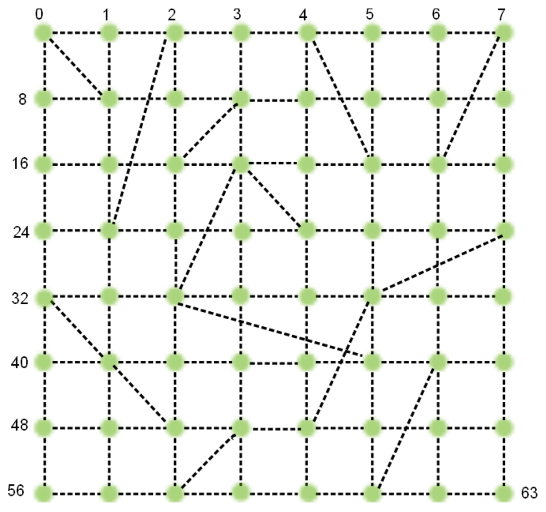

Figure 5.3 Network topology

There are 8 flows in the network, and each of the flows is associated with a fixed route. The flow routes are shown in Table 5.1. From the table we can see, there are no two flows sharing a single link. In this case the stability region of the network is $X_f < 1$ for all f. In the simulation, we set all X_f's to be identical, varying from 0.1 to 0.9 to represent different traffic loads. For each simulation scenario, we run it for $100,000$ time slots. Note that in the

Table 5.1 The flows in the network

Flow ID	Route
0	$8 \to 9 \to 10 \to 11 \to 12 \to 13 \to 14 \to 6$
1	$17 \to 18 \to 19 \to 28 \to 29 \to 30$
2	$38 \to 37 \to 36 \to 35 \to 34 \to 33 \to 32$
3	$46 \to 45 \to 53 \to 52 \to 60 \to 59 \to 58$
4	$17 \to 16 \to 24 \to 32 \to 41$
5	$19 \to 18 \to 26 \to 34 \to 42 \to 50 \to 49$
6	$43 \to 44 \to 36 \to 28 \to 20 \to 21 \to 22$
7	$55 \to 47 \to 39 \to 31 \to 23 \to 22$

self-regulated MaxWeight algorithm, γ is set to be 10.

We look at two performance metrics, which are the total queue length in the network, and the average end-to-end delay. Also, since [18] points out that under the back pressure algorithm, the sum of the queue lengths along a route increases quadratically as the route length, which leads to a poor delay performance, it would be interesting to see this phenomenon.

5.4.2 The Case of Constant Arrivals

First we are considering the case of constant arrivals. In other words, in each simulation scenario, $X_f(t) = \lambda$ for all f and t, where λ is the arrival rate of flows (a constant). Note that under constant arrival rate, the traffic property (5.1) is automatically satisfied.

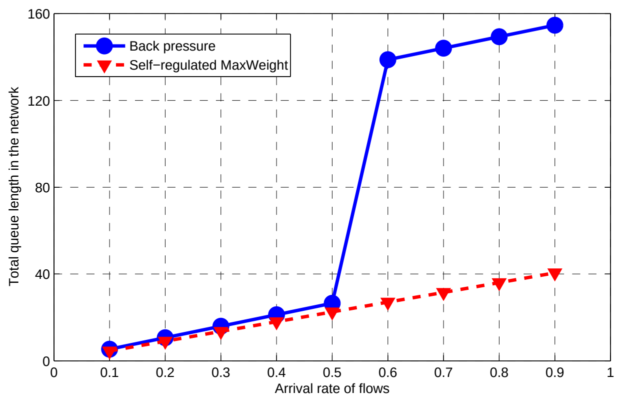

Figure 5.4 Total queue lengths in the network under constant arrivals

The total queue lengths in the network are shown in Figure 5.4. As it can be seen from the graph, when the traffic loads are low, i.e., $\lambda \leq 0.5$, the total queue lengths under both algorithms are similarly low. But under high traffic loads, the back pressure algorithm performs much worse than the self-regulated MaxWeight algorithm. The reason why the back pressure has a good performance in low traffic regime is that, when $\lambda \leq 0.5$, each source node at most generates a packet in two consecutive slots in a deterministic pattern due to constant arrival. Suppose a source generates a packet at slot 1, then it will be immediately transmitted to the second hop on the route since the second hop has queue length 0. In slot 2, the packet is transmitted to the third hop. So when the source generates another packet in slot 3, it will be transmitted to the second hop immediately because the queue length at the second hop has already return 0. So on and so forth. Therefore the packet transmission under the back pressure algorithm when $\lambda < 0.5$ is highly efficient which results in a good performance. However, under high traffic loads above packet transmission pattern does not hold so the performance of the back pressure algorithm becomes poor.

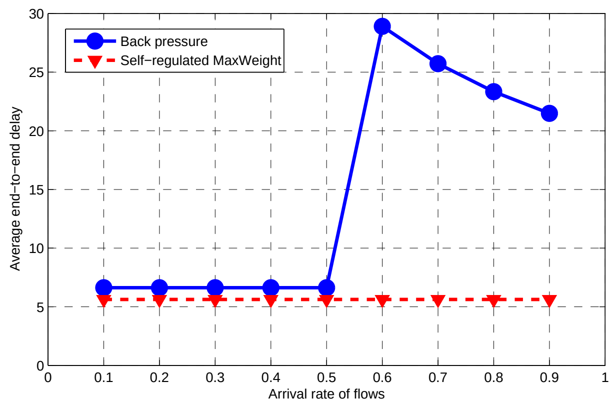

Figure 5.5 Average end-to-end delay under constant arrivals

The situation of average end-to-end delay can be seen in Figure 5.5. Average end-to-end delay is defined to be the average time consumed to transmit a packet from the source to the destination. From the figure we can see, similar to the situation of total queue lengths, two algorithms have similar performance under low traffic loads, but in the regime of heavy traffic, the self-regulated MaxWeight algorithm outperforms the back pressure algorithm significantly. We notice that under heavy traffic, as λ increases the delay decreases for the back pressure algorithm. It is possible because due to the Little's Law, when both λ and total queue lengths increase, whether the delay will increase or not depends on the rates at which λ and total queue lengths increase.

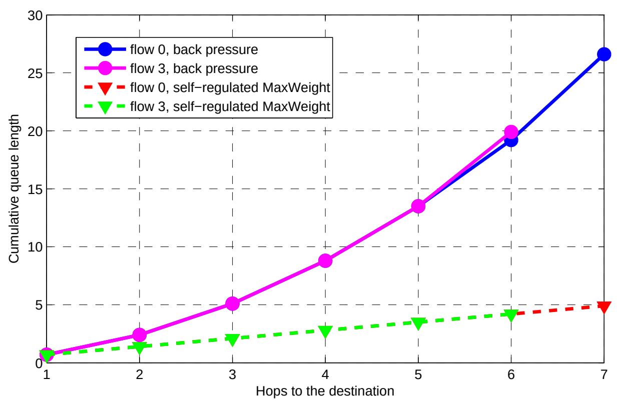

Figure 5.6 Cumulative queue lengths versus hops under constant arrivals

Figure 5.6 depicts the cumulative queue lengths versus hops to the destination. On the graph, X-axis denotes the segment of the route that is within n hops to the destination, and Y-axis is the cumulative queue length of the corresponding route segment. We randomly pick two flows which are flow 0 and flow 3 to see the situation. Basically, for the back pressure algorithm, the cumulative queue length increases quadratically as n increases, which is consistent with the findings in [18]. For the self-regulated MaxWeight algorithm, the cumulative queue length increases linearly as n. This is reasonable because we don't need to build up positive "pressure" to push the packets to the destination under the self-regulated MaxWeight algorithm. Whenever there is a packet in queue, it can be push to the next hop. This is the essential reason why our self-regulated MaxWeight algorithm outperforms the back pressure algorithm in terms of queue lengths and delay.

5.4.3 The Case of Light-tailed Stochastic Traffic

Next we consider the case of light-tailed stochastic traffic. We assume the number of arriving packets of each flow follows Poisson distribution with parameter λ, i.e., $\mathbf{E}[X_f(t)] = \lambda$ for all f.

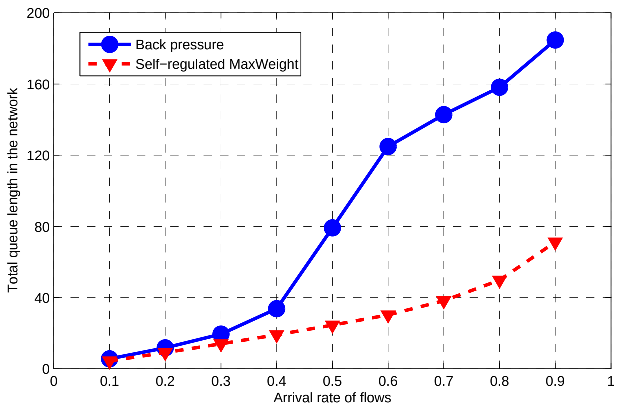

Figure 5.7 Total queue lengths in the network under Poisson arrivals

The situation of total queue lengths is shown in Figure 5.7. We can see from the curves that our proposed algorithm has a much better performance than the well-known back pressure algorithm in terms of total queue lengths, especially in heavy traffic regime. The reason is that for our algorithm, the network does not need to build up positive queue difference to transmit packets down to the destination, so the packets queued in the network are less than the case of back pressure algorithm

Furthermore, Figure 5.8 shows that the self-regulated MaxWeight algorithm results in much smaller end-to-end delay compared to the back pressure algorithm. This is a natural result following directly from the situation of total queue lengths and the Little's Law.

Similar to the case of constant arrivals, Figure 5.9 illustrates that the queue length accumulates quadratically from the destination to the source for the back pressure algorithm, while for the self-regulated MaxWeight algorithm, it accumulates only linearly. This phenomenon directly leads to different performances of the two algorithms in total queued packets in the network.

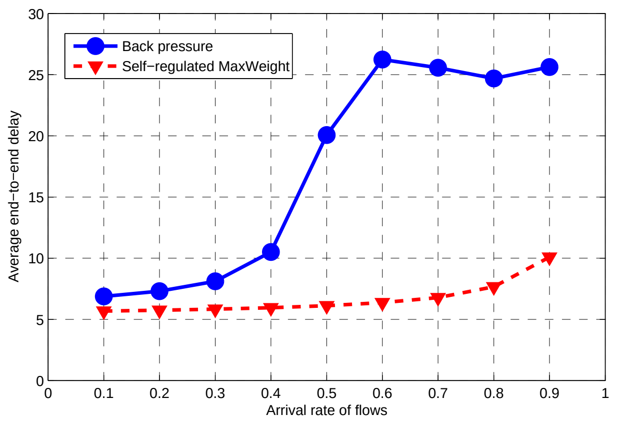

Figure 5.8 Average end-to-end delay under Poisson arrivals

5.4.4 The Case of Heavy-tailed Stochastic Traffic

This subsection we look at the scenario where the traffic is heavy-tailed stochastic traffic. In this case there are sometimes burst arrivals in the network. We want to see whether they will have bad impacts on the system-wide performance or not. We assume the number of arrival packets follows Pareto distribution with scale parameter $x_m = 0.05$ and shape parameter α. Note that here the arrival rate $\lambda = \frac{\alpha x_m}{\alpha - 1}$. So given a $\lambda > 0.1$, we should have $\alpha = \frac{\lambda}{\lambda - x_m}$.

The situations of total queued packets, average end-to-end delay and cumulative queue lengths along routes are illustrated in Figure 5.10, 5.11 and 5.12 respectively. From them we can see, even when the packet arrival processes are heavy-tailed, we can still see similar phenomena as in the case of light-tailed traffic. In other words, the existence of burst arrivals does not have bad effect on our proposed algorithm, and it still has much better performances than the back pressure algorithm.

5.5 Conclusion

In this chapter, we considered the scheduling problem in multihop wireless networks,and proposed the self-regulated MaxWeight scheduling that does not require the exchange of queue-length information among neighboring nodes, hence completely eliminates the communication

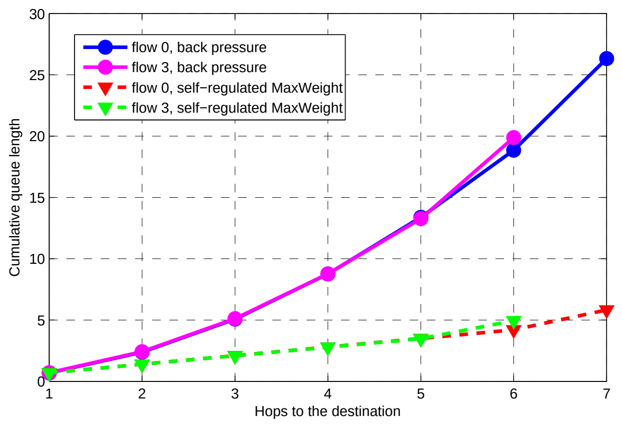

Figure 5.9 Cumulative queue lengths versus hops under Poisson arrivals

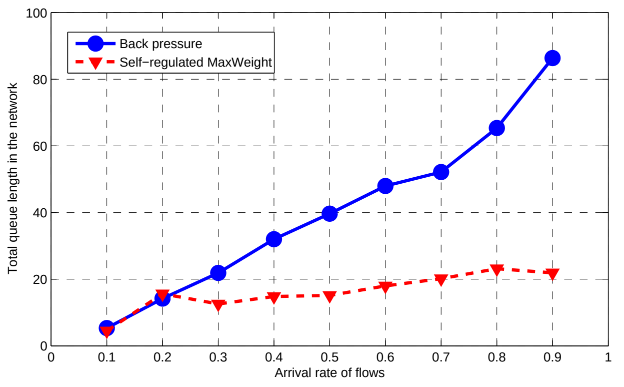

Figure 5.10 Total queue lengths in the network under Pareto arrivals

overhead when combined with recent CSMA-based scheduling algorithms. The new algorithm is proved to be throughput optimal and has a much better performance compared to the well-known back pressure algorithm.

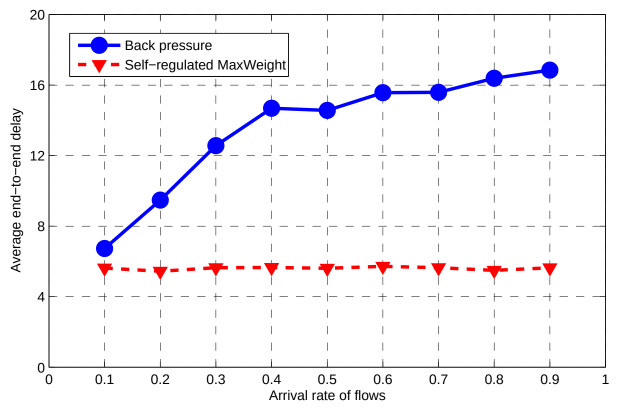

Figure 5.11 Average end-to-end delay under Pareto arrivals

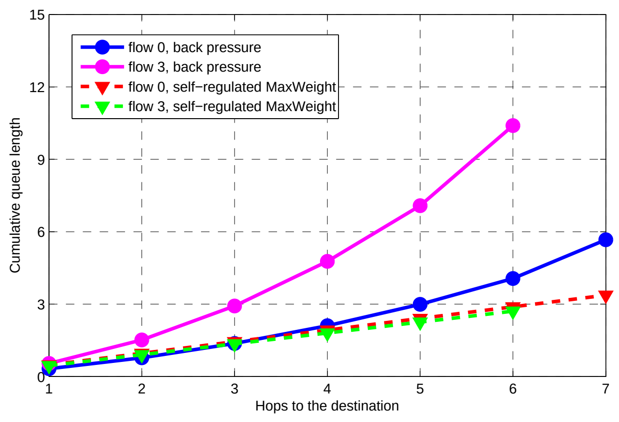

Figure 5.12 Cumulative queue lengths versus hops under Pareto arrivals

CHAPTER 6. Summary and Acknowledgement

In my Ph.D. research, I focused on the resource allocation problem in wireless networks in the presence of flow-level dynamics. I first investigated the scheduling problem in wireless cellular networks, including single-channel and multi-channel networks. Then, I investigated the joint congestion control and scheduling problem in wireless peer-to-peer networks and proposed an optimal architecture which can maximize the social welfare while satisfying the delay constraints of packets. At last, in Chapter 5, I described my work on the scheduling problem in multihop wireless networks. The scheduling algorithm developed is proved to be optimal and has superior performance than previous algorithms.

Acknowledgement: My Ph.D. works are joint works with Prof. Lei Ying, Prof. R. Srikant, Prof. Eylem Ekici and Dr. J. J. Jaramillo. I hereby thank them for their insightful guidance.

BIBLIOGRAPHY

[1] P. van de Ven, S. Borst, and S. Shneer, "Instability of MaxWeight Scheduling Algorithms," in *Proc. IEEE Infocom.*, Rio de Janeiro, Brazil, April 2009.

[2] B. Sadiq and G. de Veciana, "Throughput Optimality of Delay-driven Maxweight Scheduler for a Wireless System with Flow Dynamics," in *Proc. Ann. Allerton Conf. Communication, Control and Computing*, 2009.

[3] L. Tassiulas and A. Ephremides, "Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks", in *IEEE Transactions on Automatic Control*, Vol. 37, No. 12, pp. 1936-1949, December 1992.

[4] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, R. Vijayakumar, and P. Whiting, "Scheduling in a Queueing System with Aynchronously Varying Service Rates," *Probability in the Engineering and Informational Sciences*, vol. 18, pp. 191–217, 2004.

[5] A. Eryilmaz, R. Srikant, and J. R. Perkins, "Stable Scheduling Policies for Fading Wireless Channels," in *IEEE/ACM Trans. Network.*, vol. 13, no. 2, pp. 411–424, 2005.

[6] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic Power Allocation and Routing for Time Varying Wireless Networks," in *IEEE Journal on Selected Areas in Communications, Special Issue on Wireless Ad-Hoc Networks*, vol. 23, no. 1, pp. 89-103, Jan. 2005.

[7] X. Liu, E. Chong, and N. Shroff, "Opportunistic Transmission Scheduling with Resource-Sharing Constraints in Wireless Networks," in *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 10, pp. 2053-2064, Oct, 2001.

[8] P. Viswanath, D. Tse and R. Laroia, “Opportunistic Beamforming Using Dumb Antennas,” in *IEEE Transactions on Information Theory*, Vol. 48, No. 6, pp. 1277-1294, June 2002.

[9] X. Lin, N. B. Shroff and R. Srikant, “A Tutorial on Cross-Layer Optimization in Wireless Networks,” in *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, August 2006.

[10] L. Georgiadis, M. Neely and L. Tassiulas, *Resource Allocation and Cross Layer Control in Wireless Networks*, NoW publishers, 2006.

[11] S. Shakkottai and R. Srikant, *Network Optimization and Control.* NoW publishers, 2007.

[12] S. Asmussen. *Applied Probability and the Theory of Queues.* Springer, 2003.

[13] S. Meyn and R. L. Tweedie, *Markov chains and stochastic stability.* Cambridge University Press, 2009.

[14] L. Tassiulas, and A. Ephremides, “Dynamic Server Allocation to Parallel Queues with Randomly Varying Connectivity,” in *IEEE Transaction on Information Theory*, vol. 39, pp. 466-478, March 1993.

[15] Rajiv Laroia, “Future of Wireless? The Proximate Internet,” 2010. `http://www.cedt.iisc.ernet.in/people/kuri/Comsnets/Keynotes/Keynote-Rajiv-Laroia.pdf`

[16] S. Liu, L. Ying and R. Srikant, “Throughput-Optimal Opportunistic Scheduling in the Presence of Flow-Level Dynamics,” in *IEEE/ACM Transactions on Networking*, Vol. 19, No. 4, pp 1057-1070, August, 2011.

[17] S. Liu, L. Ying, and R. Srikant, “Scheduling in Multichannel Wireless Networks with Flow-level Dynamics,” in *Proc. Ann. ACM Sigmetrics Conf*, New York City, NY, 2010.

[18] L. Bui, R. Srikant, and A. L. Stolyar, “Novel Architecture and Algorithms for Delay Reduction in Back-pressure Scheduling and Routing”, in *Proc. IEEE INFOCOM Mini Conference*, 2009.

[19] L. Jiang and J. Walrand, “Distributed CSMA Algorithm for Throughput and Utility Maximization in Wireless Networks”, in *Proc. Allerton Conference*, Monticello, Illinois, 2008.

[20] J. Ni and R. Srikant, “Distributed CSMA/CA Algorithms for Achieving Maximum Throughput in Wireless Networks”, in *Information Theory and Applications Workshop*, 2009.

[21] S. Rajagopalan, D. Shah, and J. Shin, “Network Adiabatic Theorem: An Efficient Randomized Protocol for Contention Resolution”, in *Proc. ACM SIGMETRICS*, 2009.

[22] C. Humes, “A Regulator Stabilization Technique: Kumar-seidman Revisited”, in *IEEE Transactions on Automatic Control*, vol. 39, no. 1, pp. 191C196, Janunary 1994.

[23] X. Wu and R. Srikant, “Regulated Maximal Matching: A Distributed Scheduling Algorithm for Multi-hop Wireless Networks with Nodeexclusive Spectrum Sharing”, in *Proc. Conference on Decision and Control*, 2005.

[24] J. J. Jaramillo, R. Srikant and L. Ying, “Scheduling for Optimal Rate Allocation in Ad Hoc Networks With Heterogeneous Delay Constraints”, in *IEEE Journal on Selected Areas in Communications*, Vol. 29, No. 5, May, 2011.

[25] D. G. Luenberger, “Linear and Nonlinear Programming”, 2nd ed. Norwell, MA: Kluwer Academic Publishers, 2003.

[26] S. Boyd and L. Vandenberghe, “Convex Optimization”, 1st ed. New York, NY: Cambridge University Press, Mar. 2004.

[27] J. J. Jaramillo and R. Srikant, “Optimal Scheduling for Fair Resource Allocation in Ad Hoc Networks with Elastic and Inelastic Traffic”, in *Proc. IEEE INFOCOM*, San Diego, CA, USA, 2010.

[28] M. J. Neely, E. Modiano, and C.-P. Li, “Fairness and Optimal Stochastic Control for Heterogeneous Networks”, in *Proc. IEEE INFOCOM*, Miami, FL, 2005.

[29] A. Stolyar, “Maximizing Queueing Network Utility Subject to Stability: Greedy Primal-dual Algorithm”, in *Queueing Systems*, vol. 50, no. 4, pp. 401 – 457, Aug. 2005.

[30] A. Eryilmaz and R. Srikant, “Fair Resource Allocation in Wireless Networks Using Queue-length-based Scheduling and Congestion Control”, in *Proc. IEEE INFOCOM*, Miami, FL, 2005.