SUPPORTING DELAY GUARANTEES OVER UNRELIABLE WIRELESS CHANNELS

BY

I-HONG HOU

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Doctoral Committee:

Professor P.R. Kumar, Chair
Professor Tarek Abdelzaher
Professor Klara Nahrstedt
Professor R. Srikant
Professor Jean Walrand, University of California, Berkeley

Abstract

Many emerging applications of networks require delay guarantees for packet deliveries. It is particularly challenging to provide services for these applications over wireless channels, since wireless transmissions are usually unreliable. In this dissertation, we provide a theory that formulates and addresses the problem of serving flows with delay guarantees over unreliable wireless channels.

The core of this theory is an analytical model that jointly considers several practical aspects of flows with delay guarantees: traffic patterns, per-packet delay bounds, throughput requirements, and channel reliabilities. The model can also address fading channels and the usage of rate adaptation. Based on this model, we obtain solutions for three important mechanisms: admission control, packet scheduling, and utility maximization. In addition, we address the scenario of broadcasting flows with delay constraints and incorporate various network coding mechanisms. We also extend models used in the real-time system literature and discuss the scheduling problem for a multimedia server.

Table of Contents

Chapter 1

Introduction

There are increasing demands for using wireless networks to serve applications that require delay guarantees. Such applications include VoIP, video streaming, real-time surveillance, networked control, etc. One common characteristic of these applications is that they have a strict deadline associated with each packet. Each packet needs to be delivered before its deadline, or it expires and is no longer useful for its application. On the other hand, while each such applications may tolerate a small portion of its packets missing their deadlines, it still requires a specified *timely throughput*, which is defined as the throughput of packets that are delivered before their deadlines, in order to maintain its performance.

Serving such applications is especially challenging in wireless networks. Wireless transmissions are subject to shadowing, fading, and interference from other transmissions. Thus, wireless transmissions are usually unreliable. Further, the channel reliabilities of different clients can be different, and can even vary over time.

There are of course several alternative ways in which to model a wireless network consisting of flows that have delay constraints. The difficulty has been that these formulations have by and large led to intractable analytical problems. The end result has been that essentially no significant progress has been made in this area. In this thesis, we provide a useful and tractable framework for the modeling, analysis and design of real-time wireless communications. This formulation is also generalizable in several directions to handle various additional features, while still providing tractable solutions, and, in some cases, somewhat surprising answers. This framework is built on top of an analytical model that jointly considers the three important aforementioned challenges: a strict deadline for each packet, the timely throughput requirement specified by each client or application, and finally the unreliable and heterogeneous nature of wireless transmissions. An important feature is that this model is suitable for characterizing the needs of a wide range of applications, and allows each application to specify its individual demand. Thus the contracts that result from this framework are on the one hand supportable by protocols, and on the other usable by application designers.

Using this framework, we provide several important solutions for serving applications with both delay guarantees and timely throughput requirements. We provide a simple and sharp characterization of when

all the demands of the clients in a system are feasible under the limitations of their channel reliabilities, and obtain a polynomial-time algorithm for admission control. Further, we address the problem of packet scheduling. We propose an on-line scheduling policy that is feasibility optimal in the sense that it fulfills the demands of any set of clients as long as the demands of the set of clients are feasible.

This framework can be further generalized in various directions of interest. We extend the framework to address scenarios where the timely throughput requirements of clients are elastic. This can be formulated as a utility maximization problem. A bidding game, in which both clients and servers follow simple on-line strategies, then achieves the maximum total utility in the system.

The framework can also incorporate the usage of rate adaptation, where different clients use different transmission rates in order to guarantee error-free transmissions. Both the problems of packet scheduling and utility maximization can be addressed when rate adaptation is used. Further, we also study the behaviors of selfish and strategic clients, and design a truthful auction, under which strategic clients gain nothing by hiding their private preferences. We show that these results can also be applied to other wireless applications, including dynamic spectrum allocation and mobile wireless cellular networks.

We also generalize this framework to broadcasting flows that require delay guarantees. We show that the resulting model can also accommodate the optional usage of various network coding mechanisms. We then propose scheduling policies for several different network coding mechanisms.

Finally, we study the scenario of a multimedia server where packets from the same flow are of differing importances, such as packets of MPEG-4 video streaming. We extend the models of imprecise computation and increasing reward with increasing service (IRIS)in the real-time system literature. We sharply characterize when it is feasible to fulfill the demands of clients, and derive a linear time admission control algorithm. We also propose a feasibility optimal scheduling policy, as well as an on-line policy that is proved to achieve an approximation bound.

The rest of the thesis is organized as follows. Chapter 2 describes the framework and model. Chapter 3 studies the problem of admission control and scheduling for a simplified base case. Chapter 4 introduces a polynomial-time algorithm for the base case, and discusses the admission control problem for a more generalized case. Chapter 5 proposes a general condition for designing feasibility optimal scheduling policies for different scenarios. The usage of this condition is demonstrated by deriving policies for two particular cases. Chapter 6 considers scenarios where the timely throughput requirements of clients are elastic and addresses the utility maximization problem that arises. Chapter 7 solves the utility maximization problem for systems where rate adaptation is employed, and proposes an auction mechanism to prevent selfish clients from gaining additional benefits by lying about their utility functions. Chapter 8 considers the problem of broadcasting flows with delay constraints and the optional usage of network coding. Chapter 9 studies

the scheduling problem for a multimedia server where packets from the same flow may be of differing importance.

Chapter 2

System Model and Problem Formulation

2.1 System Model

We start by describing a model that can incorporate traffic patterns, delay bounds, and delivery ratio requirements for clients, as well as time-varying channels both with and without rate adaptation. The model can address both uplink and downlink traffic.

Consider a wireless system with N clients, $\{1, 2, \ldots, N\}$, and one access point (AP). Time is slotted with slots denoted by $t \in \{0, 1, 2, \ldots\}$. Time slots are further grouped into *intervals* $[kT, (k+1)T)$, with interval length T. Packets for each client are generated at the beginning of each interval, at time slots $\{0, T, 2T, \ldots\}$, probabilistically, with no more than one packet per client. We model the packet generations as a stationary, irreducible Markov process with finite state. The average probability that packets are generated for subset S of clients is $R(S)$. Packet generations can be dependent between clients, and packet generations in an interval can depend on other intervals. Packets can be for either downlink or uplink.

Each client n specifies a delay requirement τ_n measured in slots, with $\tau_n \leq T$. If the packet for client n is not delivered by the ${\tau_n}^{th}$ time slot of the interval, the packet expires and is discarded. The AP is in charge of scheduling all transmissions. When a downlink packet is scheduled, the AP first sends the packet to the designated client. The client needs to reply with an ACK if it receives this packet. The packet is considered delivered if the AP receives the ACK. On the other hand, if an uplink packet is scheduled, the AP first sends a poll message, such as the CF-POLL packet in IEEE 802.11, to indicate the scheduled client. That client sends the uplink packet to the AP upon receiving the poll packet. In this case, the packet is considered delivered if the AP receives the uplink packet. Note that the AP does not have to transmit an ACK. Thus, in either case of uplink or downlink, the AP obtains the information whether a transmission is successful. This scheme applies naturally to a wide range of server-centric wireless communication technologies, such as IEEE 802.11 Point Coordination Function (PCF), WiMax, and Bluetooth.

Next we turn to the model of the channel. We consider an unreliable, heterogeneous, and time-varying channel model. We model the channel condition as a stationary, irreducible Markov process with a finite

set of channel states $\mathcal{C}$. The average probability that channel state c occurs is f_c, with the channel state remaining constant within each interval. We consider the system both with rate adaptation and without. When rate adaptation is not available, that is, when all packets are transmitted at a fixed rate, the AP can make exactly one transmission in each time slot. Under channel state c, the link reliability between the AP and client n is $p_{c,n}$, by which we mean that a packet transmitted by the AP for client n is delivered with probability $p_{c,n}$. On the other hand, when the system uses rate adaptation, the channel states describe the maximal rates that can be supported between the AP and clients, which in turn decide the service times for transmissions. Under channel state c, it takes $s_{c,n}$ time slots to make an error free transmission to client n.

The channel state and the packet arrivals in an interval are assumed to be independent of each other. We also assume that the AP has knowledge of channel state.

The performance of a client is measured in terms of its *timely throughput*, defined as the long-term average number of packets that is delivered for a client per interval. Each client n requires a timely throughput of at least $q_n > 0$ packets per interval. Since, on average, there are $\sum_{S:n\in S} R(S)$ packets for client n per interval, this timely throughput bound can also be interpreted as a delivery ratio requirement of $\frac{q_n}{\sum_{S:n\in S} R(S)}$.

Definition 1. *A set of clients with the above delay constraints and timely throughput is said to be* fulfilled *by a particular scheduling policy η if the timely throughput of each client n is at least q_n with probability 1. More formally,* $\liminf_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} 1(A \text{ packet of client } n \text{ is delivered successfully in interval } k) \geq q_n$*, with probability one, for each client $n = 1, 2, \cdots, N$, where $1(\cdot)$ is the indicator function of the event.*

2.2 Problem Formulation

Based on this model, we consider three fundamental problems for serving applications with delay bounds and timely throughput requirements: admission control, scheduling, and utility maximization.

Due to the limited wireless resource, it is not always possible to fulfill every set of clients. To ensure all the admitted clients are fulfilled, it is important to enforce admission control mechanisms, which essentially consists of verifying whether a set of clients is *feasible*:

Definition 2. *A set of clients is* feasible *if there exists some scheduling policy that fulfills it.*

In addition to evaluating feasibility, it is also important to design a *feasibility optimal* scheduling policy that actually fulfills a feasible set of clients.

Definition 3. *A scheduling policy is said to be* feasibility optimal *if it fulfills every feasible set of clients.* [1]

[1] This can be regarded as the analog of throughput optimality.

So far, we have assumed that the timely-throughput requirements, q_n, are specified by each client. However, in many scenarios, the timely throughput requirements are not specified, and the AP is in charge of assigning timely throughput levels to each client due to some criteria. One natural criteria is to assign timely throughput levels so as to maximize the total utility of the system. Suppose that each client has a certain utility function, $U_n(q_n)$. The problem of choosing q_n to maximize the total utility, under the feasibility constraint, can be described by the following optimization problem:

SYSTEM:

$$\text{Max } \sum_{i=1}^{N} U_i(q_i) \tag{2.1}$$

$$\text{s.t. Feasibility conditions} \tag{2.2}$$

$$\text{over } q_n \geq 0, \forall 1 \leq n \leq N. \tag{2.3}$$

Chapter 3

The Base Case: Static Channel and Homogeneous Delay Bounds

In this chapter, we study a simplified base case where rate adaptation is not applied, the channel is static, and all clients require the same delay bounds that are equal to the interval length. That is, $\mathcal{C} = 1$ and $\tau_n \equiv T$. Though the model is simplistic, studying this special case offers insights into other more complicated scenarios. In this special case, we will use p_n instead of $p_{c,n}$ since the channel is static.

3.1 Feasibility Condition for Static Channels

In this section, we derive a necessary condition for a set of clients to be feasible under the special case being discussed.

Intuitively, the more often the server attempts transmissions for a client, the higher the timely throughput that the client gets. More formally, we have the following lemma:

Lemma 1. *The long-term average timely throughput of a client n is at least q_n packets per interval if and only if the AP, on average, transmits packets for that client $w_n := w_n(q_n) = \frac{q_n}{p_n}$ times per interval.*

Proof. Define:

$$u_n(t) = \begin{cases} 1, & \text{if client } n \text{ makes a transmission at time } t, \\ 0, & \text{otherwise,} \end{cases}$$

and

$$d_n(t) = \begin{cases} 1, & \text{if client } n \text{ delivers a packet at time } t, \\ 0, & \text{otherwise.} \end{cases}$$

Let $\mathfrak{F}_t$ be the σ-algebra generated by $\{(u_n(k), d_n(k-1)), \text{for } 1 \leq k \leq t \text{ and } 1 \leq n \leq N\}$. (We set $d_n(0) = 0$ for all n.)

Then $E[d_n(t)|\mathfrak{F}_t] = p_n u_n(t)$. Hence, by the martingale stability theorem of Loeve [53],

$$\lim_{\mathfrak{T}\to\infty} \frac{1}{\mathfrak{T}} \sum_{t=1}^{\mathfrak{T}} [d_n(t) - p_n u_n(t)] = 0, \text{ a.s.} \tag{3.1}$$

Since the timely throughput of client n must be at least q_n,

$$\liminf_{\mathfrak{T}\to\infty} \frac{T}{\mathfrak{T}} \sum_{t=1}^{\mathfrak{T}} d_n(t) \geq q_n, \text{ a.s.}$$

Hence, $\liminf_{\mathfrak{T}\to\infty} \frac{T}{\mathfrak{T}} \sum_{t=1}^{\mathfrak{T}} u_n(t) \geq \frac{q_n}{p_n}$ from (3.1).

□

We will hereby refer to w_n as the *workload for client* n. Thus, a set of clients is fulfilled if and only if the average numbers of transmissions per interval for packets for each client is higher than its workload.

Since the delay bounds for all clients are T time slots and the AP can make at most one transmission in each time slot, the following necessary condition can be obtained:

Lemma 2. *A set of N clients is feasible only if $\sum_{n=1}^{N} w_n \leq T$.*

This necessary condition turns out, however, to be not sufficient. Since undelivered packets are discarded at the end of each interval, the AP can only transmit packets that are generated in the current interval. It is possible that, at some time slot of an interval, all packets are delivered and the AP is subsequently forced to stay idle. While the number of idle time slots depends on the scheduling policy, its probability distribution is the same for the following particular set of policies.

Definition 4. *A scheduling policy is said to be* work conserving *if the AP never idles whenever there is any undelivered packet.*

Lemma 3. *The probability distribution of the amount of idle time slots in an interval is the same for all work conserving scheduling policies.*

Proof. Let γ_n be the random variable denoting the number of transmissions the AP needs to make for a packet for client n before it is successfully delivered. γ_n has the geometric distribution with parameter p_n, that is, $Prob\{\gamma_n = t\} = p_n(1-p_n)^{t-1}$ for all positive integers t. Further, assume that a subset S of clients generates packets in an interval. Let $L_{S,\eta}$ be the random variable indicating the number of idle time slots in such an interval under scheduling policy η. We have:

$$L_{S,\eta} = \begin{cases} T - \sum_{n\in S} \gamma_n, & \text{if } \sum_{n\in S} \gamma_n < T, \\ 0, & \text{otherwise,} \end{cases}$$

for all work conserving policies. Thus, the probability distribution of $L_{S,\eta}$ is the same for all work conserving policies. □

We can therefore define $L_S := L_{S,\eta}$, where η is any work conserving policy.

The following observation shows that we can always construct a work conserving policy, from any policy, by modifying it so that it performs at least as well as the original policy.

Lemma 4. *Let η be a scheduling policy that fulfills some sets of clients. Then there exists a work conserving policy η' that fulfills the same set of clients.*

Proof. The policy η can be modified into a work conserving one by attempting any undelivered packet whenever η idles. This modification cannot reduce the number of deliveries for any client and thus would fulfill any set of clients that η fulfills. □

Based on this lemma, we can therefore limit our discussion to work conserving policies throughout the rest of this chapter. Suppose a subset S of clients generates packets in an interval. Then Lemma 3 implies that the expected number of idle time slots in that interval is $E[L_S]$. Since such an interval occurs with probability $R(S)$, the average number of idle time slots in an interval is $\sum_S R(S)E[L_S]$, and the AP can, on average, therefore make $T - \sum_S R(S)E[L_S]$ transmissions in an interval. This observation leads to the following refined necessary condition:

$$\sum_{n=1}^{N} w_n \leq T - \sum_{S} R(S)E[L_S]. \tag{3.2}$$

However, we can go even further by considering *all* subsets of the set of all clients $\{1, 2, \ldots, N\}$. For any subset $S' \subseteq \{1, 2, \ldots, N\}$, let

$$I_{S'} := \sum_{S} R(S)E[\max\{0, T - \sum_{n \in S \cap S'} \gamma_n\}] = \sum_{S} R(S)E[L_{S \cap S'}].$$

This is the average number of time slots spent idling in an interval, if S' were the set of all clients. Clearly, if a set of clients is feasible, all subsets of it are also feasible. Hence, we can further refine the necessary condition (3.2):

Lemma 5. *A set of clients is feasible only if $\sum_{n \in S} w_n \leq T - I_S$ holds for every subset S.*

Surprisingly, we will show that the necessary condition stated in Lemma 5 is indeed sufficient in Section 3.3.

3.2 Scheduling Policies

In this section, we propose two scheduling policies for flows with delay bounds. Both policies are what we call *largest debt first policies*. The idea of a largest debt first policy is to compute the *debt* owed to each client at the beginning of every interval. The AP then determines the priorities of clients according to their debts, a client with larger debt getting a higher priority. Ties are broken by lexicographic order. In each time slot of the interval, the client with the highest priority among those who have an undelivered packet is scheduled to transmit. The only difference between the two policies lies in the definitions of debts.

The first policy, which we call the *largest time-based debt first* policy, tries to make every client get a share of time at least as large as its implied work load. To see how much a client lags behind its implied work load, we define debt as follows:

Definition 5. *The* time-based debt *of client n at time t is defined as $\frac{t}{T} \times w_n$ minus the actual number of time slots that client n has transmitted by time slot t. The policy which assigns priorities according to it is the* largest time-based debt first policy.

The next policy, which we call the *largest weighted-delivery debt first* policy, approaches the timely throughput requirements more directly. It seeks to make every client have a timely throughput higher than its requirement, that is, q_n.

Definition 6. *Let $c_n(t)$ be the number of successful transmissions of client n up to time t. The* weighted-delivery debt *of client n at time t is defined as $(\frac{t}{T} \times q_n - c_n(t))/p_n$. The policy which assigns priorities accordingly is called the* largest weighted-delivery debt first policy.

3.3 Proofs of Optimality

In this section, we prove that both policies above are feasibility optimal for fairly general traffic patterns. Our proof is based on Blackwell's approachability theorem [8]. We first describe the content of this theorem.

Consider a single player game with a *vector* payoff in each round determined by a probability distribution on the Euclidean N-dimensional space which depends on the action taken by the player in that round. Suppose, under some policy, the player takes action $a(i)$ and gets a payoff $v(i)$, which is an N-dimensional vector, in each round i. Blackwell studied the long-term average payoff the player gets, that is, $\lim_{j\to\infty} \sum_{i=1}^{j} v(a(i))/j$, and introduced the concept of *approachability*:

Definition 7. *Let $A \subseteq \mathbb{R}^N$ be any set in the N-dimensional space. Consider a policy η, which incurs payoffs:*

$$v(a(1)), v(a(2)), \ldots.$$

Let δ_j be the distance between the point $\sum_{i=1}^{j} v(a(i))/j$ and A. We say A is approachable *under policy η, if for every $\varepsilon > 0$ there is a j_0 such that,*

$$Prob\{\delta_j \geq \varepsilon \text{ for some } j \geq j_0\} \leq \varepsilon.$$

Blackwell derived a sufficient condition for approachability:

Theorem 1. *Let $A \subseteq \mathbb{R}^N$ be any closed set in N-dimensional space. Let η be a policy whose action depends solely on the average payoff to date, $x_j = \sum_{i=1}^{j-1} v(a(i))/j$. Thus, we can express $a(j)$ as $a'(x_j)$. Then A is approachable under η if the following policy is used:*

If $x_j \notin A$, let y be the closest point in A to x_j, and let H be the hyperplane passing through y and perpendicular to the line segment $x_j y$.

Then A is approachable under η if H separates x_j and the expected payoff of round j, that is, $E[v(a'(x_j))]$.

Utilizing this fundamental theorem, we prove that both largest debt first policies are feasibility optimal. Since a feasible set of clients must satisfy the necessary condition in Lemma 5, we only need to prove that the two policies fulfill every set of clients that satisfy the necessary condition.

Theorem 2. *The largest time-based debt first policy is feasibility optimal.*

Proof. We first translate the model into a single player game. A round in the game corresponds to an interval containing T time slots in the model. The player is the AP. The action the player can take is to choose the priorities of clients for that interval, with the interpretation that an undelivered packet for a client is transmitted in an interval only after all packets from clients with higher priorities are delivered in that interval. The payoff the player gets is the net change of the time-based debt owed to each client, which is thus an N-dimensional vector. To be more precise, the payoff the player gets in a round is $v = [v_1, v_2, \ldots, v_N]$, where v_n equals w_n minus the number of times the AP transmits the packet for client n during that interval.

By Lemma 1, the demand of a client n is met if the AP transmits its packets at least $w_n = q_n/p_n$ times per interval on average, or equivalently, the client approaches a non-positive time-based debt. Thus, to establish the optimality of the largest time-based debt first policy, we only need to show that the set $A := \{z = [z_1, z_2, \ldots, z_N] | z_n \leq 0, \forall n\}$ is approachable under this policy.

Suppose that at the beginning of some interval, the average payoff is $x = [x_1, x_2, \ldots, x_N]$. If $x \in A$, no

action violates approachability by Theorem 1. If $x \notin A$, at least one of $x_1, x_2, \ldots, x_N$ is strictly positive, and we can reorder the clients so that $x_1 \geq x_2 \geq \cdots \geq x_m > 0 \geq x_{m+1} \cdots \geq x_N$. The closest point in A to x is

$$y = [0, 0, \ldots, 0, x_{m+1}, x_{m+2}, \ldots, x_N].$$

The hyperplane passing through y and perpendicular to the line segment xy is $H := \{z | h(z) := \sum_{n=1}^{m} x_n z_n = 0\}$.

Let $\bar{x}$ be the payoff of this interval according to the largest time-based debt first policy. Also, let $\bar{w}_n$ be the number of times the AP transmits the packet from client n in the interval. We can express $\bar{x}$ as $\bar{x} = [w_1 - \bar{w}_1, w_2 - \bar{w}_2, \ldots, w_N - \bar{w}_N]$.

Since $h(x) = \sum_{n=1}^{m} x_n^2 > 0$, in order to show H separates x and $E[\bar{x}]$, it suffices to show that $h(\bar{x}) \leq 0$. We have:

$$\begin{aligned} h(\bar{x}) &= \sum_{n=1}^{m} x_n (w_n - \bar{w}_n) \\ &= \sum_{n=1}^{m-1} [(x_n - x_{n+1})(\sum_{k=1}^{n} w_k - \sum_{k=1}^{n} \bar{w}_k)] + x_m (\sum_{k=1}^{m} w_k - \sum_{k=1}^{m} \bar{w}_k). \end{aligned}$$

Next we evaluate the value of $\sum_{k=1}^{n} \bar{w}_k$ for each n. First assume that a subset S of clients generate packets at the beginning of an interval. By the largest time-based debt first policy, the server will give priority according to the ordering $1, 2, \ldots, N$. Hence, $\sum_{k=1}^{n} \bar{w}_k$ is the number of transmissions the AP makes if there are only packets for a subset $S_n = \{1, 2, \ldots, n\} \cap S$ of clients.

In other words, $\sum_{k=1}^{n} \bar{w}_k = T - L_{S_n}$, where L_{S_n} is the random variable indicating the number of time slots that remain idle in an interval when only packets from clients in the subset S_n are present. Thus we have $E[\sum_{k=1}^{n} \bar{w}_k | S] = T - E[L_{S_n}]$. Taking the expected value over S yields:

$$\begin{aligned} E[\sum_{k=1}^{n} \bar{w}_k] &= E[E[\sum_{k=1}^{n} \bar{w}_k | S]] = \sum_{S} R(S) E[\sum_{k=1}^{n} \bar{w}_k | S] \\ &= T - \sum_{S} R(S) E[L_{S \cap \{1,2\ldots,n\}}] = T - I_{\{1,2,\ldots,n\}}. \end{aligned}$$

Now, according to the necessary condition stated in Lemma 5, we have $\sum_{k=1}^{n} w_k \leq T - I_{\{1,2,\ldots,n\}} = \sum_{k=1}^{n} \bar{w}_k$, for all n. Further, $x_1 \geq x_2 \geq \cdots \geq x_m > 0$. Thus, $E[h(\bar{x}) \leq 0]$, and A is approachable under the largest time-based debt first policy by Theorem 1, which also implies that the largest time-based debt first policy is feasibility optimal. □

Theorem 3. *The largest weighted-delivery debt first policy is also feasibility optimal.*

Proof. As in the previous proof, we need to translate this policy into one for the single player game. Again, a round in the game corresponds to an interval consisting of T time slots in our model. The action a player, which is the AP, can take, is to decide the priorities of clients for that interval. However, in this case, the payoff the player gets is the net change of the weighted-delivery debt. In other words, the payoff is an N-dimensional vector $v = [v_1, v_2, \ldots, v_N]$, where $v_n = (q_n - 1)/p_n$ if the AP delivers a packet for client n in the interval, or $v_n = q_n/p_n$ if not. The timely throughput of a client n is at least q_n packets per interval if it approaches a non-positive weighted-delivery debt. Thus, we can prove that the largest weighted-delivery debt is optimal by showing that the set $A := \{z = [z_1, z_2, \ldots, z_N] | z_n \leq 0, \forall n\}$ is approachable.

Let $x = [x_1, x_2, \ldots, x_N]$ be the average payoff at the beginning of an interval. Again, we only need to evaluate the performance of the largest weighted-delivery debt first policy under the case $x \notin A$. We can reorder the clients so that $x_1 \geq x_2 \geq \cdots \geq x_m > 0 \geq x_{m+1} \geq \cdots \geq x_N$. The closest point in A to x is $y = [0, 0, \ldots, 0, x_{m+1}, x_{m+2}, \ldots, x_N]$. The hyperplane passing through y and perpendicular to the line segment xy is $H := \{z | h(z) := \sum_{n=1}^{m} x_n z_n = 0\}$.

Let π_n be the indicator function that the AP delivers a packet from client n, which is a random variable. The payoff of this interval is $\bar{x} = [(q_1 - \pi_1)/p_1, (q_2 - \pi_2)/p_2, \ldots, (q_N - \pi_N)/p_N]$.

By Theorem 1, the set A is approachable if H separates x and $E[\bar{x}]$. Since $h(x) = \sum_{n=1}^{m} x_n^2 > 0$, we only need to show $E[h(\bar{x})] \leq 0$ to complete the proof. We have:

$$\begin{aligned}
h(\bar{x}) &= \sum_{n=1}^{m} x_n \frac{q_n - \pi_n}{p_n} \\
&= \sum_{n=1}^{m-1} [(x_n - x_{n+1})(\sum_{k=1}^{n} \frac{q_k}{p_k} - \sum_{k=1}^{n} \frac{\pi_k}{p_k})] + x_m (\sum_{k=1}^{m} \frac{q_k}{p_k} - \sum_{k=1}^{m} \frac{\pi_k}{p_k}) \\
&= \sum_{n=1}^{m-1} [(x_n - x_{n+1})(\sum_{k=1}^{n} w_k - \sum_{k=1}^{n} \frac{\pi_k}{p_k})] + x_m (\sum_{k=1}^{m} w_k - \sum_{k=1}^{m} \frac{\pi_k}{p_k}) \text{ (since } w_k = \tfrac{q_k}{p_k}).
\end{aligned}$$

Since $x_1 \geq x_2 \geq \cdots \geq x_m > 0$, it suffices to show $\sum_{k=1}^{n} w_k \leq E[\sum_{k=1}^{n} \frac{\pi_k}{p_k}]$, for every n. Recall that the necessary condition stated in Lemma 5 requires $\sum_{k=1}^{n} w_k \leq T - I_{\{1,2,\ldots,N\}}$ for every n, to be feasible. Thus, we only need to show $E[\sum_{k=1}^{n} \frac{\pi_k}{p_k}] = T - I_{\{1,2,\ldots,N\}}$ to establish optimality. Further, we have $E[\sum_{k=1}^{n} \frac{\pi_k}{p_k}] = \sum_S R(S) E[\sum_{k=1}^{n} \frac{\pi_k}{p_k} | S]$ and $I_{\{1,2,\ldots,n\}} = \sum_S R(S) E[L_{S \cap \{1,2,\ldots,n\}}]$. The proof is hence completed by showing that $E[\sum_{k=1}^{n} \frac{\pi_k}{p_k} | S] = T - E[L_{S \cap \{1,2,\ldots,n\}}]$ for every S and n, which is done in Lemma 6 below. □

Lemma 6. *Under the priority order* $\{1, 2, \ldots, N\}$, $E[\sum_{k=1}^{n} \frac{\pi_k}{p_k} | S] = T - E[L_{S \cap \{1,2,\ldots,n\}}]$, *for* $n = 1, 2, \ldots, N$,

and all $S \subseteq \{1, 2, \ldots, N\}$.

Proof. Suppose client m doesn't generate a packet in the interval, that is, $m \notin S$. Then the server cannot make any transmissions for client m, and we have $E[\pi_m|S] = 0$. Also, since $m \notin S$, $m \notin (S \cap \{1, 2, \ldots, n\})$, and client m plays no role in deciding the value of $E[L_{S\cap\{1,2,\ldots,n\}}]$. Thus, we can delete every client that is not in S and reorder the remaining clients. Equivalently, we only need to prove $E[\sum_{k=1}^{n} \frac{\pi_k}{p_k}|S] = T - E[L_{\{1,2,\ldots,n\}}]$, for $S \supseteq \{1, 2, \ldots, n\}$.

We prove this by induction. First consider the case $n = 1$. Since client 1 has the highest priority, its packet is delivered unless the AP fails in all the T attempts. Thus,

$$E[\frac{\pi_1}{p_1}|S] = \frac{Prob\{\text{the job of client 1 is accomplished}\}}{p_1} = \frac{1 - (1 - p_1)^T}{p_1}.$$

On the other hand, we also have:

$$\begin{aligned} E[L_{\{1\}}] = & \sum_{t=1}^{T-1} Prob\{\text{the packet for client 1 is delivered in at most } T - t \text{ transmissions}\} \\ = & \sum_{t=1}^{T-1}(1 - (1 - p_1)^{T-t}) = T - \frac{1-(1-p_1)^T}{p_1}. \end{aligned} \tag{3.3}$$

This gives us $E[\frac{\pi_1}{p_1}|S] = T - E[L_{\{1\}}]$, and the lemma holds for the case $n = 1$.

Assume that $E[\sum_{k=1}^{n} \frac{\pi_k}{p_k}|S] = T - E[L_{\{1,2,\ldots,n\}}]$ holds for all $n \leq m$. Consider the case $n = m + 1$. Since the client $m + 1$ has the lowest priority among clients $\{1, 2, \ldots, m + 1\}$, its packet is transmitted only after all packets from client 1 through client m are delivered. Since there are $L_{\{1,2,\ldots,m\}}$ time slots left after the AP delivers packets from the first m clients, we have:

$$\begin{aligned} E[\pi_{m+1}|S, L_{\{1,2,\ldots,m\}} = \sigma] =& Prob\{\text{the job of client } m + 1 \text{ is accomplished in } T - \sigma \text{ attempts}\} \\ =& 1 - (1 - p_{m+1})^{T-\sigma}. \end{aligned}$$

On the other hand, since $L_{\{1,2,\ldots,m\}} - L_{\{1,2,\ldots,m+1\}}$ is the number of transmissions that the AP makes for a packet from client $m + 1$, we also have:

$$\begin{aligned} & E[L_{\{1,2,\ldots,m\}} - L_{\{1,2,\ldots,m+1\}}|L_{\{1,2,\ldots,m\}} = \sigma] \\ =& \sum_{t=\sigma+1}^{T} Prob\{\text{the server makes at least } t - \sigma \text{ transmissions for the packet for client } m + 1\} \\ =& \sum_{t=\sigma+1}^{T} (1 - p_{m+1})^{t-\sigma-1} = \frac{1 - (1 - p_{m+1})^{T-\sigma}}{p_{m+1}} \\ =& E[\frac{\pi_{m+1}}{p_{m+1}}|S, L_{\{1,2,\ldots,m\}} = \sigma], \end{aligned}$$

for all σ. Thus, $E[\frac{\pi_{m+1}}{p_{m+1}}|S] = E[L_{\{1,2,\ldots,m\}} - L_{\{1,2,\ldots,m+1\}}]$. Finally, we have:

$$\begin{aligned} E[\sum_{k=1}^{m+1} \frac{\pi_k}{p_k}|S] =& E[\sum_{k=1}^{m} \frac{\pi_k}{p_k}|S] + E[\frac{\pi_{m+1}}{p_{m+1}}|S] \\ =& T - E[L_{\{1,2,\ldots,m\}}] + E[L_{\{1,2,\ldots,m\}} - L_{\{1,2,\ldots,m+1\}}] \\ =& T - E[L_{\{1,2,\ldots,m+1\}}]. \end{aligned}$$

By induction, the lemma holds for all n. □

A final remark is that since both policies fulfill every set of clients that satisfy the necessary condition in Lemma 5, this condition is also sufficient for feasibility.

Theorem 4. *A set of clients is feasible if and only if*

$$\sum_{n \in S} w_n \leq T - I_S$$

holds for every subset S.

3.4 Simulation Results

In this section, we evaluate the performance of the largest time-based debt first policy and the largest weighted-delivery debt first policy. We also evaluate the naive approach of using the IEEE 802.11 DCF standard, the Enhanced DCF (EDCF), which is proposed in IEEE 802.11e to enhance QoS, and a random priority policy that assigns priorities randomly.

We conduct two sets of simulations, one with clients carrying VoIP traffic, and one with clients carrying video streaming traffic. The major difference between the two settings lies in their traffic patterns. Many VoIP codecs generate packets periodically. On the other hand, video streaming technology, such as MPEG, may generate traffic with variable-bit-rate (VBR). Thus, packets arrive at the AP probabilistically, with probability depending on the context of the current frame, and arrivals are independent among different clients. We first describe the details of the settings of these two applications.

For the VoIP traffic, we follow the standards of the ITU-T G.729.1 [37] and G.711 [36] codecs. Both codecs generate traffic periodically. G.729.1 generates traffic with bit rates 8 – 32 kbits/s, while G.711 generates traffic at a higher rate of 64 kbits/s. We assume the interval length, T, is 20 ms, and the payload size of a packet is 160 Bytes. The codecs generate one packet every several intervals; with the duration

Table 3.1: MPEG Traffic Pattern

Activity	Great	High	Regular
Data rate	501597	392237	366587
Arrival probability	1	0.8	0.75

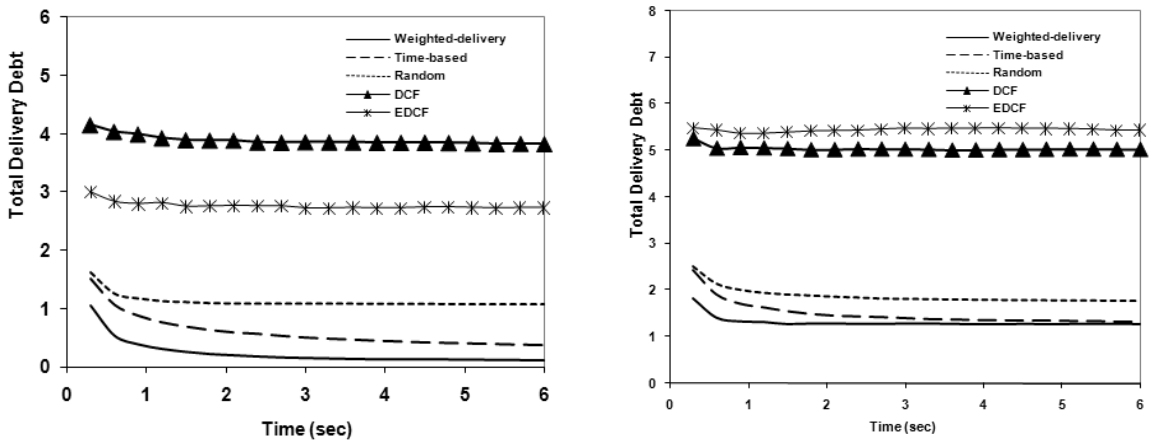
between packet arrivals depending on the bit rate used. We use IEEE 802.11b, which can provide a maximum transmission rate of 11 Mbits/s, as the underlying MAC.

We use MPEG for the video streaming setting. MPEG VBR traffic is usually modeled as a Markov chain consisting of three activity states [46] [20]. Each state generates traffic probabilistically at a different mean rate, with the state being determined by the current frame of the video. The statistical mean rates in each state are those obtained in an experimental study [20]. We use them in setting the traffic patterns of MPEG traffic. We assume the period length to be 6 ms and the payload size of a packet to be 1500 Bytes. Table 3.1 shows the statistical results of the experimental study [20], where we also present them in terms of the packet arrival probability of our setting. In Table 3.1, "Data rate" is measured in bits/GoP, where 1 GoP= 240 ms. Since the data rate is much higher for MPEG traffic, we use IEEE 802.11a, which can support up to 54 Mbits/s, as the MAC.

We measure the performance of each application for each application by the resulting *time-averaged total delivery debt*, defined as $\sum_{n=1}^{N}(q_n -$ actual timely throughput of client $n)^+$, where $x^+ := \max(x, 0)$.

For evaluating the VoIP traffic. we consider two groups of clients, group A and group B. Each client in group A generates packets periodically with period 60 ms, resulting in a 21.3 kbits/s flow, and requires 99% delivery ratio. Clients in group B also generate packets periodically but with period 40 ms, which corresponds to 32 kbits/s flows, and require 80% delivery ratio. The starting times of clients in each group are separated evenly. To be more specific, we can further divide the two groups into subgroups A_1, A_2, A_3, B_1, and B_2. Clients in subgroup A_i generate packets at the beginning of intervals $i, i+3, i+6, \ldots$, while clients in subgroup B_j generate packets at the beginning of intervals $j, j+2, j+4, \ldots$. The channel reliability of the n^{th} client in each subgroup is $(60+n)\%$. Evaluating the necessary and sufficient condition in Theorem 4 suggests that a set of 6 clients in each of the subgroups A_i and 5 clients in each B_j is feasible while a set of 6 clients in each A_i and 6 clients in each B_j is not.

Figure 3.1a shows the simulation results for the aforementioned feasible set of clients on the five tested policies, namely, the two largest debt first policies, the random policy, the DCF mechanism, and the EDCF mechanism. The total delivery debts of both largest debt first policies converge to 0 over time, showing that they fulfill this set of clients. However, the largest weighted-delivery debt first policy converges much faster than the largest time-based debt first policy. This is because the weighted-delivery debt reflects the actual



(a) Performance on a feasible set

(b) Performance on an infeasible set

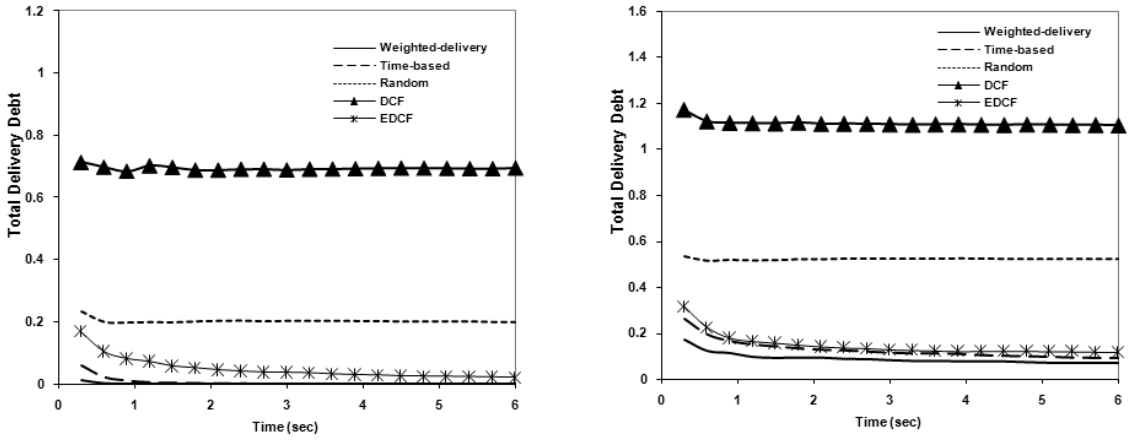
Figure 3.1: Total delivery debt for VoIP traffic

timely throughput a client has had, and thus uses a more direct and precise measure than the time-based debt. While the largest time-based debt first policy may be easier to implement, the largest weighted-delivery debt first policy should be preferred when tight performance is important. The other three policies all fail to fulfill this set of clients, indicating that they cannot be feasibility optimal.

To verify the correctness of the necessary and sufficient condition in Theorem 4, we also simulate the predicted infeasible set of clients composed of 6 clients in each subgroup A_i and 6 clients in each subgroup B_j. The results are shown in Figure 3.1b. All the five tested policies of course fail to fulfill this set of clients, since it is indeed infeasible. However, it should be noted that although the two largest debt first policies do not fulfill this set of clients, they still incur less total delivery debt than the other two policies. This result shows that the proposed policies perform well in comparison to some other policies even when dealing with an infeasible set of clients.

Next we evaluate MPEG traffic. We assume there are two groups of clients, A and B. Clients in group A require high quality video that generates packets according to Table 3.1, and a 90% delivery ratio for each packet, resulting in a requested timely throughput of 0.765 packet per interval. Clients in group B only require low quality video that generates packets only 50% as often as those in group A and demand a 80% delivery ratio, for a timely throughput requirement of 0.34 packet per interval. The channel reliability of the n^{th} client in each group is $(60+n)\%$. By evaluating the necessary and sufficient condition in Theorem 4, we predict that a set of 4 group A clients and 4 group B clients is feasible, while a set of 5 group A clients and 4 group B clients is not.

Figure 3.2a shows the simulation results on the feasible set of clients composed by 4 group A clients and 4 group B clients. Like in the case of VoIP traffic, the total delivery debts of both the two largest debt



(a) Performance on a feasible set

(b) Performance on an infeasible set

Figure 3.2: Total delivery debt for video streaming

policies converge to zero over time, and the two policies therefore fulfill this set of clients. Also, the largest weighted-delivery debt first policy converges faster than the largest time-based debt first policy. The random policy, EDCF, and DCF, on the other hand, fail to fulfill this set of clients.

Simulations on the infeasible set consisting of 5 group A clients and 4 group B clients are also shown in Figure 3.2b. All the five tested policies of course fail to fulfill this set of clients. Finally, the two largest debt first policies have the least total delivery debts among the five tested policies, showing that they offer good performance even for an infeasible set of clients.

Chapter 4

Admission Control

In Chapter 3, we have derived a necessary and sufficient condition for feasibility. Admission control therefore consists of evaluating this necessary and sufficient condition. However, the condition requires testing the inequalities $\sum_{n\in S} w_n \leq T - I_S$ for every subset S, which results in exponentially many tests and may not be computationally efficient. Also, the condition assumes that the channel state is static. In this chapter, we first propose a polynomial time algorithm for admission control under the case where the channel state is static and all clients generate packets at the beginning of each interval. We then discuss how to extend the necessary and sufficient condition to time-varying channels.

4.1 An Efficient Algorithm for Deterministic Packet Generations

In Chapter 3, we have developed a condition for feasibility stated in Theorem 4, which, however, entails evaluating inequalities for every subset of N clients, and thus results in exponentially many tests in N. In this section, we consider the special case where all clients generate one packet at the beginning of every interval. We first show that we only need to evaluate a total number of N conditions to determine feasibility. We then develop a polynomial-time algorithm for admission control.

Theorem 5. *Suppose all clients generate one packet at the beginning of each interval. Order the clients so that $q_1 \geq q_2 \geq \cdots \geq q_N$. Let S_k be the subset of clients $\{1, 2, \cdots, k\}$. The set of all clients is feasible if and only if $\sum_{n\in S_k} w_n \leq T - I_{S_k}$ for all k.*

Proof. It is obvious that the condition is necessary for feasibility. Hence, we only need to prove that it is also sufficient. We prove the converse statement: If a set of clients is not feasible, then for at least one k, $\sum_{n\in S_k} w_n > T - I_{S_k}$.

Define a *minimal infeasible subset* as a smallest subset S with $\sum_{n\in S} w_n + I_S > T$. Within every infeasible set there is always at least one such subset. Let m be the client with the largest index in S, i.e., the client latest in the order in S. We claim that $\sum_{n\in S_m} w_n + I_{S_m} > T$. If S is S_m itself, then we are done. Otherwise, pick a client m' latest in the order from $S_m \setminus S$ and let $S' := S \cup \{m'\}$. Let $D_w(S, m') := \sum_{n\in S\cup\{m'\}} w_n - \sum_{n\in S} w_n$

and $D_I(S, m') := I_{S\cup\{m'\}} - I_S$. Since $\sum_{n\in S} w_n + I_S > T$, we can show $\sum_{n\in S'} w_n + I_{S'} > T$ by establishing $D_w(S, m') + D_I(S, m') \geq 0$.

It is obvious that $D_w(S, m') = w_{m'}$. The expression for $D_I(S, m')$ is more complicated. Note that $-D_I(S, m')$ is the number of time slots that client m' will be transmitting if it is given the least priority when the set of clients is S'. Suppose it takes t time slots for every client in S to deliver its packet, then (3.3) shows the expected number of time slots client m' spends transmitting is $[1 - (1 - p_{m'})^{T-t}]/p_{m'}$. Hence, $-D_I(S, m') = \sum_{t=1}^{T} g_S(t)[1 - (1 - p_{m'})^{T-t}]/p_{m'}$, where $g_S(t)$ is the probability that all clients in S deliver their packets in exactly t time slots . Now we have

$$\begin{aligned} D_w(S, m') + D_I(S, m') &= w_{m'} - \sum_{t=1}^{T} g_S(t) \frac{1 - (1 - p_{m'})^{T-t}}{p_{m'}} \\ &> \frac{q_{m'}}{p_{m'}} - \sum_{t=1}^{T} g_S(t) \frac{1}{p_{m'}} \\ &= \frac{1}{p_{m'}} (q_{m'} - \sum_{t=1}^{T} g_S(t)). \end{aligned}$$

What remains is to determine the value of $\sum_{t=1}^{T} g_S(t)$. Since S is a minimal infeasible subset, the set $S \setminus \{m\}$ is feasible. Let η be any policy that fulfills $S \setminus \{m\}$. Consider the following policy η' for S: Whenever there is an undelivered packet for clients among $S \setminus \{m\}$, schedule clients as η does. Client m is scheduled to transmit only after packets for all other clients are delivered. Now, every client in $S \setminus \{m\}$ is fulfilled under this policy. However, since S is not feasible, at least one client in S is not fulfilled under any policy. Hence, client m is not fulfilled under policy η', and so we have $Prob(\text{client } m \text{ succeeds}) < q_m$. Further, since client m is given the least priority under policy η', we have $Prob(\text{client } m \text{ succeeds}) = Prob(\text{all packets in } S \text{ are delivered}) = \sum_t g_S(t)$. This implies $\sum_t g_S(t) < q_m$. Inserting this in the inequality in the previous paragraph yields

$$\begin{aligned} D_w(S, m') + D_I(S, m') &> \frac{1}{p_{m'}} (q_{m'} - \sum_t g_S(t)) \\ &> \frac{1}{p_{m'}} (q_{m'} - q_m) \geq 0, \end{aligned}$$

since client m occurs later in the order than client m', and we have sorted clients so that $q_1 \geq q_2 \geq \cdots \geq q_N$.

Now we have established $\sum_{n\in S'} w_n + I_{S'} > T$, where $S' = S \cup \{m'\}$. If $S' = S_m$, we are done. Otherwise, we can choose another client m'' with the highest order in $S_m \setminus S'$ and repeat the above argument to show $\sum_{n\in S'\cup\{m''\}} w_n + I_{S'\cup\{m''\}} > T$. By induction, $\sum_{n\in S_m} w_n + I_{S_m} > T$. □

In addition to reducing the number of needed tests to N, this theorem also helps improve the efficiency

of evaluating each test. In each test, we need to obtain the values of $\sum_{n\in S_m} w_n$ and I_{S_m}, both of which require more than a constant computation time. However, by using the fact that $S_m = S_{m-1} \cup \{m\}$, we can incrementally obtain these values and improve complexity.

Obtaining $\sum_{n\in S_m} w_n$ is easy since it equals $\sum_{n\in S_{m-1}} w_n + w_m$, and requires only one addition operation given the value of $\sum_{n\in S_{m-1}} w_n$. The computation of I_{S_m} is more complicated. As defined in the proof, let $g_{S_m}(t)$ be the probability that all packets in S_m are delivered at t, resulting in $T-t$ idle time slots. I_{S_m}, being the expected number of idle time slots in an interval, is $\sum_{i=1}^{T}(T-i)g_{S_m}(t)$. Consider the value of $g_{S_m}(t)$:

$$\begin{aligned} g_{S_m}(t) =& Prob(\text{all packets in } S_m \text{ are delivered at } t) \\ =& \sum_{i=1}^{T} Prob(\text{all packets in } S_{m-1} \text{ are delivered at time } t-i, \\ & \text{and clients } m \text{ takes } i \text{ time slots to succeed}) \\ =& \sum_{i=1}^{\tau} g_{S_{m-1}}(t-i)[p_m(1-p_m)^{i-1}]. \end{aligned}$$

Thus, the vector of $[g_{S_m}(i)]$ is indeed the convolution of the vectors of $[g_{S_{m-1}}(i)]$ and $[p_m(1-p_m)^{i-1}]$, which can be computed in $O(\tau^2)$ time by brute force, or $O(\tau \log \tau)$ by using the Fast Fourier Transform algorithm.

A complete algorithm for deciding whether a set of clients is feasible is given in Algorithm 1. The complexity of the algorithm is $O(N\tau^2)$ or $O(N\tau \log \tau)$, depending on the implementation of convolution.

Algorithm 1 IsFeasible

Assume clients are sorted so that $q_1 \geq q_2 \geq \cdots \geq q_N$
$totalW \leftarrow 0$
$[g_{S_0}(i)] \leftarrow [1, 0, \cdots, 0]$
for $m = 1$ to N **do**
  $totalW \leftarrow totalW + \frac{q_m}{p_m}$
  $[g_{S_m}(i)] \leftarrow [g_{S_{m-1}}(i)] * [p_m(1-p_m)^{i-1}]$
  $totalI \leftarrow \sum_{i=1}^{T}(T-i)g_{S_m}(i)$
  if $totalW + totalI > T$ **then**
    return Infeasible
return Feasible

4.2 Extension to Time-Varying Channels

We now discuss how to extend the aforementioned condition to time-varying channels. One intuitive approach is to decouple the channel states. The AP assigns a timely-throughput requirement $q_{c,n}$ for each channel state c and client n, with $\sum_{c\in\mathcal{C}} f_c q_{c,n} \geq q_n$. Also, for each channel state c, the assigned throughput requirements must be feasible under that channel state, that is, $\sum_{n\in S} \frac{q_{c,n}}{p_{c,n}} \leq T - E[I_{c,S}]$ for all

$S \subseteq \{1, 2, \ldots, N\}$, where $I_{c,S}$ is the expected number of time slots that the AP is forced to stay idle in an interval under channel state c for any work conserving scheduling policy, given that the AP only transmits packets for the subset S of clients. More formally, we therefore seek a matrix $Q = [q_{c,n}]$ that solves the following linear programming problem:

$$\begin{aligned} &\text{Max } \textstyle\sum_{n=1}^{N} \sum_{c \in \mathcal{C}} f_c q_{c,n} \\ \text{s.t. } &\textstyle\sum_{c \in \mathcal{C}} f_c q_{c,n} \geq q_n, \forall n \\ &\textstyle\sum_{n \in S} \frac{q_{c,n}}{p_{c,n}} \leq T - E[I_{c,S}], \forall c, \forall S \subseteq \{1, 2, \cdots, N\}. \end{aligned}$$

Theorem 6. *When all clients require the same delay bound and rate adaptation is not applied, a set of clients is feasible if and only if there exits a matrix Q that solves the above linear programming problem.*

Chapter 5

Scheduling Policies

We have proposed two largest debt first scheduling policies in Chapter 3. The two polices are feasibility optimal when rate adaptation is not applied, all clients require the same delay bound, and the channel state is static. In this chapter, we prove a sufficient condition for feasibility optimality without any assumptions on rate adaptation, delay bounds, and channel states. This condition describes a class of feasibility optimal scheduling policies. To demonstrate the utility of the class of policies, we study two particular scenarios of interest. The first scenario employs rate adaptation and treats time-varying channels, as well as allowing different delay bounds for different clients. The other scenario treats the case where rate adaptation is not available and considers time-varying channels. We derive computationally tractable scheduling policies and prove that they are feasibility optimal for both scenarios.

5.1 Preliminaries

We first revisit the definition of *fulfill*:

Definition 8. *A set of clients,* $\{1, 2, \ldots, N\}$ *is* fulfilled *under a scheduling policy* η*, if for every* $\epsilon > 0$*,*

$$Prob\{\frac{d_n(t)}{t/T} > q_n - \epsilon, \textit{ for every } n\} \to 1, \textit{ as } t \to \infty,$$

where $d_n(t)$ *is the number of packets delivered to client* n *up to time* t*.*[1]

Since the overall system can be viewed as a controlled Markov chain, we have:

Lemma 7. *For any set of clients that can be fulfilled, there exists a stationary randomized policy that fulfills the clients, which uses a probability distribution based only on the channel state, the set of undelivered packets, and the number of time slots remaining in the system (and not any events depending on past intervals), according to which it randomly chooses an undelivered packet to transmit, or stays idle.*

[1] In other words, we require $(\frac{d_n(t)}{t/T} - q_n)^+$ converges to zero in probability, for every n. In contrast, previous chapters require $(\frac{d_n(t)}{t/T} - q_n)^+$ converges to zero almost surely.

Since the computational overhead for some complex policies may be excessive and thus preclude supporting applications with delay bounds, we also consider the limited set of *priority-based policies*, which require computation only at the beginning of each interval:

Definition 9. *A* priority-based policy *is a scheduling policy which assigns priorities to some of the clients, based on past history and current state of the system, at the beginning of each interval. During the interval, a packet for a client is transmitted only after all packets for clients with higher priorities have been delivered. Packets for clients which do not receive a priority are never transmitted. A* stationary randomized priority-based policy *is one which chooses the priority order randomly according to a probability distribution that depends only on the channel state and packet arrivals at the beginning of each interval. We denote by* $\mathbb{P}$ *and* $\mathbb{P}_{rand}$ *the sets of priority-based policies and stationary randomized priority-based policies.*

The concept of feasibility can also be defined for the set of priority-based policies.

Definition 10. *A set of clients is* feasible in the set $\mathbb{P}$ *(or* $\mathbb{P}_{rand}$*) if there exists some scheduling policy in* $\mathbb{P}$ *(or* $\mathbb{P}_{rand}$*) that fulfills it.*

Similar to Lemma 7, if $[q_n]$ is feasible in the set $\mathbb{P}$, it is also feasible in the set $\mathbb{P}_{rand}$.

Definition 11. *We call the region in the* N*-space formed by vectors* $[q_n]$ *for which the clients are feasible in* $\mathbb{P}$ *(or all policies), as the* feasible region under $\mathbb{P}$ *(or all policies).*

Lemma 8. *The feasible regions under the class of all policies, or* $\mathbb{P}$*, are both convex sets.*

Proof. Let $[q_n]$ and $[q'_n]$ be two vectors in the feasible region under $\mathbb{P}$, and thus also feasible in $\mathbb{P}_{rand}$. Let η and η' be policies in $\mathbb{P}_{rand}$ that fulfill the two vectors, respectively. Then, the policy in $\mathbb{P}$ that randomly picks one of the two policies, with η being chosen with probability α, at the beginning of each interval, fulfills the vector $[\alpha q_n + (1-\alpha)q'_n]$. Further, since q_n and q'_n are both larger than 0 for each n, $\alpha q_n + (1-\alpha)q'_n > 0$ for all n. Thus, the vector $[\alpha q_n + (1-\alpha)q'_n]$ also falls in the feasible region under $\mathbb{P}$. A similar proof holds for the class of all policies. □

Note that if $[q_n]$ is feasible in $\mathbb{P}$, then so is $[q'_n]$, where $0 < q'_n \leq q_n$.

Definition 12. $[q_n]$ *is* strictly feasible in $\mathbb{P}$ *(or the class of all policies) if there exists some* $\alpha \in (0,1)$ *such that* $[q_n/\alpha]$ *is feasible in* $\mathbb{P}$ *(or the class of all policies).*[2]

Finally, we extend the concept of feasibility optimal for priority-based policies.

[2]Equivalently, $[q_n]$ is an interior point of the feasible region under $\mathbb{P}$ (or the class of all policies).

Definition 13. *A scheduling policy η is* feasibility optimal among $\mathbb{P}$ *(or the class of all policies) if it fulfills every set of clients that is strictly feasible in $\mathbb{P}$ (or the class of all policies).*

In the rest of the chapter, unless otherwise specified, the default is the set of all policies.

5.2 A Sufficient Condition for Feasibility Optimality

We now describe a more general class of policies that is feasibility optimal. We start by extending the concept of "debt."

Definition 14. *A variable $r_n(k)$, whose value is determined by the past history of the client n up to the k^{th} interval, or time slot kT, is called a* pseudo-debt *if:*

1. $r_n(0) = 0$*, for all n.*
2. *At the beginning of each interval, $r_n(k)$ increases by a constant strictly positive number $z_n = z_n(q_n)$, which is an increasing linear function of q_n.*
3. $r_n(k+1) = r_n(k) + z_n(q_n) - \mu_n(k)$*, where $\mu_n(k)$ is a non-negative and bounded random variable whose value is determined by the behavior of client n. Further, $\mu_n(k) = 0$ if the AP does not transmit any packet for client n.*
4. *The set of clients is fulfilled if and only if $Prob\{\frac{r_n(k)}{k} < \varepsilon\} \to 1$, as $k \to \infty$, for all n and all $\varepsilon > 0$.*

In the following example, we illustrate that both the time-based debt and the weighted-delivery debt are pseudo-debts under a static channel model.

Example 1. *At the beginning of each interval, the time-based debt $r_n^{(1)}(k)$ increases by $w_n = \frac{q_n}{p_n}$, and decreases by the number of time slots that the AP has transmitted packets for client n during the interval. Lemma 1 shows that condition (4) is satisfied.*

Similarly, $r_n^{(2)}(k)$, the weighted-delivery debt is also a special case. It increases by $\frac{q_n}{p_n}$ at the beginning of each interval, and decreases by $\frac{1}{p_n}$ if a packet is delivered for client n during that interval, and 0 otherwise. It satisfies condition (4) by definition. □

We can also define the *feasible region for debt in* $\mathbb{P}$ (or in the set of all policies) as the set of $[z_n]$ such that the corresponding $[q_n]$ is feasible in $\mathbb{P}$ (or in the set of all policies). Since z_n is a linear function of q_n and the feasible region for $[q_n]$ is a convex set (Lemma 8), the feasible region for $[z_n]$ is also a convex set.

Using the concept of pseudo-debt, we prove a sufficient condition for feasibility optimality. The proof resembles one used by Neely [56], though in a different context, and is based on:

Theorem 7 (Lyapunov Drift Theorem)**.** *Let $L(t)$ be a non-negative Lyapunov function. Suppose there exists some constant $B > 0$ and non-negative function $f(t)$ adapted to the past history of the system such that:*

$$E\{L(t+1) - L(t) | \textit{history up to time } t\} \leq B - \epsilon f(t),$$

for all t, then: $\limsup_{t\to\infty} \frac{1}{t}\sum_{i=0}^{t} E\{f(i)\} \leq B/\epsilon$. □

Theorem 8. *Let $r_n(k)$ be a pseudo-debt.*

1. *A policy that maximizes the* payoff function

$$\sum_{n=1}^{N} E\{r_n(k)^+ \mu_n(k) | c_k, S_k, [r_m(k)]\} \tag{5.1}$$

at the beginning of each interval is feasibility optimal, where c_k denotes the channel state in the k^{th} interval, and S_k is the subset of clients whose packets arrive at the AP at the beginning of the k^{th} interval.

2. *A priority-based policy that maximizes (5.1) over all policies in $\mathbb{P}$ is feasibility optimal in $\mathbb{P}$.*

Proof. We present the proof for $\mathbb{P}$ only. A similar proof works for the class of all policies too. Define $L(k) = \frac{1}{2}\sum_{n=1}^{N} r_n(k)^2$. Since $r_n(k+1) = r_n(k) + z_n - \mu_n(k)$,

$$\begin{aligned}\Delta(L(k)) :=& E\{L(k+1) - L(k) | [r_m(k)]\} = E\{\frac{1}{2}\sum_{n=1}^{N} r_n(k+1)^2 - \frac{1}{2}\sum_{n=1}^{N} r_n(k)^2 | [r_m(k)]\} \\ =& E\{\sum_{n=1}^{N} r_n(k)[z_n - \mu_n(k)] + \frac{1}{2}\sum_{n=1}^{N}[z_n - \mu_n(k)]^2 | [r_m(k)]\}.\end{aligned}$$

Define $B(k) := E\{\frac{1}{2}\sum_{n=1}^{N}[z_n - \mu_n(k)]^2 | [r_m(k)]\}$. Then $B(k) \leq B$, for all k, for some B. Hence for any policy in $\mathbb{P}$:

$$\Delta(L(k)) \leq E\{\sum_{n=1}^{N} r_n(k)[z_n - \mu_n(k)] | [r_m(k)]\} + B. \tag{5.2}$$

Suppose $[q_n]$ is strictly feasible in $\mathbb{P}$. The vector $[z_n]$ is thus an interior point of the feasible region (for debt) under $\mathbb{P}$, and there therefore exists some $\alpha \in (0,1)$ such that $[z_n/\alpha]$ is also in the feasible region under $\mathbb{P}$. Let $z_{min} = \min\{z_1, z_2, \ldots, z_N\}$. The N-dimensional vector $[z_{min}]$ whose elements are all z_{min}, falls in the feasible region under $\mathbb{P}$. Since the feasible region under $\mathbb{P}$ is a convex set, the vector $\alpha[z_n/\alpha] + (1-\alpha)[z_{min}] = [z_n + (1-\alpha)z_{min}]$ is also in the feasible region under $\mathbb{P}$.

By Lemma 7, there exists a stationary randomized policy η' in $\mathbb{P}$ that fulfills the set of clients with timely-throughput bounds for the vector $[z_n + (1-\alpha)z_{min}]$. Let $\mu'_n(k)$ be the decrease in the pseudo-debt for client

n under η' during the interval. Then, we have:

$$E\{\mu'_n(k)|[r_m(k)]\} = E\{E\{\mu'_n(k)|c_k, S_k, [r_m(k)]\}\}$$
$$\geq z_n + (1-\alpha)z_{min}.$$

Above, the outer expectation in the RHS is taken over channel states and the vectors of packet arrivals.

Let η be a policy that maximizes the payoff function (5.1), for all k, among all policies in $\mathbb{P}$. Then defining $\mu_n(k)$ and $r_n(k)$ as the decrease resulting from policy η and the pseudo-debt, we have:

$$\sum_{n=1}^{N} E\{r_n(k)^+ \mu_n(k)|c_k, S_k, [r_m(k)]\} \geq \sum_{n=1}^{N} E\{r_n(k)^+ \mu'_n(k)|c_k, S_k, [r_m(k)]\}.$$

We can assume without loss of generality that the policy does not work on any client n with $r_n(k) \leq 0$, that is, $\mu_n(k) = 0$ if $r_n(k) \leq 0$.[3] From (5.2), we obtain:

$$\begin{aligned}\Delta(L(k)) &\leq E\{\textstyle\sum_{n=1}^{N} r_n(k)^+ [z_n - \mu_n(k)]|[r_m(k)]\} + B \\ &\leq E\{\textstyle\sum_{n=1}^{N} r_n(k)^+ [z_n - \mu'_n(k)]|[r_m(k)]\} + B \\ &\leq -\textstyle\sum_{n=1}^{N} r_n(k)^+ (1-\alpha) z_{min} + B.\end{aligned}$$

Let $\epsilon := (1-\alpha)z_{min}$. By Theorem 7,

$$\limsup_{k\to\infty} \frac{1}{k} \sum_{i=0}^{k} E\{\textstyle\sum_{n=1}^{N} r_n(k)^+\} \leq B/\epsilon. \tag{5.3}$$

Finally, since z_n is a constant and $\mu_n(k)$ is a bounded function, $|r_n(k+1) - r_n(k)|$ is bounded, which implies that $|\sum_{n=1}^{N} r_n(k+1)^+ - \sum_{n=1}^{N} r_n(k)^+|$ is also bounded for all k. Thus, (5.3) implies that $\frac{1}{k} E\{\sum_{n=1}^{N} r_n(k)^+\} \to 0$ as $k \to \infty$, as shown in Lemma 9 below. This shows that $\frac{r_n(k)^+}{k}$ converges to 0 in probability for all n. Hence, η is feasibility optimal in $\mathbb{P}$. □

Lemma 9. *Let $f(t)$ be a non-negative function such that $|f(t+1) - f(t)| \leq M$, for some $M > 0$, for all t. If $\limsup_{t\to\infty} \frac{1}{t} \sum_{i=0}^{t} f(i) \leq B/\epsilon$, then $\lim_{t\to\infty} \frac{1}{t} f(t) = 0$.*

Proof. The proof is by contradiction. Suppose $\limsup_{t\to\infty} \frac{1}{t} f(t) > \delta$, for some $\delta > 0$. Thus, $f(t) > t\delta$ infinitely often. Suppose $f(t) > t\delta$ for some t. Since $|f(t) - f(t-1)| < M$, we have $f(t-1) > t\delta - M$. Similarly, $f(t-2) > t\delta - 2M$, $f(t-3) > t\delta - 3M, \ldots, f(t - \lfloor t\delta/M \rfloor) > t\delta - \lfloor t\delta/M \rfloor M \geq 0$. Summing over these terms gives: $\sum_{i=t-\lfloor t\delta/M \rfloor}^{t} f(i) > \frac{t\delta \lfloor t\delta/M \rfloor}{2}$, and thus, $\sum_{i=0}^{t} \frac{1}{t} f(i) > \frac{\delta \lfloor t\delta/M \rfloor}{2}$. Since $f(t) > t\delta$ infinitely

[3] Since a policy cannot lose its feasibility optimality by doing more work, this assumption is not restrictive.

often, $\limsup_{t\to\infty} \sum_{i=0}^{t} \frac{1}{t} f(i) = \infty$, which is a contradiction. □

Theorem 8 suggests a more general procedure to design feasibility optimal scheduling policies. To design a scheduling policy in a particular scenario, we need to choose an appropriate pseudo-debt and obtain a policy to maximize the payoff function. Maximizing the payoff function is, however, in general, difficult. Nevertheless, in some special cases, evaluating the payoff function gives us simple feasibility optimal policies or at least some insights into designing a reasonable heuristic, as long as we choose the correct pseudo-debt. In the following sections, we demonstrate the utility of this approach.

5.3 Scheduling Policy with Rate Adaptation

We now propose a feasibility optimal scheduling policy when rate adaptation is employed. Channel qualities can be time-varying and clients may have different deadlines.

To derive the scheduling policy, we define the *delivery debt* $r_n^{(3)}(k) := q_n k - d_n(kT)$, where $d_n(t)$ is the number of delivered packets for client n up to time slot t. Thus, $z_n := q_n$, while $\mu_n(k) = 1$ if a packet for client n is delivered in the interval, and $\mu_n(k) = 0$ otherwise.

Suppose at the beginning of interval k, the delivery debt vector is $[r_n^{(3)}(k)]$, the channel state is c, and the set of arrived packets is S. The transmission time for client n is $s_{c,n}$ time slots, and client n stipulates a delay bound of τ_n. Since transmissions are assumed to be error-free when rate adaptation is applied, the scheduling policy consists of finding an ordered subset $S' = \{m_1, m_2, \ldots, m_{N'}\}$ of S such that $\sum_{n=1}^{l} s_{c,n} \leq \tau_l$, for all $1 \leq l \leq m_{N'}$. That is, when clients are scheduled according to the ordering, no packets for clients in S' would miss their respective delay bounds. By Theorem 8, a policy using an ordered set S' that maximizes $\sum_{n\in S'} r_n^{(3)}(k)$ with the above constraint is feasibility optimal. This is a variation of the knapsack problem. When S' is selected, reordering clients in S' in an earliest-deadline-first fashion also allows all packets to meet their respective delay bounds. Based on this observation, we derive the feasibility optimal scheduling algorithm, the Modified Knapsack Algorithm. Let $M[n,t]$ be the maximum debt a policy can collect if only clients 1 through n can be scheduled and all transmissions need to complete before time slot t. Thus, $\max_{S'} \sum_{n\in S'} r_n^{(3)}(k) = M[N,T]$. Also, iteratively:

$$M[n,t] = \begin{cases} M[n,t-1] & \text{if } t > \tau_n, \\ \max\{M[n-1,t], r_n^{(3)}(k) + M[n-1,t-s_{c,n}]\} & \text{otherwise,} \end{cases}$$

where $M[n-1,t]$ is the maximum debt that can be collected when client n is not scheduled, and $r_n^{(3)}(k) + M[n-1,t-s_{c,n}]$ is that when client n is scheduled. The complexity of this algorithm is $O(N\tau)$, and it is thus

reasonably efficient.

Algorithm 2 Modified Knapsack Policy

for $n = 1$ to N **do**
  $r_n^{(3)}(k) = q_n k - d_n(kT)$
Sort clients such that $\tau_1 \leq \tau_2 \leq \cdots \leq \tau_N$
$S'[0,0] = \phi$
$M[0,0] = 0$
for $n = 1$ to N **do**
  for $t = 1$ to T **do**
    if $t > \tau_n$ **then**
      $M[n,t] = M[n,t-1]$
      $S'[n,t] = S'[n,t-1]$
    else if client n has a packet AND $r_n^{(3)}(k) + M[n-1, t-s_{c,n}] > M[n-1,t]$ **then**
      $M[n,t] = r_n^{(3)}(k) + M[n-1, t-s_{c,n}]$
      $S'[n,t] = S'[n-1, t-s_{c,n}] + \{n\}$
    else
      $M[n,t] = M[n-1,t]$
      $S'[n,t] = S'[n-1,t]$
schedule according to $S'[N,T]$

5.4 Scheduling Policy for Time-Varying Channels

We now consider the case when rate adaptation is not available, and propose a scheduling policy for time-varying channels and homogeneous delay bounds. We show that the policy is feasibility optimal among all priority-based policies. We use the delivery debt, $r_n^{(3)}(k)$, of Section 5.3.

Suppose at the beginning of an interval, the delivery debt vector is $[r_n^{(3)}(k)]$, the channel state is c, and the set of arrived packets is S. We wish to find the priority ordering that maximizes the payoff function $\mu_{tot}(k) = \sum_{n=1}^{N} r_n^{(3)}(k)^+ E\{\mu_n(k)\}$, where in the expectation we suppose that the channel state c and the set of arrival packets S are both fixed. Obviously, transmitting a packet from a client n with $r_n^{(3)}(k) \leq 0$ will not increase the value of $\mu_{tot}(k)$. Thus, we do not give priorities to clients with such non-positive delivery debts. For ease of the remaining discussion, in the sequel we assume $r_n^{(3)}(k) > 0$ for all n.

Consider two orderings, A and B: In A, the priority order is $\{1, 2, \ldots, N\}$, while, in B, the priority order is $\{1, 2, \ldots, m-1, m+1, m, m+2, m+3, \ldots, N\}$. Let the values of the payoff functions be μ_{tot}^A and μ_{tot}^B. Since clients 1 through $m-1$ have the same priorities in both orderings and their priorities are higher than the remaining clients, the values of $E\{\mu_n(k)\}$, $1 \leq n \leq m-1$ are the same for both orderings. On the other hand, clients $m+2$ through N also have the same priorities in both orderings and they can be scheduled only after the packets for clients 1 through $m+1$ are delivered. The probabilities of packet deliveries for these clients are the same under the two orderings. Thus, to compare the two orderings, one only needs to

evaluate the probabilities of packet delivery for client m and $m+1$. We further notice that the probabilities that packets for both clients m and $m+1$ are delivered are also the same for both orderings. With e_n denoting the event that the packet for client n is delivered,

$$\mu_{tot}^A - \mu_{tot}^B = r_m^{(3)}(k) Prob\{e_m \backslash e_{m+1} | \text{ordering A}\} - r_{m+1}^{(3)}(k) Prob\{e_{m+1} \backslash e_m | \text{ordering B}\}.$$

Suppose that there are τ' time slots left after all packets from client 1 through $m-1$ have been delivered. The probability distribution of τ' is the same under both orderings. Since the channel reliability is $p_{c,n}$,

$$\begin{aligned}
&\mu_{tot}^A - \mu_{tot}^B \\
=&r_m^{(3)}(k) E\{\textstyle\sum_{t=1}^{\tau'} p_{c,m}(1-p_{c,m})^{t-1}(1-p_{c,m+1})^{\tau'-t}\} \\
&-r_{m+1}^{(3)}(k) E\{\textstyle\sum_{t=1}^{\tau'} p_{c,m+1}(1-p_{c,m+1})^{t-1}(1-p_{c,m})^{\tau'-t}\} \\
=&[r_m^{(3)}(k) p_{c,m} - r_{m+1}^{(3)}(k) p_{c,m+1}] \times E\{\textstyle\sum_{t=0}^{\tau'-1}(1-p_{c,m})^{t}(1-p_{c,m+1})^{\tau'-t-1}\}.
\end{aligned}$$

Thus, $\mu_{tot}^A \geq \mu_{tot}^B$ if $r_m^{(3)}(k) p_{c,m} \geq r_{m+1}^{(3)}(k) p_{c,m+1}$. This leads us to obtain the Joint Debt-Channel Policy. Its computation time is only $O(N \log N)$.

Algorithm 3 Joint Debt-Channel Policy

1: **for** $n = 1$ to N **do**
2: $\quad r_n^{(3)}(k) = q_n k - d_n(kT)$, for all n
3: Sort clients with a packet arrival such that $r_1^{(3)}(k)p_{c,1} \geq r_2^{(3)}(k)p_{c,2} \geq \cdots \geq r_{N_0}^{(3)}(k)p_{c,N_0} > 0 \geq r_{N_0+1}^{(3)}(k)p_{c,N_0+1} \geq \cdots$
4: Transmit packets for clients 1 through N_0 by the ordering

Theorem 9. *The joint debt-channel policy is feasibility optimal among all priority-based policies.*

Proof. Let η be the joint debt-channel policy and η' any priority-based policy. Suppose the priorities assigned by the policies are $\eta_1, \eta_2, \ldots, \eta_m$, and $\eta'_1, \eta'_2, \ldots, \eta'_{m'}$. We modify η' as follows:

1. Delete any element in $\eta'_1 \sim \eta'_{m'}$ with $r_{\eta'_n}^{(3)}(k) \leq 0$.
2. For any client n with $r_n^{(3)}(k) > 0$ that is not in $\eta'_1 \sim \eta'_{m'}$, append it at the end of the ordering.
3. If $\eta'_1 \sim \eta'_{m'}$ is still different from $\eta_1 \sim \eta_m$, there exists some n such that $r_{\eta'_n}^{(3)}(k)p_{c,\eta'_n} < r_{\eta'_{n+1}}^{(3)}(k)p_{c,\eta'_{n+1}}$. Swap η'_n and η'_{n+1}.
4. Repeat Step 3 until the two orderings are the same.

Steps 1 and 2 will not decrease the value of the payoff function. As derived above, Step 3 does not decrease the value of the payoff function, either. Thus, η maximizes the payoff function and is feasibility optimal in $\mathbb{P}$. □

5.5 Simulation Results

We present simulation results in the section. Similar to Section 3.4, we consider two applications, VoIP and video streaming.

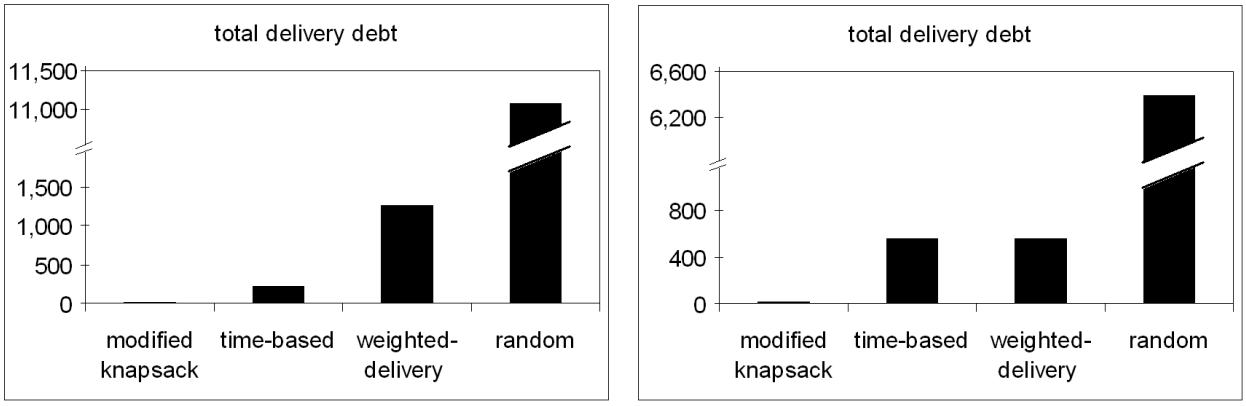
5.5.1 Rate Adaptation

We consider the scenario where rate adaptation is applied in this section. We also consider the time-varying channels and allow different clients to specify different delay bounds. We evaluate four policies, including the modified knapsack policy, the two largest debt first policies, and the random policy.

For VoIP traffic, we assume that the channel capacity of each client alternates between 11 Mb/s and 5.5 Mb/s. There are two groups of clients, A and B. Clients in group A generate one packet every three periods, or at rate 21.3 kbits/s, and require 90% of each of the clients' packets to be delivered, or a timely-throughput requirement of 19.2 kbits/s. Clients in group B generate one packet every two periods at rate 32 kbits/s, and require 70% of each of the clients' packets to be delivered, corresponding to a timely-throughput requirement of 22.4 kbits/s. The two groups can be further divided into subgroups, A_1, A_2, A_3, B_1, and B_2, each with 22 clients. Clients in subgroup A_i generate packets at periods $[i, i+3, i+6, \dots]$, and clients in subgroup B_i generate packets at periods $[i, i+2, i+4, \dots]$. Finally, clients in group A require a delay bound equal to the period length, while clients in group B require a delay bound equal to two-thirds of the period length.

Simulation results are shown in Figure 5.1a. The modified knapsack policy incurs the least total delivery debt among all evaluated policies.

Next we consider the scenario with MPEG traffic. We assume that channel capacity for each client alternates between 54 Mb/s and 24 Mb/s. We again assume there are two groups of clients. Clients in group A generate packets according to Table 3.1, and clients in group B are assumed to offer only lower quality video by generating packets only 80% as often as those in group A, in each of the three states. We assume clients in group A require 90% delivery ratios, and clients in group B require 60% delivery ratios. Since the length of a period for MPEG is very small, it is less meaningful to discuss heterogeneous delay bounds. Thus, we assume all clients require a delay bound equal to the length of a period. We further assume that there are 6 clients in both groups.



(a) VoIP traffic

(b) MPEG traffic

Figure 5.1: Performance for VoIP traffic with rate adaptation.

Simulation results are shown in Figure 5.1b. As in the case of VoIP traffic, the modified knapsack policy achieves the smallest total delivery debt among all the four policies.

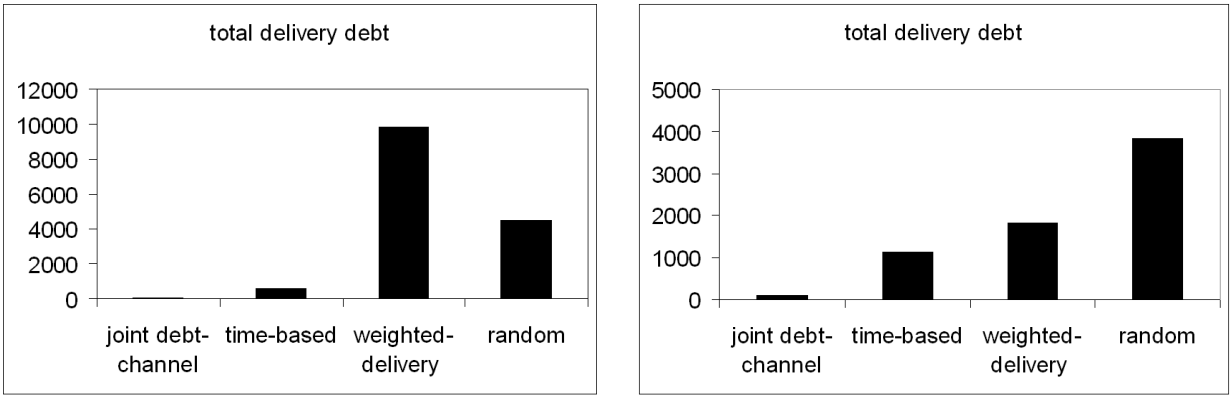
5.5.2 Time-varying Channels

We now consider the scenario with time-varying channels, with all clients requiring delay bounds equal to period length. We model the wireless channel by the widely used Gilbert-Elliot model [18] [27] [73], with the wireless channel considered as a two-state Markov chain, with "good" state and "bad" state. A simulation study by Bhagwat et al [6] shows that the link reliability can be modeled as 100% when the channel is in the good state, and 20% when the channel is in the bad state. The duration that the channel stays in one state is exponentially distributed with mean 1 – 10 sec for the good state, and 50 – 500 msec for the bad state.

For the case of VoIP traffic, we use a fixed transmission rate of 11 Mb/s. We consider the same two groups of clients as in the previous section. We assume that the mean duration of the bad state is 500 msec for all clients, and the mean duration of the good state is $1 + 0.5n$ sec for the n^{th} client in each subgroup. The time-average link reliability of the n^{th} client in each subgroup can be computed as $\frac{2.2+n}{3+n}$. There are 19 clients in each of the subgroups.

Simulations results are shown in Figure 5.2a. The joint debt-channel policy incurs near zero total delivery debt, while all the other policies have much larger total delivery debts.

For MPEG traffic, we assume there are two groups of clients, with the same traffic patterns and delivery ratio requirements as those in the previous section. We use 802.11a with a fixed data rate of 54 Mb/s as the underlying MAC. The mean duration when the channel is in the bad state is 500 msec for all clients, and the mean duration in the good state is assumed to be $1 + 0.5n$ sec for the n^{th} client in each group. There are 4 clients in both groups.



(a) VoIP traffic

(b) MPEG traffic

Figure 5.2: Performance under time-varying channels.

Simulation results are shown in Figure 5.2b. As in the case of VoIP traffic, the joint debt-channel policy incurs very small total delivery debt while all the other policies have significantly higher total delivery debts.

Chapter 6

Utility Maximization without Rate Adaptation

6.1 Problem Formulation and Decomposition

In the previous chapters, it is assumed that the timely throughput requirements, $[q_n]$, are given and fixed. In this chapter, we address the problem of how to choose $q := [q_n]$ so that the total utility of all the clients in the system can be maximized. We consider the scenario as discussed in Chapter 3, where rate adaptation is not applied, channel state is static, and all clients require the same delay bound T. As in Chapter 3, we use p_n, instead of $p_{c,n}$, to denote the channel reliability between the AP and client n.

We begin by supposing that each client has a certain utility function, $U_n(q_n)$, which is strictly increasing, strictly concave, and continuously differentiable function over the range $0 < q_n \leq 1$, with the value at 0 defined as the right limit, possibly $-\infty$. The problem of choosing q_n to maximize the total utility, under the feasibility constraint of Theorem 4, can be described by the following convex optimization problem:

SYSTEM:

$$\text{Max } \sum_{i=1}^{N} U_i(q_i) \tag{6.1}$$

$$\text{s.t. } \sum_{i \in S} \frac{q_i}{p_i} \leq T - I_S, \forall S \subseteq \{1, 2, \ldots, N\}, \tag{6.2}$$

$$\text{over } q_n \geq 0, \forall 1 \leq n \leq N. \tag{6.3}$$

It may be difficult to solve $SYSTEM$ directly. So, we decompose it into two simpler problems, namely, $CLIENT$ and $ACCESS\text{-}POINT$, as described below. This decomposition was first introduced by Kelly [41], though in the context of dealing with rate control for non-real time traffic.

Suppose client n is willing to pay an amount of ρ_n per interval, and receives a long-term average timely throughput q_n proportional to ρ_n, with $\rho_n = \psi_n q_n$. If $\psi_n > 0$, the utility maximization problem for client n is:

CLIENT$_n$:

$$\text{Max } U_n(\frac{\rho_n}{\psi_n}) - \rho_n \tag{6.4}$$

$$\text{over } 0 \leq \rho_n \leq \psi_n. \tag{6.5}$$

On the other hand, given that client n is willing to pay ρ_n per interval, we suppose that the AP wishes to find the vector q to maximize $\sum_{i=1}^{N} \rho_i \log q_i$, under the feasibility constraints. In other words, the AP has to solve the following optimization problem:

ACCESS-POINT:

$$\text{Max } \textstyle\sum_{i=1}^{N} \rho_i \log q_i \tag{6.6}$$

$$\text{s.t. } \textstyle\sum_{i \in S} \frac{q_i}{p_i} \leq T - I_S, \forall S \subseteq \{1, 2, \ldots, N\}, \tag{6.7}$$

$$\text{over } q_n \geq 0, \forall 1 \leq n \leq N. \tag{6.8}$$

We begin by showing that solving $SYSTEM$ is equivalent to jointly solving $CLIENT_n$ and $ACCESS\text{-}POINT$.

Theorem 10. *There exist non-negative vectors q, $\rho := [\rho_n]$, and $\psi := [\psi_n]$, with $\rho_n = \psi_n q_n$, such that:*

(i) For n such that $\psi_n > 0$, ρ_n is a solution to $CLIENT_n$;

(ii) Given that each client n pays ρ_n per interval, q is a solution to $ACCESS\text{-}POINT$.

Further, if q, ρ, and ψ are all positive vectors, the vector q is also a solution to $SYSTEM$.

Proof. We will first show the existence of q, ρ, and ψ that satisfy (i) and (ii). We will then show that the resulting q is also the solution to $SYSTEM$.

There exists some $\epsilon > 0$ so that by letting $q_n \equiv \epsilon$, the vector q is an interior point of the feasible region for both $SYSTEM$ (6.2) (6.3), and $ACCESS\text{-}POINT$ (6.7) (6.8). Also, by setting $\rho_n \equiv \epsilon$, ρ_n is also an interior point of the feasible region for $CLIENT_n$ (6.5). Therefore, by Slater's condition, a feasible point for $SYSTEM$, $CLIENT_n$, or $ACCESS\text{-}POINT$, is the optimal solution for the respective problem if and only if it satisfies the corresponding Karush-Kuhn-Tucker (KKT) condition for the problem. Further, since the feasible region for each of the problems is compact, and the utilities are continuous on it, or since the utility converges to $-\infty$ at $q_n = 0$, there exists an optimal solution to each of them.

The Lagrangian of $SYSTEM$ is:

$$L_{SYS}(q,\lambda,\nu) := -\sum_{i=1}^{N} U_i(q_i) + \sum_{S\subseteq\{1,2,\ldots,N\}} \lambda_S[\sum_{i\in S}\frac{q_i}{p_i} - (T - I_S)] - \sum_{i=1}^{N}\nu_i q_i,$$

where $\lambda := [\lambda_S : S \subseteq \{1,2,\ldots,N\}]$ and $\nu := [\nu_n : 1 \leq n \leq N]$ are the Lagrange multipliers. By the KKT condition, a vector $q^* := [q_1^*, q_2^*, \ldots, q_N^*]$ is the optimal solution to SYSTEM if q^* is feasible and there exists vectors λ^* and ν^* such that:

$$\begin{aligned}\left.\frac{\partial L_{SYS}}{\partial q_n}\right|_{q^*,\lambda^*,\nu^*} &= -U_n'(q_n^*) + \frac{\sum_{S\ni n}\lambda_S^*}{p_n} - \nu_n^* \\ &= 0, \forall 1 \leq n \leq N,\end{aligned} \tag{6.9}$$

$$\lambda_S^*[\sum_{i\in S}\frac{q_i^*}{p_i} - (T - I_S)] = 0, \forall S \subseteq \{1,2,\ldots,N\}, \tag{6.10}$$

$$\nu_n^* q_n^* = 0, \forall 1 \leq n \leq N, \tag{6.11}$$

$$\lambda_S^* \geq 0, \forall S \subseteq \{1,\ldots,N\}, \text{ and } \nu_n^* \geq 0, \forall 1 \leq n \leq N. \tag{6.12}$$

The Lagrangian of $CLIENT_n$ is:

$$L_{CLI}(\rho_n,\xi_n) := -U_n(\frac{\rho_n}{\psi_n}) + \rho_n - \xi_n\rho_n,$$

where ξ_n is the Lagrange multiplier for $CLIENT_n$. By the KKT condition, ρ_n^* is the optimal solution to $CLIENT_n$ if $\rho_n^* \geq 0$ and there exists ξ_n^* such that:

$$\left.\frac{dL_{CLI}}{d\rho_n}\right|_{\rho_n^*,\xi_n^*} = -\frac{1}{\psi_n}U_n'(\frac{\rho_n^*}{\psi_n}) + 1 - \xi_n^* = 0, \tag{6.13}$$

$$\xi_n^*\rho_n^* = 0, \tag{6.14}$$

$$\xi_n^* \geq 0. \tag{6.15}$$

Finally, the Lagrangian of $ACCESS\text{-}POINT$ is:

$$L_{NET}(q,\zeta,\mu) := -\sum_{i=1}^{N}\rho_i \log q_i + \sum_{S\subseteq\{1,2,\ldots,N\}}\zeta_S[\sum_{i\in S}\frac{q_i}{p_i} - (T - I_S)] - \sum_{i=1}^{N}\mu_i q_i,$$

where $\zeta := [\zeta_S : S \subseteq \{1,2,\ldots,N\}]$ and $\mu := [\mu_n : 1 \leq n \leq N]$ are the Lagrange multipliers. Again, by the KKT condition, a vector $q^* := [q_n^*]$ is the optimal solution to $ACCESS\text{-}POINT$ if q^* is feasible and there

exists vectors ζ^* and μ^* such that:

$$\begin{aligned}\frac{\partial L_{NET}}{\partial q_n}\Big|_{q^*,\zeta^*,\mu^*} &= -\frac{\rho_n}{q_n^*} + \frac{\sum_{S\ni n}\zeta_S^*}{p_n} - \mu_n^* \\ &= 0, \forall 1 \le n \le N,\end{aligned} \tag{6.16}$$

$$\zeta_S^*[\sum_{i\in S}\frac{q_i^*}{p_i} - (T - I_S)] = 0, \forall S \subseteq \{1,2,\ldots,N\}, \tag{6.17}$$

$$\mu_n^* q_n^* = 0, \forall 1 \le n \le N, \tag{6.18}$$

$$\zeta_S^* \ge 0, \forall S \subseteq \{1,\ldots,N\}, \text{ and } \mu_n^* \ge 0, \forall 1 \le n \le N. \tag{6.19}$$

Let q^* be a solution to $SYSTEM$, and let λ^*, ν^* be the corresponding Lagrange multipliers that satisfy conditions (6.9)–(6.12). Let $q_n = q_n^*$, $\rho_n = \frac{\sum_{S\ni n}\lambda_S^*}{p_n} q_n^*$, and $\psi_n = \frac{\sum_{S\ni n}\lambda_S^*}{p_n}$, for all n. Clearly, q, ρ, and ψ are all non-negative vectors. We will show (q,ρ,ψ) satisfy (i) and (ii).

We first show (i) for all n such that $\psi_n = \frac{\sum_{S\ni n}\lambda_S^*}{p_n} > 0$. It is obvious that $\rho_n = \psi_n q_n$. Also, $\rho_n \ge 0$, since $\lambda_S^* \ge 0$ (by (6.12)) and $q_n^* \ge 0$ (since q^* is feasible). Further, let the Lagrange multiplier of $CLIENT_n$, ξ_n, be equal to $\nu_n^*/\frac{\sum_{S\ni n}\lambda_S^*}{p_n} = \nu_n^*/\psi_n$. Then we have:

$$\begin{aligned}\frac{\partial L_{CLI}}{\partial \rho_n}\Big|_{\rho_n,\xi_n} &= -\frac{1}{\psi_n}U_n'(\frac{\rho_n}{\psi_n}) + 1 - \xi_n \\ &= \frac{1}{\psi_n}(-U_n'(\frac{\rho_n}{\psi_n}) + \psi_n - \psi_n\xi_n) \\ &= \frac{1}{\psi_n}(-U_n'(q_n^*) + \frac{\sum_{S\ni n}\lambda_S^*}{p_n} - \nu_n^*) = 0, \text{ by (6.9)},\end{aligned}$$

$$\xi_n\rho_n = \frac{\nu_n^*}{\psi_n}\psi_n q_n^* = \nu_n^* q_n^* = 0, \text{ by (6.11)}$$

$$\xi_n = \nu_n^*/\frac{\sum_{S\ni n}\lambda_S^*}{p_n} \ge 0, \text{ by (6.12)}.$$

In sum, (ρ,ψ,ξ) satisfies the KKT conditions for $CLIENT_n$, and therefore ρ_n is a solution to $CLIENT_n$, with $\rho_n = \psi_n q_n$.

Next we establish (ii). Since $q = q^*$ is the solution to $SYSTEM$, it is feasible. Let the Lagrange multipliers of $ACCESS\text{-}POINT$ be $\zeta_S = \lambda_S^*, \forall S$, and $\mu_n = 0, \forall n$, respectively. Given that each client n pays

ρ_n per interval, we have:

$$\begin{aligned}
\left.\frac{\partial L_{NET}}{\partial q_n}\right|_{q,\zeta,\mu} &= -\frac{\rho_n}{q_n} + \frac{\sum_{S\ni n}\zeta_S}{p_n} - \mu_n \\
&= -\psi_n + \psi_n - 0 = 0, \forall n, \\
\zeta_S[\textstyle\sum_{i\in S}\frac{q_i}{p_i} - (T - I_S)] &= \lambda^*_S[\textstyle\sum_{i\in S}\frac{q^*_i}{p_i} - (T - I_S)] \\
&= 0, \forall S, \text{ by (6.10)}, \\
\mu_n q_n &= 0 \times q_n = 0, \forall n, \\
\zeta_S &= \lambda^*_S \geq 0, \forall S \text{ (by (6.12))}, \text{ and } \mu_n \geq 0, \forall n.
\end{aligned}$$

Therefore, (q, ζ, μ) satisfies the KKT condition for $ACCESS\text{-}POINT$ and thus q is a solution to $ACCESS\text{-}POINT$.

For the converse, suppose (q, ρ, ψ) are positive vectors with $\rho_n = \psi_n q_n$, for all n, that satisfy (i) and (ii). We wish to show that q is a solution to $SYSTEM$. Let ξ_n be the Lagrange multiplier for $CLIENT_n$. Since we assume $\psi_n > 0$ for all n, the problem $CLIENT_n$ is well-defined for all n, and so is ξ_n. Also, let ζ and μ be the Lagrange multipliers for $ACCESS\text{-}POINT$. Since $q_n > 0$ for all n, we have $\mu_n = 0$ for all n by (6.18). By (6.16), we also have:

$$\begin{aligned}
\left.\frac{\partial L_{NET}}{\partial q_n}\right|_{q,\zeta,\mu} &= -\frac{\rho_n}{q_n} + \frac{\sum_{S\ni n}\zeta_S}{p_n} - \mu_n \\
&= -\psi_n + \frac{\sum_{S\ni n}\zeta_S}{p_n} = 0,
\end{aligned}$$

and thus $\psi_n = \frac{\sum_{S\ni n}\zeta_S}{p_n}$. Let $\lambda_S = \zeta_S$, for all S, and $\nu_n = \psi_n \xi_n$, for all n. We claim that q is the optimal solution to $SYSTEM$ with Lagrange multipliers λ and ν.

Since q is a solution to $ACCESS\text{-}POINT$, it is feasible. Further, we have:

$$\begin{aligned}
\left.\frac{\partial L_{SYS}}{\partial q_n}\right|_{q,\lambda,\nu} &= -U_n'(q_n) + \frac{\sum_{S\ni n}\lambda_S}{p_n} - \nu_n \\
&= -U_n'(\tfrac{\rho_n}{\psi_n}) + \psi_n - \psi_n\xi_n = 0, \forall n, \text{ by (6.13)}, \\
\lambda_S[\textstyle\sum_{n\in S}\frac{q_n}{p_n} - (T - I_S)] &= \zeta_S[\textstyle\sum_{n\in S}\frac{q_n}{p_n} - (T - I_S)] \\
&= 0, \forall S, \text{ by (6.17)}, \\
\nu_n q_n &= \xi_n \rho_n = 0, \forall n, \text{ by (6.14)}, \\
\lambda_S &= \zeta_S \geq 0, \forall S, \text{ by (6.19)}, \\
\nu_n &= \psi_n \xi_n \geq 0, \forall n, \text{ by (6.15)}.
\end{aligned}$$

Thus, (q, λ, ν) satisfy the KKT condition for $SYSTEM$, and so q is a solution to $SYSTEM$. □

6.2 A Bidding Game between Clients and Access Point

Above, we have shown that the maximum total utility of the system can be achieved when the solutions to the problems $CLIENT_n$ and $ACCESS\text{-}POINT$ agree. In this section, we formulate a repeated game for such reconciliation. We also discuss the meanings of the problems $CLIENT_n$ and $ACCESS\text{-}POINT$ in this repeated game.

The repeated game is formulated as follows:

Step 1: Each client n announces an amount ρ_n that it pays per interval.

Step 2: After noting the amounts, $\rho_1, \rho_2, \ldots, \rho_N$, paid by the clients, the AP chooses a scheduling policy so that the resulting long-term timely throughput, q_n, for each client maximizes $\sum_{i=1}^{N} \rho_i \log q_i$, subject to feasibility of $[q_n]$.

Step 3: The client n observes its own timely throughput, q_n. It computes $\psi_n := \rho_n/q_n$. It then determines $\rho_n^* \geq 0$ to maximize $U_n(\frac{\rho_n^*}{\psi_n}) - \rho_n^*$. Client n updates the amount it pays to $(1-\alpha)\rho_n + \alpha\rho_n^*$, with some fixed $0 < \alpha < 1$, and announces the new bid value.

Step 4: Go back to Step 2.

In Step 3 of the game, client n chooses its new amount of payment as a weighted average of the past amount and the derived optimal value, instead of the derived optimal value. This design serves two purposes. First, it seeks to avoid the system oscillating between two extreme values. Second, since ρ_n is initiated to a positive value, and ρ_n^* derived in each iteration is always non-negative, this design guarantees ρ_n to be positive throughout all iterations. Since $\psi_n = \rho_n/q_n$, this also ensures $\psi_n > 0$ and the function $U_n(\frac{\rho_n}{\psi_n})$ is consequently always well-defined.

We show that the fixed point of this repeated game maximizes the total utility of the system:

Theorem 11. *Suppose at the fixed point of the repeated game, each client n pays ρ_n^* per interval, and receives timely throughput q_n^*. If both ρ_n^* and q_n^* are positive for all n, the vector q^* maximizes the total utility of the system.*

Proof. Let $\psi_n^* = \frac{\rho_n^*}{q_n^*}$. It is positive since both ρ_n^* and q_n^* are positive. Since the vectors q^* and ρ^* are derived from the fixed point, ρ_n^* maximizes $U_n(\frac{\rho_n}{\psi_n^*}) - \rho_n$, over all $\rho_n \geq 0$, as described in Step 3 of the game. Thus, ρ_n^* is a solution to $CLIENT_n$, given $\rho_n^* = \psi_n^* q_n^*$. Similarly, from Step 2, q^* is the feasible vector that maximizes

$\sum_{i=1}^{N} \rho_i^* \log q_i$, over all feasible vectors q. Thus, q^* is a solution to $ACCESS\text{-}POINT$, given that each client n pays ρ_n^* per interval. By Theorem 10, q^* is the unique solution to $SYSTEM$ and therefore maximizes the total utility of the system. □

Next, we describe the meaning of the game. In Step 3, client n assumes a linear relation between the amount it pays, ρ_n, and the timely throughput it receives, q_n. To be more precise, it assumes $\rho_n = \psi_n q_n$, where ψ_n is the price. Thus, maximizing $U_n(\frac{\rho_n}{\psi_n}) - \rho_n$ is equivalent to maximizing $U_n(q_n) - \rho_n$. Recall that $U_n(q_n)$ is the utility that client n obtains when it receives timely throughput q_n. $U_n(q_n) - \rho_n$ is therefore the net profit that client n gets. In short, in Step 3, the goal of client n is to selfishly maximize its own net profit using a first order linear approximation to the relation between payment and timely throughput.

We next discuss the behavior of the AP in Step 2. The AP schedules clients so that the resulting timely throughput vector q is a solution to the problem $ACCESS\text{-}POINT$, given that each client n pays ρ_n per interval. Thus, q is feasible and there exist vectors ζ and μ that satisfy conditions (6.16)–(6.19). While it is difficult to solve this problem, we consider a special restrictive case that gives us a simple solution and insight into the AP's behavior, which we will subsequently generalize. Let $TOT := \{1, 2, \ldots, N\}$ be the set that consists of all clients. We assume that a solution (q, ζ, μ) to the problem has the following properties: $\zeta_S = 0$, for all $S \neq TOT$, $\zeta_{TOT} > 0$, and $\mu_n = 0$, for all n. By (6.16), we have:

$$-\frac{\rho_n}{q_n} + \frac{\sum_{S \ni n} \zeta_S}{p_n} - \mu_n = -\frac{\rho_n}{q_n} + \frac{\zeta_{TOT}}{p_n} = 0,$$

and therefore $q_n = p_n \rho_n / \zeta_{TOT}$. Further, since $\zeta_{TOT} > 0$, (6.17) requires that:

$$\sum_{i \in TOT} \frac{q_i}{p_i} - (T - I_{TOT}) = \sum_{i \in TOT} \frac{\rho_i}{\zeta_{TOT}} - (T - I_{TOT}) = 0.$$

Thus, $\zeta_{TOT} = \frac{\sum_{i=1}^{N} \rho_i}{T - I_{TOT}}$ and $\frac{q_n}{p_n} = \frac{\rho_n}{\sum_{i=1}^{N} m_i}(T - I_{TOT})$, for all n. Notice that the derived (q, ζ, μ) satisfies conditions (6.16)–(6.19). Thus, under the assumption that q is feasible, this special case actually maximizes $\sum_{i=1}^{N} \rho_i \log q_i$. In Section 6.3 we will address the general situation without any such assumption, since it need not be true.

Recall that I_{TOT} is the average number of time slots that the AP is forced to be idle in a interval after it has completed all clients. Also, by Lemma 1, $\frac{q_n}{p_n}$ is the workload of client n, that is, the average number of time slots that the AP should spend working for client n. Thus, letting $\frac{q_n}{p_n} = \frac{\rho_n}{\sum_{i=1}^{N} \rho_i}(T - I_{TOT})$, for all n, the AP tries to allocate those non-idle time slots so that the average number of time slots each client gets is proportional to its payment. Although we only study the special case of I_{TOT} here, we will show that the same behavior also holds for the general case in the Section 6.3.

In summary, the game proposed in this section actually describes a bidding game, where clients are bidding for non-idle time slots. Each client gets a share of time slots that is proportional to its bid. The AP thus assigns timely throughputs, based on which the clients calculate a price and selfishly maximize their own net profits. Finally, Theorem 11 states that the equilibrium point of this game maximizes the total utility of the system.

6.3 A Scheduling Policy for Solving $ACCESS\text{-}POINT$

In Section 6.2, we have shown that by setting $q_n = p_n \frac{\rho_n}{\sum_{i=1}^{N} \rho_i}(T - I_{TOT})$, the resulting vector q solves $ACCESS\text{-}POINT$ provided q is indeed feasible. Unfortunately, such q is not always feasible and solving $ACCESS\text{-}POINT$ is in general difficult. Even for the special case discussed in Section 6.2, solving $ACCESS\text{-}POINT$ requires knowledge of channel conditions, that is, p_n. In this section, we propose a very simple priority based scheduling policy that can achieve the optimal solution for $ACCESS\text{-}POINT$, and that too without any knowledge of the channel conditions.

In the special case discussed in Section 6.2, the AP tries, though it may be impossible in general, to allocate non-idle time slots to clients in proportion to their payments. Based on this intuitive guideline, we design the following scheduling policy. Let $u_n(t)$ be the number of time slots that the AP has allocated for client n up to time t. At the beginning of each interval, the AP sorts all clients in increasing order of $\frac{u_n(t)}{\rho_n}$, so that $\frac{u_1(t)}{\rho_1} \leq \frac{u_2(t)}{\rho_2} \leq \ldots$ after renumbering clients if necessary. The AP then schedules transmissions according to the priority ordering, where clients with smaller $\frac{u_n(t)}{\rho_n}$ get higher priorities. Specifically, in each time slot during the interval, the AP chooses the smallest i for which the packet for client i is not yet delivered, and then transmits the packet for client i in that time slot. We call this the *weighted transmission* policy (WT). Notice that the policy only requires the AP to keep track of the bids of clients and the number of time slots each client has been allocated in the past, followed by a sorting of $\frac{u_n(t)}{\rho_n}$ among all clients. Thus, the policy requires no information on the actual channel conditions, and is tractable. Simple as it is, we show that the policy actually achieves the optimal solution for $ACCESS\text{-}POINT$. In the following sections, we first prove that the vector of timely throughputs resulting from the WT policy converges to a single point. We then prove that this limit is the optimal solution for $ACCESS\text{-}POINT$.

6.3.1 Convergence of the Weighted Transmission Policy

We now prove that, by applying the WT policy, the timely throughputs of clients will converge to a vector q. To do so, we actually prove the convergence property and precise limit of a more general class of scheduling

policies, which not only consists of the WT policy but also the largest time-based debt scheduling policy considered earlier. The proof is similar to that used in Chapter 3 and is also based on Blackwell's approachability theorem [8].

Now we formulate our more general class of scheduling policies. We call a policy a *generalized transmission time policy* if, for a choice of a positive parameter vector a and non-negative parameter vector b, the AP sorts clients by $a_n u_n(t) - b_n t$ at the beginning of each interval, and gives priorities to clients with lower values of this quantity. Note that the special case $a_n \equiv \frac{1}{\rho_n}$ and $b_n \equiv 0$ yields the WT policy, while the choice $a_n \equiv 1$ and $b_n \equiv \frac{q_n}{Tp_n}$ yields the largest time-based debt first policy.

Theorem 12. *For each generalized transmission time policy, there exists a vector q such that the vector of workloads resulting from the policy converges to $w(q) := [w_n(q_n)]$.*

Proof. Given the parameters $\{(a_n, b_n) : 1 \leq n \leq N\}$, we give an exact expression for the limiting q. We define a sequence of sets $\{H_k\}$ and corresponding values $\{\theta_k\}$ iteratively as follows. Let $H_0 := \phi$, $\theta_0 := -\infty$, and

$$H_k := \arg \min_{S: S \supsetneq H_{k-1}} \frac{\frac{1}{T}(I_{H_{k-1}} - I_S) - \sum_{n \in S \setminus H_{k-1}} \frac{b_n}{a_n}}{\sum_{n \in S \setminus H_{k-1}} 1/a_n},$$
$$\theta_k := \frac{\frac{1}{T}(I_{H_{k-1}} - I_{H_k}) - \sum_{n \in H_k \setminus H_{k-1}} \frac{b_n}{a_n}}{\sum_{n \in H_k \setminus H_{k-1}} 1/a_n}, \text{ for all } k > 0.$$

In selecting H_k, we always choose a maximal subset, breaking ties arbitrary. $(H_1, \theta_1), (H_2, \theta_2), \ldots$, can be iteratively defined until every client is in some H_k. Also, by the definition, we have $\theta_k > \theta_{k-1}$, for all $k > 0$. If client n is in $H_k \setminus H_{k-1}$, we define $q_n := Tp_n \frac{b_n + \theta_k}{a_n}$, and so $w_n(q_n) = T\frac{b_n + \theta_k}{a_n}$. The proof of convergence consists of two parts. First we prove that the vector of *work performed*, defined as the vector of average numbers of time slots that the AP spends on transmitting the packet for each client, approaches the set $\{w^* | w_n^* \geq w_n(q_n)\}$. Then we prove that $w(q)$ is the only feasible vector in the set $\{w^* | w_n^* \geq w_n(q_n)\}$. Since the *feasible region for workloads*, defined as the set of all feasible vectors for workloads, is approachable under any policy, the vector of work performed resulting from the generalized transmission time policy must converge to $w(q)$.

For the first part, we prove the following statement: for each $k \geq 1$, the set $W_k := \{w^* | w_n^* \geq T\frac{b_n + \theta_k}{a_n}, \forall n \notin H_{k-1}\}$ is approachable. Since $\cap_{i \geq 0} W_i = \{w^* | w_n^* \geq w_n(q_n)\}$, we also prove that $\{w^* | w_n^* \geq w_n(q_n)\}$ is approachable.

Consider a linear transformation on the space of workloads $L(w) := [l_n : l_n = \frac{a_n w_n / T - b_n}{\sqrt{a_n}}]$. Proving W_k is approachable is equivalent to proving that its image under L, $V_k := \{l | l_n \geq \frac{\theta_k}{\sqrt{a_n}}, \forall n \notin H_{k-1}\}$, is approachable. Now we apply Blackwell's theorem. Suppose at some time t that is the beginning of a interval,

the number of time slots that the AP has worked on client n is $u_n(t)$. The work performed for client n is $\frac{u_n(t)}{t/T}$, and the image of the vector of work performed under L is $x(t) := [x_n(t)|x_n(t) = \frac{a_n u_n(t)/t - b_n}{\sqrt{a_n}}]$, which we shall suppose is not in V_k. The generalized transmission time policy sorts clients so that $a_1u_1(t) - b_1 \leq a_2u_2(t) - b_2 \leq \ldots$, or equivalently, $\sqrt{a_1}x_1(t) \leq \sqrt{a_2}x_2(t) \leq \ldots$. The closest point in V_k to $x(t)$ is $y := [y_n]$, where $y_n = \frac{\theta_k}{\sqrt{a_n}}$, if $x_n(t) < \frac{\theta_k}{\sqrt{a_n}}$ and $n \notin H_{k-1}$, and $y_n = x_n$, otherwise. The hyperplane that passes through y and is orthogonal to the line segment xy is:

$$\{z|f(z) := \sum_{n:n\leq n_0, n\notin H_{k-1}} (z_n - \frac{\theta_k}{\sqrt{a_n}})(x_n(t) - \frac{\theta_k}{\sqrt{a_n}}) = 0\}.$$

Let π_n be the expected number of time slots that the AP spends on working for client n in this interval under the generalized transmission time policy. The image under L of the expected reward in this interval is $\pi_L := [\frac{a_n\pi_n/T - b_n}{\sqrt{a_n}}]$. Blackwell's theorem shows that V_k is approachable if $x(t)$ and π_L are separated by the plane $\{z|f(z) = 0\}$. Since $f(x(t)) \geq 0$, it suffices to show $f(\pi_L) \leq 0$.

We manipulate the original ordering, for this interval, so that all clients in H_{k-1} have higher priorities than those not in H_{k-1}, while preserving the relative ordering between clients not in H_{k-1}. Note that this manipulation will not give any client $n \notin H_{k-1}$ higher priority than it had in the original ordering. Therefore, π_n will not increase for any $n \notin H_{k-1}$. Since the value of $f(\pi_L)$ only depends on π_n for $n \notin H_{k-1}$, and increases as those π_n decrease, this manipulation will not decrease the value of $f(\pi_L)$. Thus, it suffices to prove that $f(\pi_L) \leq 0$, under this new ordering. Let $n_0 := |H_{k-1}| + 1$. Under this new ordering, we have: $\sqrt{a_{n_0}}x_{n_0}(t) \leq \sqrt{a_{n_0+1}}x_{n_0+1}(t) \leq \cdots \leq \sqrt{a_{n_1}}x_{n_1}(t) < \theta_k \leq \sqrt{a_{n_1+1}}x_{n_1+1}(t) \leq \ldots$.

Let $\delta_n = \sqrt{a_n}x_n(t) - \sqrt{a_{n+1}}x_{n+1}(t)$, for $n_0 \leq n \leq n_1 - 1$ and $\delta_{n_1} = \sqrt{a_{n_1}}x_{n_1}(t) - \theta_k$. Clearly, $\delta_n \leq 0$, for all $n_0 \leq n \leq n_1$. Now we can derive:

$$\begin{aligned}
f(\pi_L) &= \sum_{n=n_0}^{n_1} (\frac{a_n\pi_n/T - b_n}{\sqrt{a_n}} - \frac{\theta_k}{\sqrt{a_n}})(x_n(t) - \frac{\theta_k}{\sqrt{a_n}}) \\
&= \sum_{n=n_0}^{n_1} (\frac{\pi_n}{T} - \frac{b_n}{a_n} - \frac{\theta_k}{a_n})(\sqrt{a_n}x_n(t) - \theta_k) \\
&= \sum_{i=n_0}^{n_1} (\frac{\sum_{n=n_0}^{i}\pi_n}{T} - \sum_{n=n_0}^{i}\frac{b_n}{a_n} - \theta_k\sum_{n=n_0}^{i}\frac{1}{a_n})\delta_i.
\end{aligned}$$

Recall that I_S is the expected number of idle time slots when the AP only caters on the subset S. Thus, under this ordering, we have $\sum_{n=1}^{i}\pi_n = T - I_{\{1,\ldots,i\}}$, for all i, and $\sum_{n=n_0}^{i}\pi_n = I_{\{1,\ldots,n_0-1\}} - I_{\{1,\ldots,i\}} =$

$I_{H_{k-1}} - I_{\{1,\ldots,i\}}$, for all $i \geq n_0$. By the definition of H_k and θ_k, we also have

$$\begin{aligned}
&\frac{\sum_{n=n_0}^{i} \pi_n}{T} - \sum_{n=n_0}^{i} \frac{b_n}{a_n} - \theta_k \sum_{n=n_0}^{i} \frac{1}{a_n} \\
=&(\sum_{n=n_0}^{i} \frac{1}{a_n})(\frac{\frac{1}{T}(I_{H_{k-1}} - I_{\{1,\ldots,i\}}) - \sum_{n=n_0}^{i} \frac{b_n}{a_n}}{\sum_{n \in \{1,\ldots,i\} \backslash H_{k-1}} 1/a_n} - \theta_k) \geq 0.
\end{aligned}$$

Therefore, $f(\pi_L) \leq 0$, since $\delta_i \leq 0$, and V_k is indeed approachable, for all k.

We have established that the set $\{w^* | w_n^* \geq w_n(q_n)\}$ is approachable. Next we prove that $[w_n(q_n)]$ is the only feasible vector in the set. Consider any vector $w' \neq w(q)$ in the set. We have $w'_n \geq w_n(q_n)$ for all n, and $w'_{n_0} > w_{n_0}(q_{n_0})$, for some n_0. Suppose $n_0 \in H_k \backslash H_{k-1}$. We have:

$$\begin{aligned}
\sum_{n \in H_k} w'_n > \sum_{n \in H_k} w_n(q_n) = \sum_{i=1}^{k} \sum_{n \in H_i \backslash H_{i-1}} T \frac{b_n + \theta_k}{a_n} \\
= \sum_{i=1}^{k} (I_{H_{i-1}} - I_{H_i}) = T - I_{H_k},
\end{aligned}$$

and thus w' is not feasible. Therefore, $w(q)$ is the only feasible vector in $\{w^* | w_n^* \geq w_n(q_n)\}$, and the vector of work performed resulting from the generalized transmission time policy must converge to $w(q)$. □

Corollary 1. *For the policy of Theorem 12, the vector of timely throughputs converges to q.*

Proof. Follows from Lemma 1. □

6.3.2 Optimality of the Weighted Transmission Policy for $ACCESS\text{-}POINT$

Theorem 13. *Given $[\rho_n]$, the vector q of long-term average timely throughputs resulting from the WT policy is a solution to $ACCESS\text{-}POINT$.*

Proof. We use the sequence of sets $\{H_k\}$ and values $\{\theta_k\}$, with $a_n := \frac{1}{\rho_n}$ and $b_n := 0$, as defined in the proof of Theorem 12. Let $K := |\{\theta_k\}|$. Thus, we have $H_K = TOT = \{1, 2, \ldots, N\}$. Also, let $m_k := |H_k|$. For convenience, we renumber clients so that $H_k = \{1, 2, \ldots, m_k\}$. The proof of Theorem 12 shows that $q_n = T p_n \theta_k \rho_n$, for $n \in H_k \backslash H_{k-1}$. Therefore, $w_n(q_n) = \frac{q_n}{p_n} = T\theta_k \rho_n$. Obviously, q is feasible, since it is indeed achieved by the WT policy. Thus, to establish optimality, we only need to prove the existence of vectors ζ and μ that satisfy conditions (6.16)–(6.19).

Set $\mu_n = 0$, for all n. Let $\zeta_{H_K} = \zeta_{TOT} := \frac{\rho_N}{w_N(q_N)} = \frac{1}{T\theta_K}$ and $\zeta_{H_k} := \frac{\rho_{m_k}}{w_{m_k}(q_{m_k})} - \frac{\rho_{m_{k+1}}}{w_{m_{k+1}}(q_{m_{k+1}})} = \frac{1}{T\theta_k} - \frac{1}{T\theta_{k+1}}$, for $1 \leq k \leq K-1$. Finally, let $\zeta_S := 0$, for all $S \notin \{H_1, H_2, \ldots, H_K\}$. We claim that the vectors ζ and μ, along with q, satisfy conditions (6.16)–(6.19).

We first evaluate condition (6.16). Suppose client n is in $H_k \backslash H_{k-1}$. Then client n is also in $H_{k+1}, H_{k+2}, \ldots, H_K$. So,

$$\begin{aligned} &-\frac{\rho_n}{q_n} + \frac{\sum_{S \ni n} \zeta_S}{p_n} - \mu_n = -\frac{1}{T\theta_k p_n} + \frac{\sum_{i=k}^{K} \zeta_{H_i}}{p_n} \\ =& -\frac{1}{T\theta_k p_n} + \frac{1}{T\theta_k p_n} = 0. \end{aligned}$$

Thus, condition (6.16) is satisfied.

Since $\mu_n = 0$, for all n, condition (6.18) is satisfied. Further, since $\frac{1}{\theta_k} > \frac{1}{\theta_{k+1}}$, for all $1 \leq k \leq K-1$, condition (6.19) is also satisfied. It remains to establish condition (6.17). Since $\zeta_S = 0$ for all $S \notin \{H_1, H_2, \ldots, H_K\}$, we only need to show $\sum_{i \in S} \frac{q_i}{p_i} - (T - I_S) = 0$ for $S \in \{H_1, H_2, \ldots, H_K\}$.

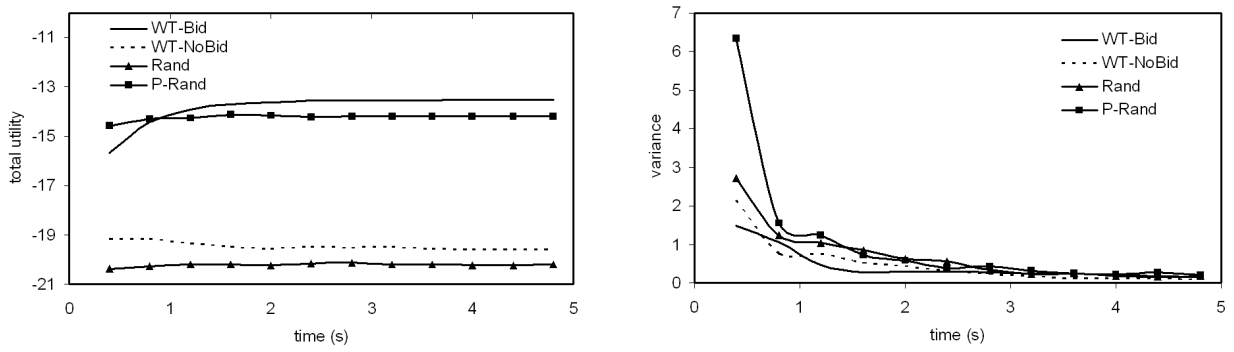
Consider H_k. For each client $i \in H_k$ and each client $j \notin H_k$, $\frac{w_i(q_i)}{\rho_i} < \frac{w_j(q_j)}{\rho_j}$. Since $w_n(q_n)$ is the average number of time slots that the AP spends on working for client n, we have $\frac{u_i(t)}{\rho_i} < \frac{u_j(t)}{\rho_j}$, for all $i \in H_k$ and $j \notin H_k$, after a finite number of intervals. Therefore, except for a finite number of intervals, clients in H_k will have priorities over those not in H_k. In other words, if we only consider the behavior of those clients in H_k, it is the same as if the AP only works on the subset H_k of clients. Further, recall that I_{H_k} is the expected number of time slots that the AP is forced to stay idle when the AP only works on the subset H_k of clients. Thus, we have $\sum_{i \in H_k} w_i(q_i) = T - I_{H_k}$ and $\sum_{i \in H_k} \frac{q_i}{p_i} - (T - I_{H_k}) = 0$, for all k. □

6.4 Simulation Results

In this section, we present the simulation results of the total utility that is achieved by iterating between the bidding game and the WT policy, which we call WT-Bid. We assume that the utility function of each client n is given by $\gamma_n \frac{q_n^{\alpha_n} - 1}{\alpha_n}$, where γ_n is a positive integer and $0 < \alpha_n < 1$. This utility function is strictly increasing, strictly concave, and differentiable for any γ_n and α_n. In addition to evaluating the policy WT-Bid, we also compare the results of three other policies: a policy that employs the WT policy but without updating the bids from clients, which we call WT-NoBid; a policy that decides priorities randomly among clients at the beginning of each period, which we call Rand; and a policy that gives clients with larger γ_n higher priorities, with ties broken randomly, which we call P-Rand.

In each simulation, we assume there are 30 clients. The n^{th} client has channel reliability $p_n = (50+n)\%$, $\gamma_n = (n \mod 3) + 1$, and $\alpha_n = 0.3 + 0.1(n \mod 5)$. In addition to plotting the average of total utility over all simulation runs, we also plot the variance of total utility.

We present the simulation results for VoIP traffic by assuming all clients generate one packet in each period. Fig. 6.1 shows the simulation results. The WT-Bid policy not only achieves the highest average total



(a) Average of total utility

(b) Variance of total utility

Figure 6.1: Performance of total utility

utility but also the smallest variance. This result suggests that the WT-Bid policy converges very fast. On the other hand, the WT-NoBid policy fails to provide satisfactory performance since it does not consider the different utility functions that clients may have. The P-Rand policy offers much better performance than both the WT-NoBid policy and the Rand policy since it correctly gives higher priority to clients with higher γ_n. Still, it cannot differentiate between clients with the same γ_n and thus can only provide suboptimal performance.

Chapter 7

Utility Maximization with Rate Adaptation

In this chapter, we treat the problem of utility maximization when rate adaptation is applied. We introduce a more general model that can be applied to not only delay-constrained wireless networks but also a variety of other applications. We propose an on-line auction design that achieves the maximum total utility in the network as well as preventing selfish clients from gaining additional net utility by lying about their utility function. Hence, the auction design is both optimal and truthful.

7.1 System Model

Consider a wireless system with one server and N clients, numbered $\{1, 2, \ldots, N\}$. Time is divided into *time intervals*. Each client desires some service in each time interval. The service requirement within a time interval of each client is indivisible; that is, the server can only either fully meet the demand of a client or not serve it at all. At the beginning of each time interval, the server obtains the current channel condition. Both the demands of clients and the channel condition can be time-varying, and together we call them the *system state* in each time interval. The sever can learn the system state by either polling, probing, or estimating. Since these operations are costly and cannot be carried out too frequently, the server assumes that the system state does not change within an interval. Due to limited wireless resources, the server may be only able to serve some particular subsets of clients in each system state. To be more specific, we denote the system state in the k^{th} time interval by $c(k) \in C$, where C is a finite set, and $\{c(1), c(2), \ldots\}$ are i.i.d. random variables with $Prob\{c(k) = c\} =: p_c$. In practice, not only the system state but also the distribution of system states can be time-varying. However, the distribution of system states usually evolves on a much slower time scale compared to the length of a time interval and thus is assumed to be static.

A subset S of clients is said to be *schedulable* under system state c if it is possible for the server to serve all clients in S. For simplicity, we represent a system state c by the collection of subsets S that are schedulable under c. Thus, we have $S \in c$ if S is schedulable under c, and $S \notin c$ otherwise. Since the constraints of schedulable sets can be defined arbitrarily, this model can be applied to a wide range of applications. We will

illustrate some examples of applications in Section 7.2. In particular, it can accommodate per-packet delay constraints and rate adaptation.

The server is in charge of choosing a schedulable subset $S \in c(k)$ to serve in each time interval k. The server's choice is described by a *scheduling policy*.

Definition 15. Let $h(k)$ be the system's history up to the k^{th} time interval. A *scheduling policy* is a function $\eta : (h(k-1), c(k)) \mapsto 2^{\{1,2,\ldots,N\}}$, such that given history $h(k-1)$ and current system state $c(k)$, the server chooses the subset $\eta[h(k-1), c(k)] \in c(k)$ of clients to serve. All clients $n \in \eta[h(k-1), c(k)]$ are considered to be *served in the k^{th} time interval*.

As the system state is time-varying, it is less meaningful to discuss the performance of clients on a per-interval base. Rather, we measure the performance of a client through its average rate of being served. We define the *service rate* of a client n as follows:

Definition 16. Let $q_n(k)$ denote the *service rate* of client n up to the k^{th} time interval, defined by the recursion:

$$q_n(k+1) = \begin{cases} (1-\alpha_k)q_n(k) + \alpha_k, & \text{if client } n \text{ is served at the } k^{th} \text{ interval,} \\ (1-\alpha_k)q_n(k), & \text{otherwise,} \end{cases}$$

where $0 \leq \alpha_k \leq 1$, for all k. The *long-term service rate* of client n is defined as $q_n := \liminf_{k\to\infty} q_n(k)$.

In the above definition, α_k is a system-wide variable that is assumed to be the same for all clients. For example, by setting $\alpha_k \equiv \frac{1}{k}$, $q_n(k)$ becomes the proportion of time intervals that client n is being served. On the other hand, setting all α_k to be a constant makes $q_n(k)$ a weighted-average of service where recent service is more important than service a long time ago.

We further assume that each client n has an *utility function* $U_n(\cdot)$. Similar to those discussed in Chapter 6, the utility functions are strictly increasing, strictly concave, and infinitely differentiable. At the k^{th} time interval, client n receives utility that is equivalent to an amount $\frac{1}{\alpha_k}U_n(q_n(k))$ of money. The scaling factor $\frac{1}{\alpha_k}$ of the amount of money received by client n is set to equalize the effects of events in each interval. Section 7.4.1 provides a more detailed explanation of this setting. The *long-term utility* of client n is defined as $\liminf_{k\to\infty} U_n(q_n(k))$, which equals $U_n(q_n)$ since $U_n(\cdot)$ is continuous.

Finally, to enforce some form of fairness among clients, we also assume that each client n has a requirement of *minimum long-term service rate*, $\underline{q}_n$; that is, it requires $q_n \geq \underline{q}_n$ with probability 1. We assume that the minimum long-term service rate requirements are strictly feasible, that is, there exists some scheduling policy that ensures $q_n > \underline{q}_n$, for all n.

We are interested in maximizing the *total long-term utility* of the network, $\sum_{n=1}^{N} U_n(q_n)$. The utility maximization problem of this framework can hence be expressed as:

$$\text{Max} \sum_{n=1}^{N} U_n(q_n)$$

$$\text{s.t. Network dynamics and schedulability constraints,}$$

$$\text{and } q_n \geq \underline{q}_n, \forall n.$$

However, this formulation only considers the long-term behavior of the system. A solution to this utility maximization problem may not translate into an implementable scheduling policy, which would have to make decisions on a per-interval basis. Thus, we also wish to design *utility-optimal* scheduling policies.

Definition 17. A scheduling policy η is said to be *utility-optimal* if, by applying η, $\sum_{n=1}^{N} U_n(q_n(k))$ converges to the optimal value of the utility maximization problem almost surely as $k \to \infty$.

7.2 Examples of Applications

We will first discuss several applications that can be described by our framework.

7.2.1 Delay-Constrained Wireless Networks with Rate Adaptation

We first show that the model introduced in Chapter 2 is a special case of the one used in this chapter when rate adaptation is employed. Assume that there are N wireless clients and one access point (AP). Time is assumed to be slotted and divided into time intervals, each consisting of T consecutive time slots. At the beginning of each time interval, packets for each client arrive at the AP. Each client specifies a delay bound of τ_n time slots, with $\tau_n \leq T$. The packet for client n is to be delivered no later than the τ_n^{th} time slot in each time interval. Otherwise, the packet expires and is dropped from the system.

Due to channel fading, the link qualities between the AP and the client can be time-varying. We assume that the AP has full knowledge of the current channel state. The AP then applies rate adaptation for error-free transmissions. Thus, the transmission rates for different clients can be different, which in turn results in different transmission times. We define $t_{c,n}$ as the number of time slots required for an error-free transmission for client n under system state c. A scheduling policy is one which selects an ordered subset $S = \{s_1, s_2, \ldots, s_m\}$ of clients and transmits packets for clients in S according to the order. The ordered subset is considered schedulable under system state c if packets for all clients in S can be delivered before

their respective delay bounds, or, to be more specific, $\sum_{n=1}^{i} t_{c,s_n} \leq \tau_{s_i}$, for all $1 \leq i \leq m$. In this scenario, the service rate of each client reflects its *timely throughput*.

7.2.2 Mobile Cellular Network

Consider a mobile cellular network with a base station and N users. The system may have more than one channel, but each channel can be occupied by at most one user at any given time. We assume that time is slotted, where a *time slot* corresponds to a *time interval* in the system model. The length of a time slot is defined as the time needed for transmitting a packet plus any control overhead. Also, due to mobility, the link qualities between the base station and an user can be time-varying. We consider an ON/OFF model for links. The link between an user and the base station is considered ON if a packet can be transmitted between the two without errors, and considered OFF otherwise. We assume that the base station never transmits packets to users with OFF links. Thus, the system state at any time slot can be described as the set of users with ON links. A subset S of users is considered schedulable under some system state c if for any user $n \in S$, the link between user n and the base station is ON, and the size of S is smaller than or equal to the number of channels. A scheduling policy is one which chooses, based on current system state and past history, a schedulable subset of users and assigns channels to each of them. Finally, the service rate of each user is equal to its throughput.

7.2.3 Dynamic Spectrum Allocation

Consider a scenario with one primary user and N secondary users. The primary user holds licenses for several channels over a large geographical region. TV broadcasters are typical examples of primary users. The primary user only uses a portion of its licensed channels and is willing to allocate unused channels to secondary users. The secondary users are scattered throughout the region and constrained to much smaller transmission powers compared to the primary user, which makes spatial reuse possible. Still, some secondary users may interfere with each other and thus cannot be allocated the same channel simultaneously. We use a conflict graph $G = (V, E)$ to represent the interference relations between secondary users, where V is the set of secondary users and there is an edge between two users if they interfere with each other.

The primary user allocates unused channels periodically. Since the network activity of the primary user can be time-varying, the number of unused channels can also be time-varying. A scheduling policy is one which chooses disjoint subsets of secondary users for each unused channel, with the constraint that two users that are assigned the same channel cannot share a link in the conflict graph.

7.3 A General Method for Utility Maximization

In this section, we propose a general method for solving the utility maximization problem in time-varying wireless networks with minimum service requirements. We first show that the utility maximization problem can be formulated as a convex programming problem. Although the formulation requires explicit knowledge of the distribution of system states, i.e., the values of probability $[p_c]$, we will show the surprising result that there exists an on-line scheduling policy that does not need any information on the distribution of system states, and is, further, also utility-optimal. For simplicity, we assume that $\alpha_k := 1/k$, that is, $q_n(k)$ is the proportion of time intervals that client n has been served until the k^{th} time interval. We will discuss the case where α_k is a constant for all k at the end of this section.

7.3.1 Convex Programming Formulation

Define $p_c(k)$ and $f_{c,S}(k)$, for all system states c and subsets $S \in c$, recursively, as follows:

$$p_c(k+1) = \begin{cases} \frac{k-1}{k} p_c(k) + \frac{1}{k}, & \text{if } c(k) = c, \\ \\ \frac{k-1}{k} p_c(k), & \text{otherwise,} \end{cases}$$

and

$$f_{c,S}(k+1) = \begin{cases} \frac{k-1}{k} f_{c,S}(k) + \frac{1}{k}, & \text{if } c(k) = c \text{ and } S \text{ is scheduled at the } k^{th} \text{ interval,} \\ \\ \frac{k-1}{k} f_{c,S}(k), & \text{otherwise.} \end{cases}$$

These two variables can be thought of as the relative frequencies of occurrence of the system state c and the event that subset S is scheduled under system state c, respectively. Also, we have $\sum_{S \in c} f_{c,S}(k) = p_c(k)$ and $\sum_c \sum_{S: S \in c, n \in S} f_{c,S}(k) = q_n(k)$ for all c and k. For ease of discussion, we only consider scheduling policies where $f_{c,S} := \lim_{k \to \infty} f_{c,S}(k)$ exists for all system states c and subsets S. Thus, we have $\sum_{S \in c} f_{c,S} = p_c$ and $\sum_c \sum_{S: S \in c, n \in S} f_{c,S} = q_n$. The utility maximization problem can be described as the following convex

programming problem:

$$\text{Max} \sum_{n=1}^{N} U_n(q_n) = \sum_{n=1}^{N} U_n(\sum_{c} \sum_{S:S\in c, n\in S} f_{c,S})$$

$$\text{s.t.} \sum_{S\in c} f_{c,S} = p_c, \forall c,$$

$$q_n = \sum_{c} \sum_{S:S\in c, n\in S} f_{c,S} \geq \underline{q}_n, \forall n,$$

$$\text{over } f_{c,S} \geq 0.$$

While typical techniques for solving a convex programming problem can be applied to solve this utility maximization problem, such solutions may not be directly translatable into a scheduling policy for our time-varying network. Also, a solution based on solving the convex programming problem would require the knowledge of the probability distribution of system states. In practice, this knowledge may not always be available to the server. Thus, a scheduling policy that makes decisions based only on past history and current system state is needed.

7.3.2 An On-line Scheduling Policy

We now describe an on-line scheduling policy, and prove that it is utility-optimal. This scheduling policy only requires information on the past history and the current system state, and, surprisingly, does not need any knowledge of the actual probability distribution of system states. Thus, it is easily implementable. The scheduling policy is based on dual decomposition, which is similar to the approach used in Lin and Shroff [49], although they do not consider network dynamics.

We assign a Lagrange multiplier λ_n for each constraint
$\sum_c \sum_{S:S\in c, n\in S} f_{c,S} \geq \underline{q}_n$. The resulting Lagrangian of the resulting convex programming problem is:

$$L(f,\lambda) = \sum_{n=1}^{N} U_n(\sum_{c,S:S\in c, n\in S} f_{c,S}) + \sum_{n=1}^{N} \lambda_n(\sum_{c,S:S\in c, n\in S} f_{c,S} - \underline{q}_n),$$

where f denotes the vector consisting of $[f_{c,S}]$, for all c and S, and λ denotes the vector $[\lambda_n]$. The dual objective function is:

$$D(\lambda) = \max_{f: f_{c,S}\geq 0; \sum_{S\in c} f_{c,S} = p_c, \forall c} L(f,\lambda).$$

Since the minimum long-term service rate requirements, $[\underline{q}_n]$, are strictly feasible, there exist $[f_{c,S}]$ such

that

$$\sum_{S \in c} f_{c,S} = p_c, \text{ and } \sum_c \sum_{S: S \in c, n \in S} f_{c,S} > \underline{q}_n,$$

for all n. By Slater's condition, $\min_\lambda D(\lambda)$ equals the maximum total utility.

Let $\lambda(k) = [\lambda_n(k)]$ denote Lagrange multipliers that are used in the k^{th} period. The maximum total utility can be achieved by solving two subproblems: maximizing

$$\lim_{k \to \infty} E[L(f(k), \lambda)],$$

for any given λ, and minimizing

$$\lim_{k \to \infty} E[D(\lambda(k))].$$

We will refer to these two subproblems as the *primal problem* and *dual problem*, respectively.

We first discuss how to solve the primal problem. Due to the constraint $\sum_{S \in c} f_{c,S} = p_c$, $[f_{c,S}]$ is an optimal solution if and only if $\frac{\partial L}{\partial f_{c,S}} := \sum_{n \in S}(U_n'(q_n) + \lambda_n) = \max_{S' \in c} \frac{\partial L}{\partial f_{c,S'}}$ for every c and S such that $f_{c,S} > 0$. Recall that $U_n(\cdot)$ is strictly concave, and $U_n'(\cdot)$ is a strictly decreasing function. Suppose, at some time interval k with $c(k) = c$, there exists a subset S schedulable under c such that $\sum_{n \in S}(U_n'(q_n(k)) + \lambda_n) > \sum_{n \in S'}(U_n'(q_n(k)) + \lambda_n)$ for all other subsets S' schedulable under c. We wish to narrow the difference between S and all other S'. One obvious choice would be to schedule the subset S in the time interval, so as to increase $q_n(k+1)$ for all $n \in S$, and thus decrease $\sum_{n \in S}(U_n'(q_n(k)) + \lambda_n)$. In fact, as we shall see in the lemma below, selecting the schedulable subset S that maximizes $\sum_{n \in S}(U_n'(q_n(k)) + \lambda_n)$ also points in the steepest ascent direction of L.

Definition 18. Given λ and $f(k)$, a *max-weight scheduling policy* is one that schedules a schedulable subset $S \in c(k)$ that maximizes $\sum_{n \in S}(U_n'(q_n(k)) + \lambda_n)$ in each time interval k.

Lemma 10. *Let $\Delta f(k)$ be the vector consisting of the elements $\Delta f_{c,S}(k) := f_{c,S}(k+1) - f_{c,S}(k)$ for all c and S. Given λ and $f(k)$, the max-weight scheduling policy also maximizes $E[\nabla L(f, \lambda) \cdot \Delta f(k) | f_{c,S}(k)]$.*

Proof. Recall that we have:

$$f_{c,S}(k+1) = \begin{cases} \frac{k-1}{k} f_{c,S}(k) + \frac{1}{k}, & \text{if } c(k) = c \text{ and } S \text{ is scheduled at the } k^{th} \text{ interval,} \\ \\ \frac{k-1}{k} f_{c,S}(k), & \text{otherwise.} \end{cases}$$

Thus, $\Delta f_{c,S}(k) = \frac{1}{k}(1 - f_{c,S}(k))$ if $c(k) = c$ and S is scheduled, and $\Delta f_{c,S}(k) = -\frac{1}{k} f_{c,S}(k)$, otherwise. Let $\hat{f}_{c,S}(k)$ be the probability that $c(k) = c$ and S is scheduled under the max-weight scheduling policy. We then

have:

$$E[\nabla L(f,\lambda)\Delta f(k)|f_{c,S}(k)] = \sum_{c,S} E[\frac{\partial L}{\partial f_{c,S}}\Delta f_{c,S}(k)|f_{c,S}(k)] = \frac{1}{k}\{\sum_{c,S}[\hat{f}_{c,S}(k) - f_{c,S}(k)][\sum_{n\in S}(U_n'(q_n(k)) + \lambda_n)]\}. \tag{7.1}$$

Since $Prob\{c(k) = c\} = p_c$, $\sum_S \hat{f}_{c,S}(k) = p_c$. The term $E[\nabla L(f,\lambda)\Delta f(k)|f_{c,S}(k)]$ is maximized by setting:

$$\hat{f}_{c,S}(k) = \begin{cases} p_c, & \text{if } S = \arg\max_{S\in c} \sum_{n\in S} U_n'(q_n(k)) + \lambda_n, \\ 0, & \text{otherwise.} \end{cases} \tag{7.2}$$

This is achieved by selecting the schedulable subset S that maximizes $\sum_{n\in S}(U_n'(q_n(k))+\lambda_n)$ for every system state c. □

Next, we prove that the max-weight scheduling policy solves the primal problem.

Theorem 14. *Under the max-weight scheduling policy,*

$$L(f(k),\lambda) \to D(\lambda), \text{ as } k \to \infty,$$

for any given λ.

Proof. Since the utility functions are infinitely differentiable, $L(f,\lambda)$ is also infinitely differentiable. By Taylor's theorem, we have that for any f, Δf, and fixed λ,

$$L(f+\Delta f,\lambda) = L(f,\lambda) + \nabla L(f,\lambda)\Delta f + r(f,\Delta f,\lambda),$$

where $|r(f,\Delta f,\lambda)| < a(\lambda)|\Delta f|^2$, for some constant $a(\lambda)$. Now we have,

$$\begin{aligned} & E[L(f(k+1),\lambda)|f(k)] \\ \geq\ & L(f(k),\lambda) + E[\nabla L(f(k),\lambda)\Delta f(k) - a(\lambda)|\Delta f(k)|^2|f(k)] \\ \geq\ & L(f(k),\lambda) + E[\nabla L(f(k),\lambda)\Delta f(k)|f(k)] - \tilde{a}/k^2, \end{aligned} \tag{7.3}$$

where $\Delta f(k)$ is defined as in Lemma 10 and $\tilde{a}$ is some constant. The last inequality follows because $|\Delta f_{c,S}(k)| \leq \frac{1}{k}$ for all c, S.

Let $p_c(k) := \sum_{S\in c} f_{c,S}(k)$, which is the empirical frequency that system state c occurs, and let $\hat{f}_{c,S}(k)$ be defined as in the proof of Lemma 10. The values of $\hat{f}_{c,S}(k)$ under the max-weight scheduling policy are

given as in (7.2). Further, let $\hat{\mu}_c(k) := \max_{S \in c} \sum_{n \in S}(U_n'(q_n(k)) + \lambda_n)$, for all c. Using (7.1) and (7.2),

$$E[\nabla L(f(k), \lambda)\Delta f(k)|f(k)] \geq \frac{1}{k}\sum_c (p_c - p_c(k))\hat{\mu}_c(k).$$

Since $kp_c(k+1)$ is the number of occurrences of system state c until the k^{th} time interval, and the system state in each time interval is i.i.d. distributed, by the law of iterated logarithm [14], there exists some positive constant b such that $\limsup_{k\to\infty} \frac{k(p_c(k+1)-p_c)}{k^{1/2}(\log\log k)^{1/2}} \leq b$. Thus, for large enough k, there exists constant $\tilde{b}$ such that

$$E[\nabla L(f(k), \lambda)\Delta f(k)|f(k)] \geq -\frac{(\log\log k)^{1/2}}{k^{3/2}}\tilde{b}.$$

For large enough k, (7.3) can hence be bounded by

$$E[L(f(k+1), \lambda)|f(k)] \geq L(f(k), \lambda) - \frac{(\log\log k)^{1/2}}{k^{3/2}}\tilde{b} - \frac{\tilde{a}}{k^2}. \tag{7.4}$$

As we can see in the above, $E[L(f(k+1), \lambda)|f(k)]$ is "almost" larger than $L(f(k), \lambda)$ except for two diminishing terms. For large enough constant d, $-L(f, \lambda) + d$ is also nonnegative for all f, and by (7.4) it is therefore a "near positive submartingale" as in [57]. Since $\sum_{k=1}^{\infty}[\frac{(\log\log k)^{1/2}}{k^{3/2}}\tilde{b} + \frac{\tilde{a}}{k^2}] < \infty$, Exercise II-4 in [57] shows that $L(f(k), \lambda)$ converges almost surely.

Next, we need to show that $\lim_{k\to\infty} L(f(k), \lambda) = D(\lambda)$. We prove this by contradiction. Recall that the necessary and sufficient condition for $L(f, \lambda) = D(\lambda)$ is that $\sum_{n\in S}(U_n'(q_n) + \lambda_n) = \max_{S' \in c}\sum_{n\in S'}(U_n'(q_n) + \lambda_n)$, for all c, S such that $f_{c,S} > 0$. Suppose $L(f(k), \lambda)$ does not converge to $D(\lambda)$. Then, there exists $\delta > 0$, $\epsilon > 0$ such that for all large enough k, there exist (c_k, S_k) so that $f_{c_k,S_k} > \delta$ and $\sum_{n\in S_k}(U_n'(q_n) + \lambda_n) < \max_{S'\in c}\sum_{n\in S'}(U_n'(q_n) + \lambda_n) - \epsilon$. Evaluating the term $E[\nabla L(f(k), \lambda)\Delta f(k)|f(k)]$ under this condition shows that $E[\nabla L(f(k), \lambda)\Delta f(k)|f(k)] > \frac{1}{k}\delta\epsilon$. Since there exists some constant K such that for all $k > K$, $\frac{\tilde{a}}{k^2} < \frac{1}{k}\delta\epsilon/2$, we obtain $E[L(f(k+1), \lambda)|f(k)] > L(f(k), \lambda) + \frac{1}{k}\delta\epsilon - \frac{\tilde{a}}{k^2} > L(f(k), \lambda) + \frac{1}{k}\delta\epsilon/2$. Since $\sum_{k=1}^{\infty}\frac{1}{k} = \infty$, we also have

$$\lim_{k\to\infty} E[L(f(k), \lambda)] = \infty,$$

which is a contradiction. Thus, $\lim_{k\to\infty} L(f(k), \lambda) = D(\lambda)$. □

Next we discuss how to solve the dual problem: $\min_\lambda D(\lambda)$. We use the subgradient method to solve it. We first find a subgradient for $D(\lambda)$.

Lemma 11. *Let* $v_n := [\sum_{c,S:S\in c, n\in S} f_{c,S}^* - \underline{q}_n]$, *where* $[f_{c,S}^*]$ *maximizes* $L(f, \lambda)$. *Then* v *is a subgradient of* $D(\lambda)$.

Proof. Let λ' be an arbitrary vector. We have:

$$\begin{aligned} D(\lambda') = & \max_{f:\sum_{S\in c} f_{c,S}=p_c,\forall c} L(f,\lambda') \\ \geq & L(f^*,\lambda') = L(f^*,\lambda) + (\lambda'-\lambda)^{\mathrm{T}} v D(\lambda) = D(\lambda) + (\lambda'-\lambda)^{\mathrm{T}} v. \end{aligned}$$

Thus, v is a subgradient of $D(\lambda)$. □

The following theorem then follows from Theorem 8.9.2 in [5]:

Theorem 15. *Let $\{\beta_k\}$ be a sequence of nonnegative numbers with $\sum_{k=1}^{\infty} \beta_k = \infty$ and $\lim_{k\to\infty} \beta_k = 0$. Update $\lambda(k)$ by:*

$$\lambda_n(k+1) = \{\lambda_n(k) - \beta_k[\sum_{c,S:S\in c, n\in S} f^*_{c,S}(k) - \underline{q}_n]\}^+,$$

where $[f^*_{c,S}(k)]$ maximizes $L(f,\lambda(k))$. Then,

$$\lim_{k\to\infty} D(\lambda(k)) = \min_{\lambda\geq 0} D(\lambda).$$

In practice, the max-weight scheduling policy may converge slowly. Thus, the values of $\lambda(k)$ should be updated at a slower time scale only after the max-weight scheduling policy converges. As $q_n(k) = \sum_{c,S:S\in c,n\in S} f^*_{c,S}(k)$ when the max-weight scheduling policy converges, the subgradient method can be further simplified as

$$\lambda_n(k+1) = \{\lambda_n(k) - \beta_k[q_n(k) - \underline{q}_n]\}^+ \tag{7.5}$$

when updating $\lambda(k)$. In practice, we will update the values of $\lambda(k)$ infrequently. A scheduling policy that jointly applies the max-weight scheduling policy and updates $\lambda(k)$ according to the subgradient method is utility-optimal. This policy is an on-line policy in the sense that it schedules clients and updates $\lambda(k+1)$ only based on $[U'_n(q_n(k))]$, $[q_n(k)]$ and $\lambda(k)$.

We close this section by discussing the case where α_k is a constant for all k. Since all the discussions in this section are based on the assumption that $\alpha_k = \frac{1}{k}$, the scheduling policy that uses the max-weight scheduling policy while updating $\lambda(k)$ may no longer be utility optimal when α_k is a constant since the system will be too sensitive to events in the current interval. For example, consider the extreme case where $\alpha_k = 1$ for all k. The total utility of the system at interval k, $\sum_{n\in S} U_n(q_n(k))$, solely depends on events in interval k and thus will never converge under any scheduling policy. However, in practice, the constant value of α_k is usually small. Under such a case, the scheduling policy that jointly applies the max-weight scheduling policy and updates $\lambda(k)$ according to the subgradient method still offers good performance. In

Section 7.6, we will show by simulations that, when the constant value of α_k is around the order of 10^{-2}, the performance of the proposed policy is better than all other compared policies in all evaluated scenarios. We refer the reader to [43] for a a broader discussion of issues related to constant step sizes and vanishing step sizes and technical issues related to convergence analysis in the two cases.

7.4 A Truthful Auction Design

We have proposed an on-line scheduling policy and proved that it is utility-optimal in Section 7.3. However, this policy requires knowledge of the utility functions of all clients. In practice, utility functions may be known only to their clients. A strategic client may hence improve its own utility by faking its utility function. In this section, we will propose an auction design that prevents clients from benefiting by faking their utility functions. In this auction, clients offer their bids for service in each time interval. The server selects a subset of clients to serve and charges them based on their bids. The decision of selecting clients and charging them is derived from an auction design similar to the VCG auction. We prove that this design restricts clients from faking their utility functions. We also show that the auction design schedules the same clients as those scheduled by the on-line scheduling policy introduced in Section 7.3. Thus, this auction is not only truthful but also utility-optimal.

7.4.1 Basic Mechanism and Truthful Property

We first describe the procedure and terminology of an auction. We then propose a VCG-based auction. At the beginning of each time interval k, every client n announces a bid $b_n(k)$ to the server. Based on the bids $[b_n(k)]$, along with the past history of the system and the current system state, the server schedules a subset S in the time interval k and charges each client n an amount $e_n(k+1)$ of money. Thus, an auction design can be specified by its decisions on selecting clients to schedule and charging them.

Recall that client n receives an utility that is equivalent to an amount $\frac{1}{\alpha_{k+1}}U_n(q_n(k+1))$ of money. The factor $\frac{1}{\alpha_{k+1}}$ is defined so as to equalize the importance of events in each interval k because whether client n is scheduled in interval k influences $q_n(k+1)$ by α_k. The *net utility* of a client can be defined as $\frac{1}{\alpha_{k+1}}U_n(q_n(k+1)) - e_n(k+1)$. We can also define the *marginal utility* of client n from being scheduled in the k^{th} time interval as follows:

Definition 19. Let $q_n^+(k+1)$ and $q_n^-(k+1)$ be the service rates of client n if it is scheduled, and if it is not scheduled, in the k^{th} time interval, respectively. The *marginal utility* of client n in the k^{th} time interval is defined as $\frac{1}{\alpha_{k+1}}[U_n(q_n^+(k+1)) - U_n(q_n^-(k+1))]$.

Suppose the goal of every client is to selfishly maximize its own net utility $\frac{1}{\alpha_{k+1}}U_n(q_n(k+1)) - e_n(k+1)$, in each time interval k. An auction design is considered *truthful* if all clients bid their marginal utilities.

Definition 20. An auction design is *truthful* if choosing

$$b_n(k) = \frac{1}{\alpha_{k+1}}[U_n(q_n^+(k+1)) - U_n(q_n^-(k+1))]$$

yields the highest net utility for client n in time interval k.

Now we propose a VCG-based auction. The server assigns a non-negative *discount*, $d_n(k)$, to each client n in time interval k. The values of discounts are announced before clients offer their bids and are thus not influenced by the bids of clients. After gathering the bids from clients, the server then schedules a schedulable subset S that maximizes $\sum_{n\in S}(b_n(k) + d_n(k))$, with ties broken arbitrarily. The server does not charge anything to those clients that are not scheduled. For each scheduled client $m \in S$, the server charges it the minimum bid $e_m(k+1)$ it should have offered to be scheduled, specifically

$$e_m(k+1) = \max_{S':m\notin S'}[\sum_{n\in S'}(b_n(k) + d_n(k))] - \sum_{n\in S, n\neq m}(b_n(k) + d_n(k)) - d_m(k). \tag{7.6}$$

This auction mechanism is usually referred to as a *weighted VCG* mechanism. The proof that such an auction mechanism is truthful can be found in Section 9.5.3 of [59].

Theorem 16. *The proposed auction is truthful.*

7.4.2 Proof of Optimality

We now prove that the scheduling policy derived from the proposed auction design is consistent with the max-weight scheduling policy. Thus, this auction also achieves the maximum total long-term utility.

Theorem 17. *Let $d_n(k) \equiv \lambda_n(k)$. Assume all clients bid their marginal utility. Then, there exists $\epsilon > 0$ such that for all $\alpha_k < \epsilon$, a schedulable subset $S \in c(k)$ that maximizes $\sum_{n\in S}(b_n(k) + d_n(k))$ also maximizes $\sum_{n\in S}(U_n'(q_n(k)) + \lambda_n(k))$ over all schedulable subsets.*

Proof. Let $M := \max_{S\in c(k)}\sum_{n\in S}(U_n'(k) + \lambda_n(k))$, and let $\mathbb{S}_M$ be the collection of all schedulable subsets S with

$$\sum_{n\in S}(U_n'(k) + \lambda_n(k)) = M.$$

Also, let $M^- := \max_{S:S\in c(k),S\notin\mathbb{S}_M}\sum_{n\in S}(U_n'(k) + \lambda_n(k))$ and let $\delta := M - M^-$.

Recall that $q_n^+(k+1) = (1-\alpha_k)q_n(k) + \alpha_k$ and $q_n^-(k+1) = (1-\alpha_k)q_n(k)$. Since utility functions are infinitely differentiable, by using Taylor's series,

$$b_n(k) = \frac{1}{\alpha_k}[U_n(q_n^+(k+1)) - U_n(q_n^-(k+1))] = U_n'((1-\alpha_k)q_n(k)) + O(\alpha_k).$$

There exists some ϵ such that for all $\alpha_k < \epsilon$,

$$|b_n(k) - U_n'(q_n(k))| < \delta/2N.$$

Now we have that $|\sum_{n\in S}(b_n(k)+d_n(k)) - \sum_{n\in S}(U_n'(k)+\lambda_n(k))| < \frac{\delta}{2}$ for all subsets S. Thus, for all schedulable subsets $S \in \mathbb{S}_M$ and $S' \notin \mathbb{S}_M$,

$$\sum_{n\in S}(b_n(k)+d_n(k)) > \sum_{n\in S'}(b_n(k)+d_n(k)).$$

Therefore, a schedulable subset that maximizes $\sum_{n\in S}(b_n(k)+d_n(k))$ also maximizes $\sum_{n\in S}(U_n'(q_n(k)) + \lambda_n(k))$ □

We have proved that the max-weight scheduling policy and the auction mechanism schedule the same set of clients if $\alpha_k < \epsilon$. Thus, when α_k is $\frac{1}{k}$, since $\alpha_k \to 0$ as $k \to \infty$, the two policies will converge eventually. For the case where α_k is set to a constant for all k, the behavior of the auction mechanism is still approximately the same as that of the max-weight policy if the constant value of α_k is small. We again refer the reader to [43] for a discussion of such issues. In Section 7.6, we will show by simulation that when the constant value of α_k is around the order of 10^{-2}, the performances of the two policies are indistinguishable.

In Section 7.3.2, we have shown that by applying the max-weight scheduling policy and updating $[\lambda_n(k)]$ according to (7.5), the total long-term utility is maximized. Thus, we can also maximize the total long-term utility by applying the proposed auction and setting discounts $[d_n(k)] \equiv [\lambda_n(k)]$.

7.4.3 Remarks on Implementation

In practice, we implement a protocol that jointly applies the proposed auction design and updates discounts by $[d_n(k)] = [\lambda_n(k)]$. To reduce overhead, all clients announce their utility functions to the server upon joining the system. In each time interval, the server computes their bids as

$$b_n(k) = \frac{1}{\alpha_{k+1}}[U_n(q_n^+(k+1)) - U_n(q_n^-(k+1))].$$

The server then schedules the schedulable subset $S \in c(k)$ that maximizes $\sum_{n\in S}[b_n(k) + d_n(k)]$ and charges all clients $n \in S$ according to (7.6). As a consequence of Theorem 16, a client that aims to greedily maximize its own net utility in each time interval k would not benefit by lying about its utility function. Also, this protocol is utility-optimal.

In addition to being truthful and utility-optimal, this protocol also provides incentives for servers to offer service by charging clients. For example, consider the scenario where a TV broadcast company holds several licenses for channels. Even though the company may not fully utilize its channels, it would not be willing to allocate unused bandwidth to unaffiliated users unless doing so can increase its own revenue. Thus, generating revenues for the server is also an important property for a protocol.

7.5 Algorithms for Specific Applications

We have shown that the protocol described in Section 7.4.3 is both truthful and utility-optimal. Given the bids $[b_n(k)]$ from clients, the protocol needs to select a subset $S \in c(k)$ that maximizes $\sum_{n\in S}[b_n(k) + d_n(k)]$ and charge them. In this section, we discuss how to design algorithms for scheduling and charging explicitly for each of the three applications discussed in Section 7.2.

7.5.1 Delay-Constrained Wireless Networks with Rate Adaptation

We consider first the scenario described in Section 7.2.1. There are N wireless clients and one AP. Each time interval contains T time slots. At the beginning of each time interval, there is one arrived packet at the AP for each client n. The packet for client n is to be delivered before the τ_n^{th} time slot, with $\tau_n \leq T$, or else the packet expires and is dropped from the system. The transmission time for client n under system state c is $t_{c,n}$.

A schedulable subset $S \in c$ can be described as an ordered subset $S = \{s_1, s_2, \ldots, s_m\}$, such that transmitting packets for clients in S according to the order will meet their respective delay bounds; that is, $\sum_{j=1}^{i} t_{c,s_j} \leq \tau_{s_i}$, for all $1 \leq j \leq m$. To find the schedulable subset $S \in c(k)$ that maximizes $\sum_{n\in S}(b_n(k) + d_n(k))$ in the k^{th} time interval, we can assign a value $v_n(k) := b_n(k) + d_n(k)$ to each client k. Now, if we have $\tau_n \equiv T$, for all n, then this is simply a knapsack problem. To deal with the case when different clients may require different delay bounds, we note that for any schedulable ordered subset S, reordering clients in S in ascending order of their delay bounds, i.e., serving clients in an "earliest deadline first" fashion, is also schedulable. Thus, an analog to the modified knapsack algorithm described in Section 5.3 finds a schedulable subset $S \in c(k)$ that maximizes $\sum_{n\in S}(b_n(k) + d_n(k))$ in the k^{th} time interval. The complete algorithm is

shown in Algorithm 4. The complexity of this algorithm is $O(NT)$. To compute the charge for a client $n \in S$, we need to determine $\max_{S':S' \in c(k), n \notin S'}(b_n(k) + d_n(k))$, which can be obtained by eliminating client n and rerunning Algorithm 4. Thus, the complexity of computing charges for all scheduled clients is $O(N^2T)$.

Algorithm 4 Delay-Constrained Networks

for $n = 1$ to N **do**
  $v_n(k) \leftarrow b_n(k) + d_n(k)$
Sort clients such that $\tau_1 \leq \tau_2 \leq \cdots \leq \tau_N$
$S[0,0] \leftarrow \phi$
$M[0,0] \leftarrow 0$
for $n = 1$ to N **do**
  for $t = 1$ to T **do**
    if $t > \tau_n$ **then**
      $M[n,t] \leftarrow M[n,t-1]$
      $S[n,t] \leftarrow S[n,t-1]$
    else if $v_n(k) + M[n-1, t - t_{c(k),n}] > M[n-1,t]$ **then**
      $M[n,t] \leftarrow v_n(k) + M[n-1, t - t_{c(k),n}]$
      $S[n,t] \leftarrow S[n-1, t - t_{c(k),n}] + \{n\}$
    else
      $M[n,t] \leftarrow M[n-1,t]$
      $S[n,t] = S[n-1,t]$
$\max_{S:S \in c(k)} \sum_{n \in S}(b_n(k) + d_n(k)) \leftarrow M(N,T)$
schedule according to $S[N,T]$

7.5.2 Mobile Cellular Networks

Consider the scenario described in Section 7.2.2 with one base station holding $\mathbb{C}$ channels and N mobile users. The links between the users and the base station can either be ON or OFF, and the base station can only schedule transmissions to users with ON links. Recall that a subset S is schedulable under system state c if every client n in S has an ON link, and the size of S is smaller or equal to the number of channels, $\mathbb{C}$. Thus, to implement the protocol, we can simply sort all clients with ON links in descending order of $b_n(k) + d_n(k)$, and schedule the first $\mathbb{C}$ clients. Also, let client m be the $(\mathbb{C}+1)^{th}$ client with an ON link. To be scheduled, a scheduled client n would have to outbid client m, that is, $b_n(k) + d_n(k) \geq b_m(k) + d_m(k)$. Thus, the price paid by each scheduled client n is $b_m(k) + d_m(k) - d_n(k)$.

7.5.3 Dynamic Spectrum Allocation

Finally, we discuss the scenario of dynamic spectrum allocation. Suppose there is a primary user holding licenses to several channels , and there are N secondary users. The interference relations between secondary users are represented by a conflict graph $G = \{V, E\}$, where V is the set of all secondary users, and two users correspond to an edge in G if they interfere with each other. Suppose the primary user is going to allocate

$\mathbb{C}(k)$ unused channels in the k^{th} time interval. A schedulable subset of secondary users can be represented as a coloring by $\mathbb{C}(k)$ colors on some nodes in V, with the restriction that any two colored nodes with the same color cannot share an edge in G. To find a subset $S \in c(k)$ with the maximum $\sum_{n \in S}(b_n(k) + d_n(k))$, we can associate a value $v(k) = b_n(k) + d_n(k)$ to each node in V and find the maximum-weight coloring with $\mathbb{C}(k)$ colors. In particular, when $\mathbb{C}(k) = 1$, this is equivalent to finding the maximum-weight independent set. While finding the maximum-weight independent set is NP-hard, there exist heuristics. Also, for a mid-sized network with about 20 secondary users, the computational overhead for finding an optimal schedule is reasonably small.

7.6 Simulation Results

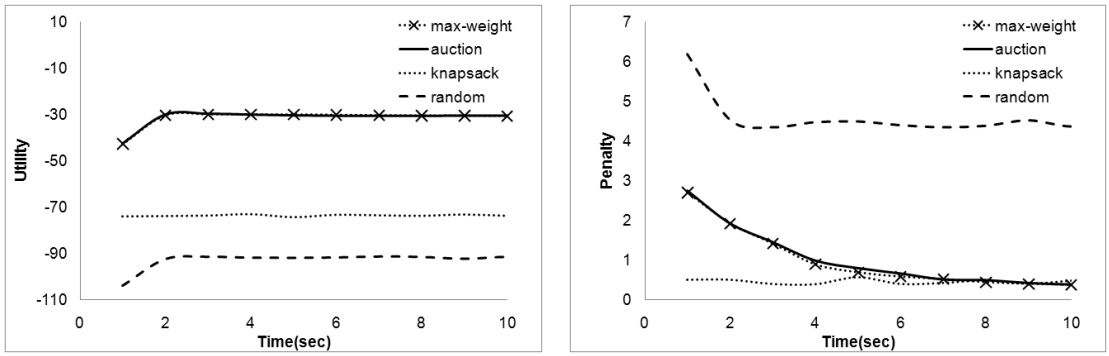
We now present simulation results for the three discussed applications: delay-constrained wireless networks with rate adaptation, mobile cellular networks, and dynamic spectrum allocation. In each of the three applications, we evaluate the max-weight scheduling policy, the VCG-based auction mechanism, and a policy that randomly orders all clients and greedily schedules a maximal schedulable subset in each time interval. In addition, we also compare our policies against a state-of-art policy in each of the three applications.

In all applications, the utility function of client n is set to be $U_n(q_n) := w_n \frac{q_n^{a_n} - 1}{a_n}$, where w_n is a positive integer and $a_n \in (0, 1)$. We set α_k to a small positive constant α, for all k. We update $\lambda_n(k)$ and $d_n(k)$ every T time intervals. To ensure our policies nearly converge within T time intervals, the value of T is chosen so that $(1 - \alpha)^T$ is small. When updating $\lambda_n(k)$ and $d_n(k)$, we also set β_k to a constant β. The choice of β involves a tradeoff between two factors. On the one hand, if β is too small, it takes a long time for $\lambda_n(k)$ and $d_n(k)$ to be incremented to a large enough value. On the other hand, if β is too big, then $\lambda_n(k)$ and $d_n(k)$ may dominate the max-weight scheduling policy and the VCG-based auction mechanism. As a result, $\lambda_n(k)$ and $d_n(k)$ may fluctuate instead of converging to an optimal value. In practice, we choose β so that $\beta \underline{q}_n$ is about one-tenth the marginal utilities of clients.

In all simulations, we evaluate two metrics: the total utility of all clients, and the total penalty, $\sum_n [\underline{q}_n - q_n(k)]^+$, defined as the sum of shortfalls between the required throughputs and the actual throughputs of clients. All results presented are the average over 20 runs.

7.6.1 Delay-Constrained Wireless Networks with Rate Adaptation

In this scenario, we compare our policies against the modified-knapsack policy proposed in 5.3, which is feasibility-optimal in the sense that it satisfies the minimum service requirements for any sets of clients as



(a) Utility (b) Penalty

Figure 7.1: Delay-Constrained Networks

long as they are feasible.

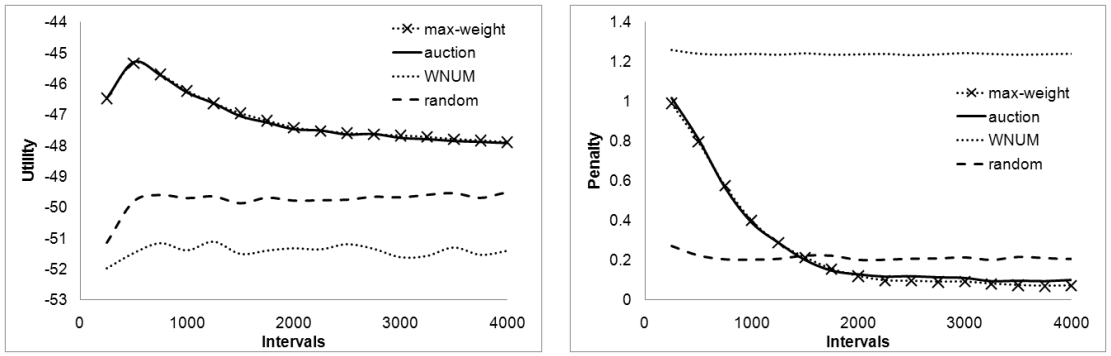
We use a similar set-up as the one in 5.5 for this scenario. Suppose that there are 45 clients that generate VoIP traffic. Each client generates one packet of size 160 Bytes every 20 ms, which is the length of a time interval. This results in a 64 kbit/sec rate. The highest data rate that can be supported by the link between the AP and client n alternates between 11 Mb/sec and 5.5 Mb/sec. The times needed for a transmission under the two data rates, including header and ACK, are 480 μs and 610 μs, respectively. Thus, by setting the duration of a time slot to 160 μs, a transmission takes 3 time slots under 11 Mb/sec, and 4 time slots under 5.5 Mb/sec. A time interval contains 125 time slots.

The utility function of client n is defined by $w_n = 3 + (n \mod 3)$ and $a_n = 0.05 + 2n$. The minimum service requirement of client n is $0.5 + 0.01(20n \mod 300)$ packets per interval. The delay bound of client n equals the length of a time interval if n is odd, and equals two-third of the length of a time interval if n is even. Further, we set $\alpha = 0.05$, $\beta = 1$, and $T = 50$.

Simulation results are shown in Figure 7.1. The max-weight scheduling policy and the VCG-based auction mechanism achieve the highest utility and near-zero penalty. The penalties resulting from the two policies converge to zero slower than the modified-knapsack policy. This is because it takes some time for the two policies to build up their discounts. On the other hand, the modified-knapsack policy has worse utility than the two proposed policies because it only concerns itself with satisfying the minimum service requirements of clients and does not consider utilities.

7.6.2 Mobile Cellular Networks

We consider a system with one base station with three non-interfering channels, and 20 users. The utility function of the n^{th} client is defined by $w_n = 1 + (n \mod 3)$ and $a_n = 0.2 + 0.1(n \mod 7)$. A time interval is the time for one packet transmission. The minimum service requirement of client n is set to be $\underline{q}_n = 0.05(n$



(a) Utility (b) Penalty

Figure 7.2: Mobile Cellular Networks

$\mod 5$) packets per interval. The probability that the link between client n and the base station is ON is $0.6 + 0.02(n \mod 10)$. Finally, we set $\alpha = 0.01$, $T = 250$, and $\beta = 10$.

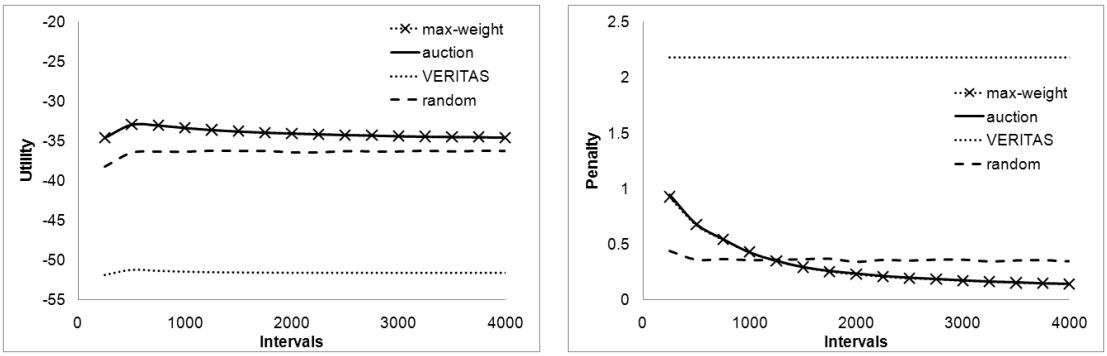
In addition to the three policies discussed above, we also evaluate the WNUM policy proposed in [61], which considers and maximizes utilities on a per-interval base. Simulation results are shown in Figure 7.2. The max-weight scheduling policy and the VCG-based auction mechanism both achieve the highest utility and near-zero penalty, suggesting that they not only satisfy the minimum service requirements of clients but are also utility-optimal. On the other hand, the performance of the WNUM policy is even worse than the random policy. This shows that, as far as long-term average performance is concerned, a policy that optimizes total utility based on per-interval performance is not adequate.

7.6.3 Dynamic Spectrum Allocation

Finally, we present the simulation results for dynamic spectrum allocation. We compare our policies against VERITAS as proposed in [81]. VERITAS is developed for the scenario where the primary only makes allocation decisions once, or at most very infrequently.

We use a similar setting as that in [81]. We assume there are 20 secondary users, randomly spaced in a 1×1 square. Two users interfere with each other if their distance is smaller than 0.3. The average node degrees of the resulting conflict graph in each simulation range from 2.9 to 6.5. We assume that the primary user always has a channel to allocate. The utility function and minimum service requirement of client n are given by $w_n = 1 + (n \mod 3)$, $a_n = 0.2 + 0.1(n \mod 7)$, and $\underline{q}_n = 0.05(n \mod 8)$. Finally, we set $\alpha = 0.01$, $\beta = 5$, and $T = 250$.

Simulation results are shown in Figure 7.3. As in the previous simulations, both the max-weight scheduling policy and the VCG-based auction have the highest utilities and near-zero penalties. Also, it can be shown that VERITAS has poor performance under this setting. This suggests that a protocol developed for spectrum



(a) Utility

(b) Penalty

Figure 7.3: Dynamic Spectrum Allocation

allocation in a one-shot fashion is not applicable to scenarios where the primary user allocates spectrum frequently.

Chapter 8

Broadcasting and Network Coding

In this chapter, we consider the problem of broadcasting real-time flows over unreliable wireless links. We extend the model introduced in Chapter 2, which only considers unicast flows, to address additional challenges introduces by broadcasting. This model also considers the optional usage of various network coding mechanisms. Based on this model, we propose a framework for designing scheduling policies under different coding mechanisms. We derive scheduling policies for three different systems, one without network coding, one employs XOR coding, and the other employs linear coding.

8.1 System Model

The model introduced in this section is similar to the one in Chapter 2. Two major differences distinguish the two models. First, in the scenario of broadcasting real-time flows, there is usually more than one client that requires packets from the same flow. Second, as ACKs are not implemented for broadcast, and it is costly to obtain feedback information from all clients, an AP that broadcasts several real-time flows does not have per-transmission feedback from clients. For completeness, we formally introduce the model in the sequel.

We consider a wireless system where there is one basestation broadcasting several flows with delay constraints to a number of wireless clients. We denote by $\mathbb{I}$ the set of flows, and by $\mathbb{N}$ the set of clients. We assume that time is slotted and numbered as $\tau \in \{0, 1, 2, \dots\}$. The basestation can make exactly one transmission in a time slot and the duration of a time slot is hence set to be the time needed for broadcasting one packet. Time slots are divided into *intervals* where each interval consists of the T consecutive time slots in $[kT, (k+1)T)$, for all $k \geq 0$. According to its specific traffic pattern, each flow may generate at most one packet at the beginning of each interval. We model the traffic patterns of flows as an irreducible Markov chain with finite states and assume that, in the steady state, the probability that the subset $S \subseteq \mathbb{I}$ of flows generates packets is $R(S)$. Each packet generated by any flow has a delay constraint of T time slots; that is, it needs to be delivered to its client within the same interval that it is generated. At the end of each interval, packets generated at the beginning of that interval expire, and are dropped from the system. Thus, the delay

undergone by successfully delivered packet is at most T time slots.

We consider heterogeneous and unreliable wireless links. We assume that the reliability of the channel between the basestation and client n is p_n. When the basestation broadcasts a packet, client n receives that packet correctly with probability p_n, and the packet is corrupted due to channel unreliability with probability $1 - p_n$. Since the overhead of gathering feedbacks from clients is large for broadcast, and ACKs are not implemented for broadcast in most mechanisms, we assume that the basestation has no knowledge of whether clients receive the packet correctly after each transmission. The lack of feedback information is one of the most important characteristics that distinguishes this model from that used in Chapter 2 for unicast flows. While it is infeasible for the basestation to gather feedbacks from clients on a per-transmission basis, the basestation can still obtain feedback infrequently. Such infrequent feedback is used to estimate the channel condition for each client, rather than to acknowledge the reception of a packet. Thus, we assume that the basestation has knowledge of the channel reliabilities p_n for all $n \in \mathbb{N}$. Since wireless links are unreliable, the basestation may need to broadcast the same packet more than once in an interval to increase the probability of delivery, and thus it is possible that a client receives duplicate packets. In such a case, the duplicate packets are dropped by the client.

Similar to Chapter 2, the performance of client n on flow i is measured by the long-term average number of packets of flow i received by client n per interval, that is, the *timely throughput* of client n on flow i. Further, we assume that each client n has a specified timely throughput requirement $q_{i,n}$ for each flow i. The system is considered *fulfilled* if the long-term average number of packets from every flow $i \in \mathbb{I}$ received by every client $n \in \mathbb{N}$, excluding duplicate packets, per interval, is at least $q_{i,n}$. Since the steady-state probability that flow i generates a packet in an interval is $\sum_{S:i\in S} R(S)$, the timely throughput requirement $q_{i,n}$ is equivalent to requiring that at least a fraction $q_{i,n}/\sum_{S:i\in S} R(S)$ of the packets generated by flow i are delivered to client n.

The goal of this chapter is to design *feasibility-optimal* broadcast policies which fulfill all *strictly feasible* systems. Since the set of broadcast policies depends on the coding mechanism used by the system, we also need to define the concept of *schedule space*.

Definition 21. A *schedule space* of a coding mechanism is the collection of all schedules for an interval. It consists of, for each $S \subseteq I$, the decision of what packet to transmit in each of time slots within the interval, that can be carried out by the coding mechanism, given that only flows in S generate a packet at the beginning of the interval.

Definition 22. A *broadcast policy* is one that, based on past system history and packet generations in the current interval, assigns a schedule, possibly at random, from the schedule space of its employed coding

mechanism.

In this paper, we consider three schedule spaces, one that only transmits raw packets without coding, one that employs XOR coding, and one that employs linear coding.

Definition 23. A system is strictly feasible for a schedule space if there exists a positive number $\epsilon > 0$, and a broadcast policy under the schedule space of its coding mechanism, that fulfills the same system with timely throughput requirement $[(1+\epsilon)q_{i,n}]$.

Definition 24. A broadcast policy is a *feasibility-optimal* policy under the schedule space of some coding mechanism if it fulfills all systems that are strictly feasible under the use of that coding mechanism.

8.2 A Framework for Designing Feasibility-optimal Policies

We now introduce a framework for designing feasibility-optimal policies under any schedule space. Since the basestation does not have feedback information from clients, it cannot know the actual timely throughput received by each client for each flow. However, it can estimate it. Let $q^*_{i,n}(k)$ be the indicator function that client n actually receives the packet from flow i in the interval $[kT, (k+1)T)$ under some policy η. Note that $q^*_{i,n}(k)$ is a random variable whose value is not known to the server. Let $\hat{q}_{i,n}(k) := \mathrm{E}[q^*_{i,n}(k)|\mathcal{H}_{kT}]$, where $\mathcal{H}_{kT}$ is the history of all packet arrivals of all the flows up to and including time kT, with the conditional expectation taken under the broadcast policy used by the basestation. Since the probability of successful reception of a packet by a client depends only on the number of times that a packet is broadcast and conditionally independent of everything else, it follows that $\hat{q}_{i,n}(k)$ is the conditional probability estimate made by the basestation, of whether a packet of flow i is successfully delivered to client n in that interval, based on its actions in that interval. We will denote this aforesaid set of actions of the basestation by $\mathcal{A}_{kT}$. So $\hat{q}_{i,n}(k) := \mathrm{E}[\, q^*_{i,n}(k)|\mathcal{H}_{kT}]$ $= \mathrm{E}[\, q^*_{i,n}(k)|\mathcal{A}_{kT}]$. The random variables $(\hat{q}_{i,n}(k) - q^*_{i,n}(k))$ are bounded and $E[\hat{q}_{i,n}(k) - q^*_{i,n}(k)|\mathcal{H}_{kT}] = 0$, for all i, n, and k. Define $q^*_{i,n} := \liminf_{K\to\infty} \frac{\sum_{k=0}^{K-1} q^*_{i,n}(k)}{K}$ and $\hat{q}_{i,n} := \liminf_{K\to\infty} \frac{\sum_{k=0}^{K-1} \hat{q}_{i,n}(k)}{K}$. The former is the actual long-term timely throughput of client n on flow i, while the latter is the asymptotic estimate made by the basestation. We then have $\hat{q}_{i,n} = q^*_{i,n}$ almost surely, by the martingale stability theorem [52, Theorem B, page 458], and thus a system is fulfilled if and only if $\hat{q}_{i,n} \geq q_{i,n}$ for all i and n.

We define the *expected delivery debt* for each client n and flow i as $d_{i,n}(k) := \sum_{j=0}^{k-1}(q_{i,n} - \hat{q}_{i,n}(j))$, that is, the difference between the number of packets of flow i that should have been delivered to client n to fulfill its timely throughput requirement, and the expected number of packets of flow i that are delivered to client n, up to time kT, as estimated by the basestation. Denote by $D(k)$ the vector consisting of all the expected delivery debts $[d_{i,n}(k)]$. We then have

Lemma 12. *A system is fulfilled by a policy η if, under η, $\limsup_{k\to\infty}(\frac{d_{i,n}(k)}{k})^+ = 0$, for all $i \in \mathbb{I}$ and $n \in \mathbb{N}$, where we define $x^+ := \max\{x, 0\}$.*

We now provide a sufficient condition for a policy to be feasibility-optimal, similar to that used in the proof of Theorem 8, and it is also based on the following Theorem 7.

Theorem 18. *Let S_k be the set of flows that generate packets in the interval $[kT, (k+1)T)$. A basestation policy η^0 that maximizes*

$$\sum_{i\in\mathbb{I},n\in\mathbb{N}} d_{i,n}(k)^+\hat{q}_{i,n}(k) \tag{8.1}$$

for all k, among all policies under its schedule space, is feasibility-optimal for its schedule space.

Proof. Consider a strictly feasible system with timely throughput requirements $[q_{i,n}]$. There exists a positive number ϵ and a stationary randomized scheduling policy η', which chooses a schedule randomly from the schedule space, based on the packet arrivals at the beginning of this interval and independent of the system history before this interval, that fulfills the same system with timely throughput requirements $[(1+\epsilon)q_{i,n}]$. Since we model packet generations in each interval as an irreducible finite-state Markov chain and η' is a stationary randomized policy, there exists a large enough positive number M such that the expected average timely throughputs under η' in any M consecutive intervals is larger than $(1+\frac{\epsilon}{2})q_{i,n}$, i.e.,

$$E[\frac{\sum_{l=k}^{k+M-1}\hat{q}_{i,n}(l)}{M}|S_k, D(k)] > (1+\frac{\epsilon}{2})q_{i,n}, \tag{8.2}$$

for all i, n, S_k, and $D(k)$.

Define $L(t) := \frac{1}{2}\sum_{i\in\mathbb{I},n\in\mathbb{N}}(d_{i,n}(tM)^+)^2$. We then have

$$L(t+1) = \frac{1}{2}\sum_{i\in\mathbb{I},n\in\mathbb{N}}[(d_{i,n}(tM) + Mq_{i,n} - \sum_{k=tM}^{(t+1)M-1}\hat{q}_{i,n}(k))^+]^2,$$

and

$$\begin{aligned}
&E[L(t+1) - L(t)|\mathcal{H}_{tM}] = E[L(t+1) - L(t)|S_{tM}, D(tM)] \\
\leq &E[\sum_{i\in\mathbb{I},n\in\mathbb{N}}(Mq_{i,n} - \sum_{k=tM}^{(t+1)M-1}\hat{q}_{i,n}(k))d_{i,n}(tM)^+ + B_0|S_{tM}, D(tM)] &(8.3)\\
= &E\{\sum_{k=tM}^{(t+1)M-1}E[\sum_{i,n}d_{i,n}(k)^+(q_{i,n} - \hat{q}_{i,n}(k))|S_k, D(k)] + B_1(\eta)|S_{tM}, D(tM)\}, &(8.4)
\end{aligned}$$

where B_0 is a positive constant and $B_1(\eta)$ is bounded by $|B_1(\eta)| < B_2$, for some $B_2 > 0$, regardless of S_{tM},

$D(tM)$, and the employed policy η, because $|d_{i,n}(k) - d_{i,n}(tM)| \leq M$, for all $k \in [tM, (t+1)M)$, and all $i \in \mathbb{I}$, $n \in \mathbb{N}$.

Let $\hat{q}^0_{i,n}(k)$ and $\hat{q}'_{i,n}(k)$ be the values of $\hat{q}_{i,n}(k)$ under the policies η^0 and η', respectively. Since η^0 maximizes

$$\sum_{i\in\mathbb{I},n\in\mathbb{N}} d_{i,n}(k)^+ \hat{q}_{i,n}(k),$$

we have that, under η^0,

$$\begin{aligned}
&E[L(t+1) - L(t)|S_{tM}, D(tM)] \\
\leq &E\{\sum_{k=tM}^{(t+1)M-1} E[\sum_{i,n} d_{i,n}(k)^+(q_{i,n} - \hat{q}^0_{i,n}(k))|S_k, D(k)] + B_1(\eta^0)|S_{tM}, D(tM)\} && \text{(by (8.4))} \\
\leq &E\{\sum_{k=tM}^{(t+1)M-1} E[\sum_{i,n} d_{i,n}(k)^+(q_{i,n} - \hat{q}'_{i,n}(k))|S_k, D(k)] + B_1(\eta^0)|S_{tM}, D(tM)\} \\
\leq &E[M\sum_{i\in\mathbb{I},n\in\mathbb{N}} (q_{i,n} - \frac{\sum_{k=tM}^{(t+1)M-1}\hat{q}'_{i,n}(k)}{M})d_{i,n}(tM)^+ + B_1(\eta^0) + B_0 - B_1(\eta')|S_{tM}, D(tM)] && \text{(by (8.4))} \\
< &-\frac{M\epsilon q^*}{2}\sum_{i\in\mathbb{I},n\in\mathbb{N}} d_{i,n}(tM)^+ + B,
\end{aligned}$$

where $q^* := \min_{i,n:q_{i,n}>0} q_{i,n}$ and $B := 2B_2 + B_0$. The last inequality follows from (8.2).

By Theorem 7, we have that

$$\limsup_{K\to\infty} \frac{1}{K}\sum_{t=0}^{K-1} E\{\sum_{i\in\mathbb{I},n\in\mathbb{N}} d_{i,n}(tM)^+\} \leq \frac{2B}{M\epsilon q^*}.$$

Lemma 9 shows that $\limsup_{k\to\infty}(\frac{d_{i,n}(k)}{k})^+ = 0$ and the system is also fulfilled by the policy η^0. Thus, η^0 is feasibility-optimal. □

Theorem 18 provides an avenue for designing scheduling policies. For any system and any coding mechanism, we can design a policy that aims to maximize (8.1). Such a policy is feasibility-optimal.

8.3 Scheduling without Network Coding

Next, we consider three different kinds of coding mechanisms and show how Theorem 1 suggests tractable scheduling policies. We first consider a system where network coding is not employed. In each time slot, the basestation can only broadcast a raw packet from a flow that has generated one packet in the interval.

Suppose some subset of flows S_k have generated packets at the beginning of the interval $[kT, (k+1)T)$

and that the packet from flow i, $i \in S_k$, is transmitted t_i times within the interval. The probability that client n receives the packet from flow i in this interval is then $\hat{q}_{i,n}(k) = 1 - (1 - p_n)^{t_i}$. Since the basestation can make T broadcasts in an interval, we have $\sum_{i \in S_k} t_i \leq T$. We can then formulate the condition in Theorem 18 as an integer programming problem:

$$\begin{aligned} \max & \sum_{i \in S_k} \sum_{n} d_{i,n}(k)^+ [1 - (1 - p_n)^{t_i}] \\ \text{s.t.} & \sum_{i \in S_k} t_i \leq T, \\ & t_i \geq 0, \forall i \in S_k. \end{aligned}$$

We show that there exists a polynomial time algorithm that solves the integer programming problem. Suppose that, at some time in an interval, the packet of flow i has been broadcast $t_i - 1$ times. The probability that client n has not received the packet from flow i during the first $t_i - 1$ transmissions, and receives this packet when the basestation broadcasts the packet from flow i for the t_i^{th} time, is $p_n(1 - p_n)^{t_i - 1}$. Thus, we can define the *weighted marginal delivery probability* of the t_i^{th} transmission of flow i as

$$m_i(t_i) := \sum_{n \in \mathbb{N}} d_{i,n}(k)^+ p_n (1 - p_n)^{t_i - 1}.$$

We now propose an online scheduling algorithm, which we call the Greedy Algorithm, as shown in Algorithm 5. In Step 7 of the algorithm, we break ties randomly. We also show that this algorithm is feasibility-optimal.

Algorithm 5 Greedy Algorithm

1: Number flows as $1, 2, \ldots, |\mathbb{I}|$
2: **for** $i = 1$ to $|\mathbb{I}|$ **do**
3: $\quad t_i \leftarrow 1$
4: $\quad m_i \leftarrow \sum_{n \in \mathbb{N}} d_{i,n}(k)^+ p_n$
5: **for** $\tau = 1$ to T **do**
6: $\quad i \leftarrow \arg\max_{j \in S_k} m_j$
7: $\quad t_i \leftarrow t_i + 1$
8: $\quad m_i \leftarrow \sum_{n \in \mathbb{N}} d_{i,n}(k)^+ p_n (1 - p_n)^{t_i - 1}$
9: **for** $i = 1$ to $|\mathbb{I}|$ **do**
10: $\quad$ **for** $\tau = 1$ to t_i **do**
11: $\quad\quad$ broadcast the packet from flow i
12: $\quad$ **for** $n \in \mathbb{N}$ **do**
13: $\quad\quad d_{i,n}(k+1) \leftarrow d_{i,n}(k) + q_{i,n} - [1 - (1 - p_{b_i,n})^{t_i}]$

Theorem 19. *Algorithm 5 is feasibility-optimal when network coding is not employed.*

Proof. Suppose that in some interval $[kT, (k+1)T)$, Algorithm 5 schedules the packet from flow i for transmission t_i^0 times. Suppose there is another algorithm that schedules the packet from flow i for transmission t_i' times, with $\sum_{i \in S_k} t_i' \leq T$. We show that

$$\begin{aligned}
&\sum_{i \in S_k, n} d_{i,n}(k)^+ [1 - (1-p_n)^{t_i^0}] \\
&= \sum_{i \in S_k, n} \sum_{t=1}^{t_i^0} m_i(t) \\
&\geq \sum_{i \in S_k, n} \sum_{t=1}^{t_i'} m_i(t) \\
&= \sum_{i \in S_k, n} d_{i,n}(k)^+ [1 - (1-p_n)^{t_i'}].
\end{aligned}$$

If $t_i' \leq t_i^0$, for all $i \in S_k$, then the inequality

$$\sum_{i \in S_k, n} \sum_{t=1}^{t_i^0} m_i(t) \geq \sum_{i \in S_k, n} \sum_{t=1}^{t_i'} m_i(t)$$

holds. If there exists some i such that $t_i' > t_i^0$, then there also exists some j such that $t_j' < t_j^0$, since $\sum_{i \in S_k} t_i' \leq T = \sum_{i \in S_k} t_i^0$. By the design of the algorithm and the fact that both $m_i(\cdot)$ and $m_j(\cdot)$ are decreasing functions, we have that $m_j(t_i') \leq m_i(t_i^0 + 1) \leq m_j(t_j^0) \leq m_j(t_j' + 1)$. Thus, we can decrement t_i' by 1 and increment t_j' by 1 without decreasing the value of $\sum_{i \in S_k, n} \sum_{t=1}^{t_i'} m_i(t)$. We repeat this procedure until $t_i' \leq t_i^0$, for all i, and deduce that $\sum_{i \in S_k, n} \sum_{t=1}^{t_i^0} m_i(t) \geq \sum_{i \in S_k, n} \sum_{t=1}^{t_i'} m_i(t)$. □

We now analyze the complexity of the Greedy Algorithm. We can implement the Greedy Algorithm using a max-heap, where there is one node for each flow i whose value is m_i. In Steps 4 and 8, it takes $O(|\mathbb{N}|)$ time to compute m_i. It takes $O(|\mathbb{I}| \log |\mathbb{I}|)$ time to construct the max-heap. In each iteration between Steps 5 and 8, it takes $O(\log |\mathbb{I}|)$ time to find $\arg\max_{j \in S_k} m_j$ and remove the node from the max-heap. It also takes $O(\log |\mathbb{I}|)$ time to insert that node back into the max-heap once its value is updated. Thus, the total complexity of computing the schedule in an interval is $O(|\mathbb{I}| \log |\mathbb{I}| + T|\mathbb{N}| + T \log |\mathbb{I}|)$.

8.4 Broadcasting with XOR Coding

In this section, we address the use of XOR coding for broadcasting. We assume that the basestation can either broadcast a raw packet from a flow, or it can choose to broadcast an encoded packet (packet from flow $i \oplus$ packet from flow j), the XOR of a packet from flow i with a packet from flow j, which we shall henceforth

denote by $i \oplus j$. A client can recover the packet from flow i either upon directly receiving a raw packet from flow i, or upon receiving a raw packet from flow j and an encoded packet $i \oplus j$, for some j. We exhibit a simple example where a system with XOR coding can achieve strictly better performance than one without network coding.

Example 2. *Consider a system with two flows that generate one packet in each interval, and only one client whose channel reliability is* $p_1 = 0.5$. *Assume that there are six time slots in an interval. Suppose that the basestation transmits each packet three times in an interval. Then we have* $\hat{q}_{1,1} = \hat{q}_{2,1} = 0.875$. *Thus, a system with timely throughput requirements* $q_{1,1} = q_{2,1} > 0.875$ *is not feasible when network coding is not employed. On the other hand, a system that employs XOR coding can transmit each of the three different types of packets, the raw packet from each flow and the encoded packet* $1 \oplus 2$*, twice in each interval, which achieves* $\hat{q}_{1,1} = \hat{q}_{2,1} = 0.890625$.

While it may be computationally complicated to design a feasibility-optimal scheduling policy when XOR coding is employed, we aim to design a tractable policy that achieves better performance than the Greedy Algorithm in Section 8.3. Suppose the Greedy Algorithm broadcasts the packet from flow i for a total of t_i^G times in an interval. We sort all flows so that $t_1^G \geq t_2^G \geq \ldots$, and enforce the following restrictions on our scheduling policy:

1. In addition to raw packets, we only allow encoded packets of the form $(2i-1) \oplus (2i)$. The intuition behind this restriction is that we only combine two packets which have each been transmitted a similar number of times under the Greedy Algorithm, which implies that they have similar importance.

2. The total number of transmissions scheduled for the raw packets from flow $2i-1$ and flow $2i$, as well as the encoded packet $(2i-1) \oplus (2i)$, equals $t_{2i-1}^G + t_{2i}^G$. The intuition behind this restriction is that we aim to enhance the performance of flows $2i-1$ and $2i$ by XOR coding without hurting other flows.

We call the above two restrictions the *pairwise combination restriction* and the *transmission conservation restriction for XOR coding*, respectively. Suppose that, under some policy η that follows the above restrictions, the raw packet from flow i is transmitted t_i times, and the encoded packet $(2i-1) \oplus (2i)$ is transmitted $t_{(2i-1)\oplus(2i)}$ times in the k^{th} interval. The probability that client n receives the packet from flow $(2i-1)$ is $\hat{q}_{2i-1,n}(k) = 1 - (1-p_n)^{t_{2i-1}}[(1-p_n)^{t_{2i}} + (1-p_n)^{t_{(2i-1)\oplus(2i)}} - (1-p_n)^{t_{2i}+t_{(2i-1)\oplus(2i)}}]$. Similarly, we also have $\hat{q}_{2i,n}(k) = 1 - (1-p_n)^{t_{2i}}[(1-p_n)^{t_{2i-1}} + (1-p_n)^{t_{(2i-1)\oplus(2i)}} - (1-p_n)^{t_{2i-1}+t_{(2i-1)\oplus(2i)}}]$. By Theorem 18, designing a feasibility-optimal policy among all policies that follow the above restrictions can be simplified to one of solving the following integer programming problem for all k:

$$\max \sum_{i=1}^{|S_k|/2} \sum_n d_{2i-1,n}(k)^+ \hat{q}_{2i-1,n}(k) + d_{2i,n}(k)^+ \hat{q}_{2i,n}(k)$$

$$\text{s.t. } t_{2i-1} + t_{2i} + t_{(2i-1)\oplus(2i)} = t^G_{2i-1} + t^G_{2i}, \forall 1 \le i \le |S_k|/2,$$

$$t_i \ge 0 \qquad , \forall 1 \le i \le |S_k|,$$

$$t_{(2i-1)\oplus(2i)} \ge 0 \qquad , \forall 1 \le i \le |S_k|/2.$$

In the formulation, we assume that $|S_k|$, the number of flows that generate a packet in the k^{th} interval, is even. If $|S_k|$ is odd, we can add an imaginary flow i^* into the system to make $|S_k|$ even. We set $q_{i^*,n} = 0$, for all n, and thus $t^G_{i^*} = 0$ since it will never be scheduled by the Greedy Algorithm. The condition $t_{2i-1} + t_{2i} + t_{(2i-1)\oplus(2i)} = t^G_{2i-1} + t^G_{2i}$ allows us to decompose this integer programming problem into $|S_k|/2$ subproblems so that the i^{th} subproblem only involves flows $2i-1$ and $2i$:

$$\max \sum_n d_{2i-1,n}(k)^+ \hat{q}_{2i-1,n}(k) + d_{2i,n}(k)^+ \hat{q}_{2i,n}(k)$$

$$\text{s.t. } t_{2i-1} + t_{2i} + t_{(2i-1)\oplus(2i)} = t^G_{2i-1} + t^G_{2i},$$

$$t_{2i-1}, t_{2i}, t_{(2i-1)\oplus(2i)} \ge 0.$$

If we further assume that $t_{(2i-1)\oplus(2i)}$ is fixed, this subproblem is equivalent to

$$\max \sum_n d_{2i-1,n}(k)^+ (1-p_n)^{t_{(2i-1)\oplus(2i)}} [1-(1-p_n)^{t_{2i-1}}] + \sum_n d_{2i,n}(k)^+ (1-p_n)^{t_{(2i-1)\oplus(2i)}} [1-(1-p_n)^{t_{2i}}] + C$$

$$\text{s.t. } t_{2i-1} + t_{2i} = t^G_{2i-1} + t^G_{2i} - t_{(2i-1)\oplus(2i)},$$

$$t_{2i-1}, t_{2i} \ge 0,$$

where C is a constant. The optimal (t_{2i-1}, t_{2i}) for this problem can be found by Algorithm 6. The complexity of Algorithm 6 is $O(|\mathbb{N}|(t^G_{2i-1} + t^G_{2i}))$. We have the following lemma, whose proof is essentially the same as that of Theorem 19.

Lemma 13. *Given $t_{(2i-1)\oplus(2i)}$, the pair (t_{2i-1}, t_{2i}) found by Algorithm 6 maximizes*

$$\sum_n d_{2i-1,n}(k)^+ \hat{q}_{2i-1,n}(k) + d_{2i,n}(k)^+ \hat{q}_{2i,n}(k).$$

Using Algorithm 6 as a building block, we propose the *Pairwise XOR* algorithm, shown in Algorithm

Algorithm 6 GreedyXOR$(i, t_{(2i-1)\oplus(2i)}, t^G_{2i-1}, t^G_{2i})$

$t_{2i-1} \leftarrow 1$
$t_{2i} \leftarrow 1$
$m_{2i-1} \leftarrow \sum_{n\in\mathbb{N}} d_{2i-1,n}(k)^+(1-p_n)^{t_{(2i-1)\oplus(2i)}}p_n$
$m_{2i} \leftarrow \sum_{n\in\mathbb{N}} d_{2i,n}(k)^+(1-p_n)^{t_{(2i-1)\oplus(2i)}}p_n$
for $\tau = 1$ to $t^G_{2i-1} + t^G_{2i} - t_{(2i-1)\oplus(2i)}$ **do**
  $j \leftarrow \arg\max\{m_{2i-1}, m_{2i}\}$
  $t_j \leftarrow t_j + 1$
  $m_j \leftarrow \sum_{n\in\mathbb{N}} d_{j,n}(k)^+(1-p_n)^{t_{(2i-1)\oplus(2i)}}p_n(1-p_n)^{t_j-1}$
return (t_{2i-1}, t_{2i})

7, to find the optimal schedule when XOR coding is employed under the two aforementioned restrictions. The complexity of the Pairwise XOR algorithm is $O(|\mathbb{I}|\log|\mathbb{I}| + T|\mathbb{N}| + T\log|\mathbb{I}| + \sum_{i=1}^{|S_k|/2}|\mathbb{N}|(t^G_{2i-1}+t^G_{2i})^2) = O(|\mathbb{I}|\log|\mathbb{I}| + T\log|\mathbb{I}| + T^2|\mathbb{N}|)$. The following theorem is the direct result of Lemma 13 and Theorem 18.

Theorem 20. *The Pairwise XOR algorithm is feasibility-optimal among all policies that follow the pairwise combination restriction and the transmission conservation restriction for XOR coding. In particular, the Pairwise XOR algorithm fulfills every system that can be fulfilled by the Greedy Algorithm.*

Algorithm 7 Pairwise XOR

Obtain $t^G_1, t^G_2, \ldots$ from the Greedy Algorithm
Sort flows so that $t^G_1 \geq t^G_2 \geq \ldots$
for $i = 1$ to $|S_k|/2$ **do**
  $Opt \leftarrow -\infty$
  for $t = 0$ to $t^G_{2i-1} + t^G_{2i}$ **do**
    $(t_{2i-1}, t_{2i}) \leftarrow GreedyXOR(i, t_{(2i-1)\oplus(2i)}, t^G_{2i-1}, t^G_{2i})$
    if $\sum_n d_{2i-1,n}(k)^+\hat{q}_{2i-1,n}(k) + d_{2i,n}(k)^+\hat{q}_{2i,n}(k) > Opt$ **then**
      $Opt \leftarrow \sum_n d_{2i-1,n}(k)^+\hat{q}_{2i-1,n}(k) + d_{2i,n}(k)^+\hat{q}_{2i,n}(k)$
      $t^X_{2i-1} \leftarrow t_{2i-1}$
      $t^X_{2i} \leftarrow t_{2i}$
      $t^X_{(2i-1)\oplus(2i)} \leftarrow t$
for $i = 1$ to $|S_k|/2$ **do**
  for $\tau = 1$ to t^X_{2i-1} **do**
    Broadcast the packet from flow $2i-1$
  for $\tau = 1$ to t^X_{2i} **do**
    Broadcast the packet from flow $2i$
  for $\tau = 1$ to $t^X_{(2i-1)\oplus(2i)}$ **do**
    Broadcast the packet $(2i-1)\oplus(2i)$

8.5 Broadcasting with Linear Coding

In this section, we address the use of linear coding to improve the performance of broadcasting delay-constrained flows. We assume that, in addition to raw packets, the basestation can also broadcast packets that contain linear combinations of packets from any subset of flows $L \subseteq S_k$. A client can decode all packets

from the subset L of flows if it receives at least $|L|$ packets that contain linear combinations of packets from these flows. If a client receives less than $|L|$ packets containing such linear combinations, none of the packets from these flows can be decoded. We first exhibit a simple example where a system that uses linear coding provides better performance than one that does not use network coding.

Example 3. *Consider a system with one client, whose channel reliability is $p_1 = 0.5$, three flows that generate one packet in each interval, and nine time slots in an interval. A similar argument as that in Example 2 shows that $q_{1,1} = q_{2,1} = q_{3,1} > 0.875$ is not feasible when network coding is not employed. On the other hand, if the basestation employs linear coding and broadcasts a linear combination of the three flows in each time slot, the client can decode all packets from the three flows if it receives at least three packets out of the nine transmissions in an interval, which has probability 0.91015625.*

As in Section 8.4, we address the problem of finding a tractable scheduling policy that achieves better performance than the Greedy Algorithm. Suppose that the Greedy Algorithm schedules t_i^G transmissions for the packet from flow i in some interval. We sort all flows so that $t_1^G \geq t_2^G \geq \ldots$, and enforce the following restrictions:

1. Flows are grouped into subsets as $L_1 = \{1, 2, \ldots, l_1\}, L_2 = \{l_1 + 1, \ldots, l_2\}, \ldots$. In each time slot, the basestation broadcasts a linear combination of packets from flows in one of the subsets $L_1, L_2, \ldots$. The intuition behind this restriction is that we only combine packets that have been scheduled similar numbers of times.

2. The basestation broadcasts linear combinations of packets from the subset $L_h = \{l_{h-1} + 1, l_{h-1} + 2, \ldots, l_h\}$ a total number of $\sum_{i=l_{h-1}+1}^{l_h} t_i^G$ times, where we set $l_0 = 0$. The intuition behind this restriction is that we aim to enhance the performance of flows within L_h without hurting other flows.

The two restrictions above are called the *adjacent combination restriction* and the *transmission conservation restriction for linear coding*, respectively.

We define $r_{n,u,v}$ as the probability that client n receives at least u packets successfully out of v transmissions. We can compute $r_{n,u,v}$ for all $n \in \mathbb{N}$, $1 \leq u \leq T$, and $1 \leq v \leq T$ in $O(|\mathbb{N}|T^2)$ time using the following iteration:

$$r_{n,u,v} = \begin{cases} 1, & \text{if } u = 0, \\ 0, & \text{if } u > 1, v = 0, \\ p_n r_{n,u-1,v-1} + (1 - p_n) r_{n,u,v-1}, & \text{else.} \end{cases}$$

If flows are grouped as $L_1, L_2, \ldots$ in the k^{th} interval, the probability that client n is able to obtain the packet from flow $i \in L_h$ is then $\hat{q}_{i,n}(k) = r_{n,|L_h|,(\sum_{i=l_{h-1}+1}^{l_h} t_i^G)}$. We need to find the optimal way to group flows such

that

$$\sum_{i \in S_k} \sum_n d_{i,n}(k)^+ \hat{q}_{i,n}(k)$$

is maximized. We solve this problem by dynamic programming. Let $H_{i,j}$ be the optimal way to group flows $i, i+1, \ldots, j$, if flows within $[i,j]$ are not allowed to be grouped with flows outside $[i,j]$. We represent $H_{i,j}$ by the collection of groups formed by flows within $[i,j]$. We then have that $H_{i,j}$ either contains one single group consisting of all flows within $[i,j]$, or is of the form $H_{i,h} \cup H_{h+1,j}$, for some $i \leq h < j$. The optimal way to group all flows, which is $H_{1,|S_k|}$, can be found by dynamic programming as in Algorithm 8. The complexity of Algorithm 8 is $O(|\mathbb{I}| \log |\mathbb{I}| + T|\mathbb{N}| + T \log |\mathbb{I}| + |\mathbb{N}||\mathbb{I}|^3)$. We have the following theorem:

Theorem 21. *The Optimal Grouping policy, as described in Algorithm 8, is feasibility-optimal among all policies that follow the adjacent combination restriction and the transmission conservation restriction for linear coding. In particular, The Optimal Grouping policy fulfills all systems that can be fulfilled by the Greedy Algorithm.*

Proof. The first part of the theorem follows from the discussion in the previous paragraph and Theorem 18. The second part of the theorem follows because a policy that sets $H_{1,|S_k|} = \{\{1\}, \{2\}, \ldots\}$ in each interval k has the same schedule as that resulting from the Greedy Algorithm. □

Algorithm 8 Optimal Grouping

Obtain $t_1^G, t_2^G, \ldots$ from the Greedy Algorithm
Sort flows so that $t_1^G \geq t_2^G \geq \ldots$
for $i = 1$ to $|S_k|$ **do**
  $O_{i,i} \leftarrow \sum_n d_{i,n}(k)^+ r_{n,1,t_i^G}$
  $H_{i,i} \leftarrow \{\{i\}\}$
for $s = 2$ to $|S_k|$ **do**
  for $i = 1$ to $|S_k| - s + 1$ **do**
    $total \leftarrow \sum_{h=i}^{i+s-1} t_h^G$
    $O_{i,i+s-1} \leftarrow \sum_{h=i}^{i+s-1} \sum_n d_{h,n}(k)^+ r_{n,s,total}$
    $H_{i,i+s-1} \leftarrow \{\{i, i+1, \ldots, i+s-1\}\}$
    for $j = i$ to $i + s - 2$ **do**
      if $O_{i,j} + O_{j+1,i+s-1} > O_{i,i+s-1}$ **then**
        $O_{i,i+s-1} = O_{i,j} + O_{j+1,i+s-1}$
        $H_{i,i+s-1} = H_{i,j} \cup H_{j+1,i+s-1}$
Group flows as in $H_{1,|S_k|}$ and broadcast them accordingly

8.6 Simulation Results

We have implemented the three scheduling algorithms proposed in this paper, namely, the Greedy Algorithm, the Pairwise XOR algorithm, and the Optimal Grouping algorithm, in ns-2. We compare their performances against a round-robin scheduling policy.

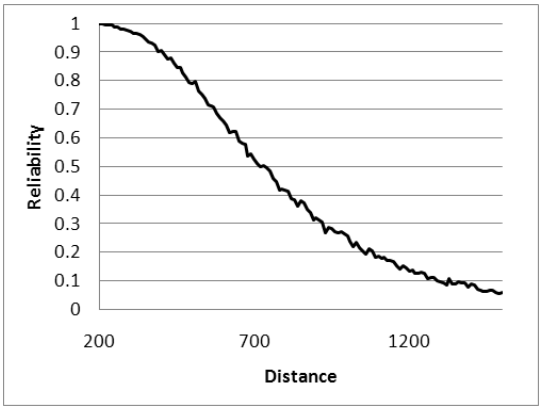


Figure 8.1: Relationship between distance and link reliability

Region	subscribed flows and timely throughput requirements
$[390, 780] \times [520, 1040]$	$q_{1,n} = q_{5,n} = \alpha$, $q_{6,n} = q_{10,n} = \beta$
$[390, 780] \times [0, 520]$	$q_{2,n} = q_{5,n} = \alpha$, $q_{7,n} = q_{10,n} = \beta$
$[0, 390] \times [520, 1040]$	$q_{3,n} = q_{5,n} = \alpha$, $q_{8,n} = q_{10,n} = \beta$
$[0, 390] \times [0, 520]$	$q_{4,n} = q_{5,n} = \alpha$, $q_{9,n} = q_{10,n} = \beta$

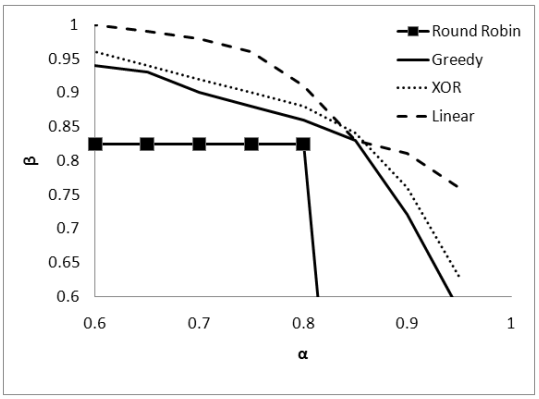
Table 8.1: Timely throughput requirements of clients in each region for the asymmetric topology

We use the `Shadowing` module in ns-2 to simulate the unreliable wireless links between the basestation and clients. In the `Shadowing` module, the link reliability decreases as the distance between two wireless devices increases. The relation between link reliability and distance is shown in Figure 8.1.

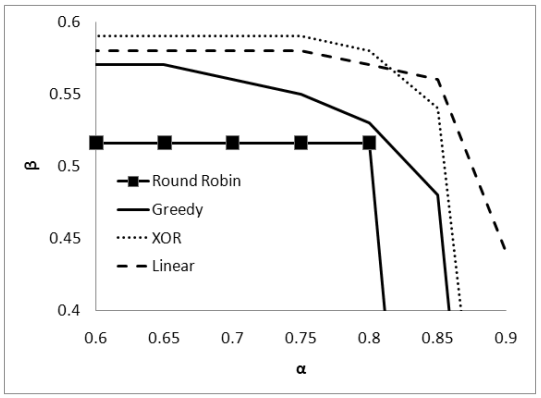
We implement our algorithms based on the IEEE 802.11 standard. Under 802.11, broadcasting a packet with size 160 bytes, which is the size of VoIP packets using the G.711 codec [72], takes about 2 ms. We assume the length of an interval is 40 ms, and hence it consists of 20 time slots.

We consider the scenario where a basestation is broadcasting 10 delay-constrained flows to 20 clients that are evenly distributed in a 780×1040 area. We consider two different topologies and timely throughput requirements of clients. The first one is called the *symmetric topology*. In the symmetric topology, the basestation is located at the center of the domain, i.e., at position $(390, 520)$. The timely throughput requirements of each client are α for flows 1–5, and β for flows 6–10. That is, we set $q_{i,n} = \alpha$ if $i \leq 5$, and $q_{i,n} = \beta$ if $i > 5$, where α and β are tunable variables to reflect that clients may have different timely throughput requirements for different flows. The other topology that we consider is called the *asymmetric topology*, where the basestation is located at position $(520, 650)$. Further, clients in different regions may subscribe to different flows. The timely throughput requirements of flows subscribed to by clients in each region are summarized in Table 8.1. We set $q_{i,n} = 0$ if $q_{i,n}$ does not appear in Table 8.1.

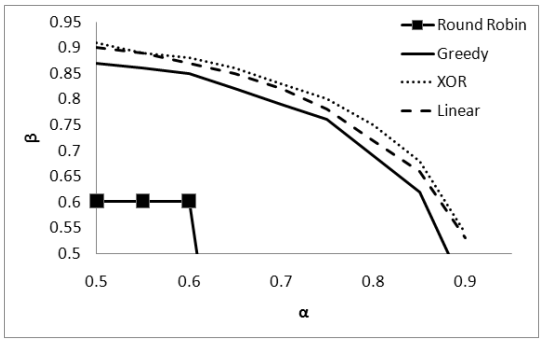
We study two different types of traffic patterns for packet arrivals, namely, *deterministic arrivals* and



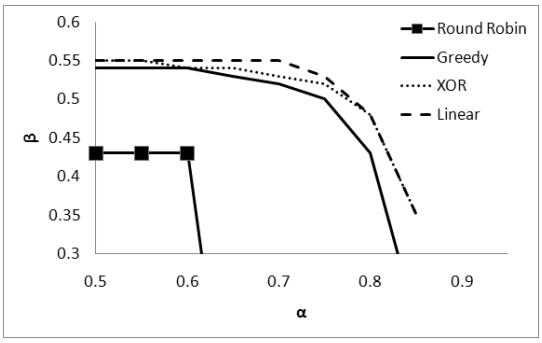
(a) Deterministic arrivals and symmetric topology



(b) Probabilistic arrivals and symmetric topology



(c) Deterministic arrivals and asymmetric topology



(d) Probabilistic arrivals and asymmetric topology

Figure 8.2: Simulation results

probabilistic arrivals. For deterministic arrivals, we assume that all the 10 flows generate one packet in each interval. This corresponds to flows carrying constant-bit-rate traffic, such as the G.711 codec for VoIP. For probabilistic arrivals, we assume that each flow generates one packet with some probability, independent of the packet generations of other flows, at the beginning of each interval. This scenario corresponds to flows carrying variable-bit-rate traffic, such as MPEG video streaming. In particular, we assume each of flows 1–5 generates one packet with probability 0.9, and each of flows 6–10 generates one packet with probability 0.6, at the beginning of each interval.

We evaluate the performances of our algorithms and the round-robin policy for each of the two topologies and each of the two traffic patterns. We compare the performances of different scheduling algorithms by comparing the pairs of (α, β) that can be fulfilled by each algorithm. A system is considered to be fulfilled if, after 500 intervals, the average timely throughput of client n on flow i is at least $q_{i,n} - 0.03$.

The simulation results are shown in Figure 8.2. As shown in the figure, all the three proposed algorithms outperform the round-robin policy in all scenarios. This is because, without the knowledge of timely throughput requirements, the round-robin policy cannot offer any tradeoff between flows. Further, the differences of performance between the round-robin policy and the three proposed policies are even more

significant in scenarios with asymmetric topology, as shown in Figures 8.2c and 8.2d, because the round-robin policy does not incorporate any knowledge of network topology. As the basestation has better links to clients in the region $[390, 780] \times [520, 104]$ than those to clients in the region $[0, 390] \times [0, 520]$, the three proposed policies allocate more transmission opportunities to flows subscribed to by clients in the region $[0, 390] \times [0, 520]$ so as to compensate for their low link qualities. These results show that an efficient policy for broadcasting delay-constrained flows needs to jointly consider both client requirements and network topology. Also, both algorithms using network coding achieve better performance than the Greedy Algorithm, which demonstrates that network coding can be used to increase the capacity of wireless networks for broadcasting delay-constrained flows.

We close this section by comparing the Pairwise XOR algorithm for XOR coding and the Optimal Grouping algorithm for linear coding. In the scenario with deterministic arrivals and symmetric topology, the Optimal Grouping algorithm has much better performance than the Pairwise XOR algorithm. However, the advantage of the Optimal Grouping algorithm becomes less prominent in the other three scenarios, and sometimes it even performs worse than the Pairwise XOR algorithm. The Optimal Grouping algorithm allows combining more than two flows, and thus explores more coding possibilities, which is why it achieves better performance in the first scenario. On the other hand, in systems where the number of generated packets in each interval is less, as in the scenario with probabilistic arrivals, or when the topologies are asymmetric, it becomes less beneficial to combine a large number of packets. In such systems, the Pairwise XOR algorithm may benefit from its simpler coding structure.

Chapter 9

Content-aware Scheduling for Multimedia Servers

In previous chapters, we have measured performance in terms of timely throughput. The underlying assumption of timely throughput is that every packet from the same flow is equally important. In some applications, such as video streaming, packets from the same flow may have different importance. In this chapter, we discuss the scheduling problem for a multimedia server where packets from the same flow are not equally important. We propose a more general model, which can accommodate not only the scheduling problems for multimedia servers, but also the imprecise computation model [15,51] and the IRIS model [16] in real-time literature.

We first derive a necessary and sufficient condition for feasibility. We next obtain an off-line feasibility optimal scheduling policy. We then study a sufficient condition for a policy to be feasibility optimal or achieve some approximation bound. This condition serves as a guideline for designing on-line scheduling policy and we obtain a greedy policy based on it. We prove that the on-line policy is feasibility optimal under some conditions, and also obtain an approximation bound for the policy under general cases. We test our policies in comparative simulations.

9.1 System Model

In this section, we introduce our system model. The model is compatible with the imprecise computation model and the IRIS model. Our model is hence suitable for all applications that fit these two models. In particular, we also demonstrate that our model can be used to model a video streaming server.

Consider a system with a set $S = \{A, B, \dots\}$ of real-time tasks. Time is slotted and denoted by $t \in \{1, 2, 3, \dots\}$. Each task X generates a job periodically with period τ_X. A job can be executed multiple times in the period where it is generated; the execution of a job does not mean its completion. The job is removed from the system when the next period begins. In other words, the relative deadline of a job generated by task X is also τ_X. We assume that all tasks in S generate a job at time $t = 0$. We denote a frame as the time between two consecutive time slots where all tasks generate a job. The length of a frame, which we denote

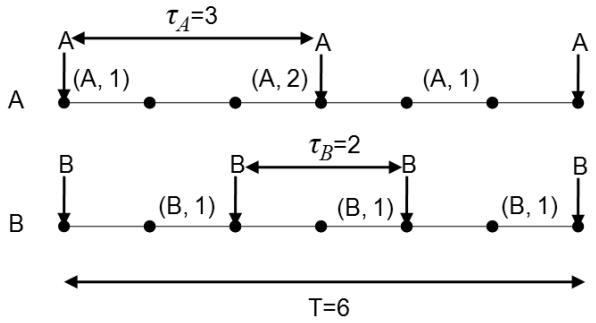


Figure 9.1: An example of the system and scheduling policy over a frame of six slots, which consists of two periods of task A and three periods of task B. Arrows in the figure indicate the beginning of a new period.

by T, is the least common multiple of $\{\tau_X | X \in S\}$. Thus, a frame consists of T/τ_X periods of task X.

As noted above, each job can be executed in an arbitrary number of time slots before its deadline. Each task obtains a certain *reward* each time that its job is executed. The total amount of reward obtained by a task in a period depends on the number of times that its job has been executed in the period. More formally, task X obtains reward $r_X^i \geq 0$ when it executes its job for the i^{th} time in a period. For example, if a job of task X is executed a total of n time slots, then the total reward obtained by task X in this period is $r_X^1 + r_X^2 + \cdots + r_X^n$. We further assume that the marginal reward of executing a job decreases as the number of executions increases, that is, $r_X^{i+1} \leq r_X^i$, for all i and X. Thus, the total reward that a task obtains in a period is an increasing and concave function of the number of time slots that its job is executed.

A *scheduling policy* η for the system is one that chooses an *action*, possibly at random, in each time slot, based on past history and system information. The action taken by η at time t is described by $\eta(t) = (X, i)$, meaning that the policy executes the job of task X at time t and that this is the i^{th} time that the job is being executed in the period. Fig. 9.1 shows an example with two tasks over one frame, which consists of two periods of task A and three periods of task B. In this example, action $(A, 1)$ is executed twice and $(A, 2)$ is executed once in a frame. Thus, the reward obtained by task A in this frame is $2r_A^1 + r_A^2$. On the other hand, the reward obtained by task B in this frame is $3r_B^1$.

The performance of the system is described by the long-term *average reward* per frame of each task in the system.

Definition 25. *Let $\tilde{q}_X(k)$ be the total reward obtained by task X in the frame $((k-1)T, kT]$ under some scheduling policy η. The* average reward *of task X is defined as $q_X := \liminf_{k\to\infty} \frac{\sum_{i=1}^{k} \tilde{q}_X(i)}{k}$.*

We assume that there is a minimum average reward requirement for each task X, $q_X^* > 0$. We wish to verify whether a system is *feasible*, that is, whether each task can have its minimum average reward requirement satisfied.

Definition 26. *A system is* fulfilled *by a scheduling policy η if, under η, $q_X \geq q_X^*$ with probability 1, for all $X \in S$.*

Definition 27. *A system is* feasible *if there exists some scheduling policy that fulfills it.*

A natural metric to evaluate a scheduling policy is the set of systems that the policy fulfills. For ease of discussion, we only consider systems that are *strictly feasible*.

Definition 28. *A system with minimum reward requirements $[q_X^*|X \in S]$ is* strictly feasible *if there exists some $\epsilon > 0$ such that the same system with minimum reward requirements $[(1+\epsilon)q_X^*]$ is feasible.*

Definition 29. *A scheduling policy is* feasibility optimal *if it fulfills all strictly feasible systems.*

We limit discussions on strictly feasible systems only to simplify the proof in Theorem 27. Since ϵ can be chosen to be any arbitrarily small positive number, this limitation does not have practical importance.

Moreover, since the overhead for computing a feasibility optimal policy may be too high in certain scenarios, we also need to consider simple approximation policies.

Definition 30. *A scheduling policy is a p-*approximation policy*, $p \geq 1$, if it fulfills all systems with minimum reward requirements $[q_X^*]$ such that the same system with minimum reward requirements $[pq_X^*]$ is strictly feasible.*

9.1.1 Extensions for Imprecise Computation Models

In this section, we discuss how our proposed model can be used to handle imprecise computation models and IRIS models. In such models, a task consists of two parts: a mandatory part and an optional part. The mandatory part is required to be completed in each period, or else the system fails. After the mandatory part is completed, the optional part can be executed to improve performance. The more optional parts executed for a task, the more rewards it gets.

Let m_X be the length of the mandatory part of task X, that is, it is required that each job of X is executed at least m_X time slots in each of its period. Let o_X be the length of the optional part of task X. To accommodate this scenario, we define a symbolic value M with the following arithmetic reminiscent of the "Big-M Method" in linear programming: $0 \times M = 0$, $aM + bM = (a+b)M$, $a \times (bM) = (ab)M$, $aM + c > bM + d$, if $a > b$, and $aM + c > aM + d$ if $c > d$, for all real numbers a, b, c, d. Loosely speaking, M can be thought of as a huge positive number. For each task X, we set $r_X^1 = r_X^2 = \cdots = r_X^{m_X} = M$, we then set $r_X^{m_X+1}, r_X^{m_X+2}, \ldots, r_X^{m_X+o_X}$ according to the rewards obtained by X for its optional part, and $r_X^i = 0$, for all $i > m_X + o_X$. The minimum reward requirement of task X is set to be $\frac{T}{\tau_X} m_X M + \hat{q}_X^*$ with $\hat{q}_X^* \geq 0$. Thus, a scheduling policy that fulfills such a system is guaranteed to complete each mandatory part with probability one.

We now discuss how to apply our model to a video streaming server. In typical MPEG videos, there are three types of frames: I-frames, P-frames, and B-frames. A video is composed of groups of frames (GOFs) where a GOF consists of a frame sequence of I,B,B,P,B,B,P,B,B,P,.... We can treat a video stream as a task in the system and its GOFs as the jobs of the task. The I-frames consist of the complete information of the frame and can be decoded independently. On the other hand, P-frames and B-frames may require the information of the I-frame and P-frames in the GOF to be decoded. If an I-frame is lost, none of the succeeding frames in the GOF can be decoded. Thus, the I-frames should be treated as mandatory parts. On the other hand, a lost B-frame does not cause problems for other frames and only results in minor video quality degradation. Thus, B-frames can be treated as optional parts and the server can drop B-frames when it is overloaded. The reward for a B-frame reflects the additional video quality it provides. Finally, while a lost P-frame can also cause problems to other P-frames and B-frames, its influence is not as severe as a lost I-frame. P-frames can therefore be treated as either mandatory parts or optional parts, depending on the requirements of subscribers of the videos. Details of MPEG encoding can be found in [3].

9.2 Feasibility Analysis

In this section, we establish a necessary and sufficient condition for a system to be feasible. Consider a feasible system that is fulfilled by a policy η. Suppose that, on average, there are f_X^i periods of task X in a frame in which the action (X, i) is taken by η. The average reward of task X can then be expressed as $q_X = \sum_{i=1}^{\tau_X} f_X^i r_X^i$. We can immediately obtain a necessary condition for feasibility.

Lemma 14. *A system with a set of tasks $S = \{A, B, \dots\}$ is feasible only if there exists $\{f_X^i | X \in S, 1 \le i \le \tau_X\}$ such that*

$$q_X^* \le \sum_{i=1}^{\tau_X} f_X^i r_X^i, \qquad \forall X \in S, \tag{9.1}$$

$$0 \le f_X^i \le T/\tau_X, \qquad \forall X \in S, 1 \le i \le \tau_X, \tag{9.2}$$

$$\sum_{X \in S} \sum_{i=1}^{\tau_X} f_X^i \le T. \tag{9.3}$$

Proof. Condition (9.1) holds because task X requires that $q_X^* \le q_X$. Condition (9.2) holds because there are T/τ_X periods of task X in a frame and thus f_X^i is upper-bounded by T/τ_X. Finally, the total average number of time slots that the system executes one of the jobs in a frame can be expressed as $\sum_{X \in S} \sum_{i=1}^{\tau_X} f_X^i$, which is upper-bounded by the number of time slots in a frame, T. Thus, condition (9.3) follows. □

Next, we show that the conditions (9.1)–(9.3) are also sufficient for feasibility. In practice, we also have

$f_X^i \geq f_X^{i+1}$, since we need to schedule the action (X, i) before the action $(X, i+1)$ can be taken. However, we note that we do not need this constraint when evaluating feasibility. To prove that the conditions (9.1)–(9.3) are sufficient, we first show that the polytope, which contains all points $f = (f_A^1, f_A^2, \ldots, f_A^{\tau_A}, f_B^1, \ldots)$ that satisfy conditions (9.2) and (9.3), is a convex hull of several integer points. We then show that for all integer points $n = (n_A^1, n_A^2, \ldots, n_A^{\tau_A}, n_B^1, \ldots)$ in the polytope, there is a schedule under which the reward obtained by task X is at least $\sum_{i=1}^{\tau_X} n_X^i r_X^i$. We then prove sufficiency using these two results.

Define a matrix $H = [h_{i,j}]$ with $2\sum_{X \in S} \tau_X + 1$ rows and $\sum_{X \in S} \tau_X$ columns as follows:

$$h_{i,j} = \begin{cases} 1, & \text{if } i = 1, \\ 1, & \text{if } i = 2j, \\ -1, & \text{if } i = 2j+1, \\ 0, & \text{else.} \end{cases} \tag{9.4}$$

Define $b = [b_i]$ to be a column vector with $2\sum_{X \in S} \tau_X + 1$ elements so that $b_1 = T$; the first τ_A elements with even indices are set to T/τ_A, that is, $b_2 = b_4 = \cdots = b_{2\tau_A} = T/\tau_A$; the next τ_B elements with even indices are set to T/τ_B, and so on. All other elements are set to 0. For example, the system shown in Fig. 9.1 would have

$$H = \begin{bmatrix} 1, & 1, & 1, & 1, & \ldots \\ 1, & 0, & 0, & 0, & \ldots \\ -1, & 0, & 0, & 0, & \ldots \\ 0, & 1, & 0, & 0, & \ldots \\ 0, & -1, & 0, & 0, & \ldots \\ 0, & 0, & 1, & 0, & \ldots \\ 0, & 0, & -1, & 0, & \ldots \\ & & \vdots & & \end{bmatrix}, b = \begin{bmatrix} T \\ T/\tau_A \\ 0 \\ T/\tau_A \\ 0 \\ \vdots \\ T/\tau_B \\ 0 \\ \vdots \end{bmatrix}.$$

Thus, conditions (9.2) and (9.3) can be described as $Hf \leq b$. Theorem 5.20 and Theorem 5.24 in [44] show the following:

Theorem 22. *The polytope defined by $\{f | Hf \leq b\}$, where b is an integer vector, is a convex hull of several integer points if for every subset R of rows in H, there exists a partition of $R = R_1 \cup R_2$ such that for every column j in H, we have $\sum_{i_1 \in R_1} h_{i_1,j} - \sum_{i_2 \in R_2} h_{i_2,j} \in \{1, 0, -1\}$.*[1]

Since all elements in b are integers, we obtain the following:

[1] Such a matrix H is called a *totally unimodular matrix* in combinatorial optimization theory.

Theorem 23. *The polytope defined by $\{f|Hf \leq b\}$, where H and b are derived using conditions (9.2) and (9.3) as above, is a convex hull of several integer points.*

Proof. Let $R = \{r_1, r_2, \dots\}$ be the indices of some subset of rows in H. If the first row is in R, we choose $R_1 = \{r|r \in R, r \text{ is odd}\}$ and $R_2 = \{r|r \in R, r \text{ is even}\}$. Since for all columns j in H, $h_{1,j} = 1$, $h_{2j,j} = 1$, $h_{2j+1,j} = -1$, and all other elements in column j are zero, we have $\sum_{i_1 \in R_1} h_{i_1,j} - \sum_{i_2 \in R_2} h_{i_2,j} \in \{1, 0, -1\}$. On the other hand, if the first row is not in R, we choose $R_1 = R$ and $R_2 = \varnothing$. Again, we have $\sum_{i_1 \in R_1} h_{i_1,j} - \sum_{i_2 \in R_2} h_{i_2,j} = \sum_{i_1 \in R_1} h_{i_1,j} \in \{1, 0, -1\}$. Thus, by Theorem 22, the polytope defined by $\{f|Hf \leq b\}$ is a convex hull of several integer points. □

Next we show that all integer points in the polytope can be carried out by some scheduling policy as follows. As noted before, in the following theorem, we do not require that $n_X^i \geq n_X^{i+1}$.

Theorem 24. *Let $n = (n_A^1, n_A^2, \dots, n_A^{\tau_A}, n_B^1, \dots)$ be an integer point in the polytope $\{f|Hf \leq b\}$. Then, there exists a scheduling policy so that $q_X \geq \sum_{i=1}^{\tau_X} n_X^i r_X^i$.*

Proof. We prove this theorem by finding a schedule with $q_X \geq \sum_{i=1}^{\tau_X} n_X^i r_X^i$. Ideally, we want to schedule the action (X, i) n_X^i times in a frame. We can schedule at most one (X, i) action in each period of X. Further, we can schedule at most one action in each time slot. We construct a flow network based on these constraints. In the flow network, there are three layers of nodes between the source r and sink s, which we refer to as $L1, L2$, and $L3$, respectively. In $L1$, each node represents an action (X, i). There is an edge from the source to the node representing (X, i) with capacity n_X^i, since we want to schedule (X, i) n_X^i times. In $L2$, we have T/τ_X nodes for each task X, where each of these nodes represents a period of X and these nodes are named as $[X, 1], [X, 2], \dots, [X, T/\tau_X]$. There is an edge from the node in $L1$ representing (X, i) to the node $[X, j]$ with capacity 1, for all X, i, and j. This represents the restriction that there is at most one (X, i) action in each period of X. In $L3$, there is one node for each time slot in the frame. Nodes in $L3$ are named as $\{1\}, \{2\}, \dots, \{T\}$. There is an edge with infinite capacity from the node $[X, j]$ in $L2$ to $\{t\}$ in $L3$, for all $t \in ((j-1)\tau_X, j\tau_X]$. This shows that the j^{th} period of X consists of the time slots $((j-1)\tau_X, j\tau_X]$. Finally, there is an edge from each node in $L3$ to the sink s with capacity 1, showing that there is at most one action in each time slot. Fig. 9.2a shows the flow network derived from a system with two tasks, A and B, with $\tau_A = 3$, $\tau_B = 2$, $n_A^1 = 1$, $n_A^2 = 2$, $n_A^3 = 0$, $n_B^1 = 2$, and $n_B^2 = 1$.

We now show that the maximum flow of this network is $\sum_i \sum_X n_X^i$. By the max-flow min-cut theorem, it suffices to show that every cut of the network has capacity at least $\sum_i \sum_X n_X^i$. For any given cut, let C_r be the complement that contains the source r and C_s be the complement that contains the sink s. The capacity of the cut is the sum of capacities of edges from C_r to C_s. We consider several different types of

cuts. First, consider a cut where some nodes in $L3$ belong to C_r and some other nodes in $L3$ belong to C_s. There exists t such that the node $\{t\}$ is in C_r and the node $\{t+1\}$ is in C_s, where $t+1 \leq T$. There must be a task X which has a period containing both time slots t and $t+1$, or otherwise t is a common multiple of $\{\tau_X | X \in S\}$ that is smaller that T, which yields a contradiction. Thus, there exists a node $[X, j]$ in $L2$ which has edges toward both $\{t\}$ and $\{t+1\}$, one of which must be between C_r and C_s. The capacity of this cut is infinite since the capacities of edges between $L2$ and $L3$ are infinite. Next, consider a cut where all nodes in $L3$ are in the same complement but some nodes in $L2$ are in the other complement. The capacity of this cut is also infinite because every node in $L2$ has at least one edge, whose capacity is infinite, towards a node in $L3$. It remains to consider cases where the nodes in $L2$ and $L3$ are all in the same complement. If they are in C_r, the capacity of the cut is T since there is one edge with capacity one from each node in $L3$ to the sink. The capacity of this cut is then greater or equal to $\sum_i \sum_X n_X^i$ by (9.3). Finally, if all nodes in $L2$ and $L3$ are in C_s, the capacity of the cut is $\sum_{(X,i) \in C_r} T/\tau_X + \sum_{(X,i) \in C_s} n_X^i \geq \sum_i \sum_X n_X^i$, where the inequality follows from (9.2), and equality is achieved when all nodes in $L1$ are in C_s. In sum, the maximum flow of this network is $\sum_i \sum_X n_X^i$.

Since the capacities of all edges in the network are integers, there exists an integer flow whose value is $\sum_i \sum_X n_X^i$. We construct a schedule based on this integer flow. We schedule the action (X, i) at time slot t if there is there is a flow from (X, i) to $\{t\}$. This schedule has the following properties: each action (X, i) is scheduled n_X^i times; each action (X, i) is scheduled at most once in a period of X; there is at most one action in each time slot. Fig. 9.2b shows an example of the resulting schedule. Note that, under this policy, there are time slots where the policy schedules an action (X, i), but it is instead the j^{th} time that the job of X is executed in the period. Thus, we may need to renumber these actions as in Fig. 9.2c and define $\bar{n}_X^i$ as the actual number of times that the action (X, i) is executed in the frame. In the example of Fig. 9.2, we have $\bar{n}_A^1 = 2, \bar{n}_A^2 = 1, \bar{n}_B^1 = 3$, and $\bar{n}_B^2 = 0$. Since the policy does not schedule two identical actions in a period, we have $\sum_{i=1}^k \bar{n}_X^i \geq \sum_{i=1}^k n_X^i$, for all k and $X \in S$. Thus, $q_X = \sum_{i=1}^{\tau_X} \bar{n}_X^i r_X^i \geq \sum_{i=1}^{\tau_X} n_X^i r_X^i$, since $r_X^1 \geq r_X^2 \geq \ldots$, for all $X \in S$.

□

Now we can derive the necessary and sufficient condition for a system to be feasible.

Theorem 25. *A system with a set of tasks $S = \{A, B, \ldots\}$ is feasible if and only if there exists $\{f_X^i | X \in S, 1 \leq i \leq \tau_X\}$ such that (9.1) - (9.3) are satisfied.*

Proof. Lemma 14 has established that these conditions are necessary. It remains to show that they are also sufficient. Suppose there exists $f = \{f_X^i | X \in S, 1 \leq i \leq \tau_X\}$ that satisfies (9.1) - (9.3). Then Theorem 23 shows that there exists integer vectors $n[1], n[2], \ldots, n[v]$ such that each of them satisfies (9.1) - (9.3), and

$f = \sum_{u=1}^{v} \alpha_u n[u]$, where α_u's are positive numbers with $\sum_{u=1}^{v} \alpha_u = 1$. Let η_u be the scheduling policy for the integer vector $n[u]$ as in the proof of Theorem 24, for each u. Theorem 24 has shown that for each u, the average reward obtained by X under η_u, $q_X[\eta_u]$, is at least $\sum_{i=1}^{\tau_X} n[u]_X^i r_X^i$. Finally, we can design a weighted round robin policy that switches among the policies $\eta_1, \eta_2, \ldots, \eta_v$, with policy η_u being chosen in α_u of the frames. The average reward obtained by X is hence $q_X = \sum_u \alpha_u q_X[\eta_u] \geq \sum_u \alpha_u (\sum_{i=1}^{\tau_X} n[u]_X^i r_X^i) = \sum_{i=1}^{\tau_X} f_X^i r_X^i \geq q_X^*$. Thus, this policy fulfills the system and so the conditions are also sufficient. □

Using Theorem 25, checking whether a system is feasible can be done by any linear programming solver. The computational overhead for checking feasibility can be further reduced by using the fact that $r_X^i \geq r_X^j$, for all $i < j$. Given a system and $\{f_X^i\}$ that satisfy conditions (9.1) - (9.3) with $f_Y^j < \frac{T}{\tau_Y}$ and $f_Y^k > 0$ for some $j < k$ and $Y \in S$, let $\delta = \min\{\frac{T}{\tau_Y} - f_Y^j, f_Y^k\}$. Construct $\{\hat{f}_X^i\}$ such that $\hat{f}_Y^j = f_Y^j + \delta$, $\hat{f}_Y^k = f_Y^k - \delta$, and $\hat{f}_X^i = f_X^i$ for all other elements. Then $\{\hat{f}_X^i\}$ also satisfies conditions (9.1) - (9.3). Based on this observation, we derive an algorithm for checking feasibility, described in Algorithm 9. This essentially transfers slots from less reward earning actions to more reward earning actions. The running time of this algorithm is $O(\sum_{X \in S} \tau_X)$. Since a specification of a system involves at least the $\sum_{X \in S} \tau_X$ variables of $\{r_X^i\}$, Algorithm 9 is essentially a linear time algorithm.

Algorithm 9 Feasibility Checker

Require: S, $\{\tau_X | X \in S\}$, $\{r_X^i | X \in S, 1 \leq i \leq \tau_X\}$, $\{q_X^* | X \in S\}$
for $X \in S$ **do**
  if $q_X^* > \frac{T}{\tau_X} \sum_{i=1}^{\tau_X} r_X^i$ **then**
    return Infeasible
for $X \in S$ **do**
  $i \leftarrow 1$
  while $q_X^* > 0$ **do**
    if $q_X^* > \frac{T}{\tau_X} r_X^i$ **then**
      $f_X^i \leftarrow T/\tau_X$
      $q_X^* \leftarrow q_X^* - \frac{T}{\tau_X} r_X^i$
    else
      $f_X^i \leftarrow q_X^*/r_X^i$
      $q_X^* \leftarrow 0$
    $i \leftarrow i + 1$
if $\sum_{X \in S} \sum_{\tau_X}^{i=1} f_X^i \leq T$ **then**
  return Feasible
else
  return Infeasible

In addition to evaluating feasibility, the proof of Theorem 25 also demonstrates an off-line feasibility optimal policy. In many scenarios, however, on-line policies are preferred. In the next section, we introduce a guideline for designing scheduling policies that is suggestive of simple on-line policies.

9.3 Designing Scheduling Policies

In this section, we study the problem of designing scheduling policies. We establish sufficient conditions for a policy to be either feasibility optimal or p-approximately so.

We start by introducing a metric to evaluate the performance of a policy η. Recall that $\tilde{q}_X(k)$ is the total reward obtained by task X during the frame $((k-1)T, kT]$ and the average reward of X is $q_X = \liminf_{k\to\infty} \frac{\sum_{i=1}^{k} \tilde{q}_X(i)}{k}$. We now define the *debt* of task X.

Definition 31. *The* debt *of task X in the frame $((k-1)T, kT]$, $d_X(k)$, is defined recursively as follows:*

$$d_X(0) = 0,$$
$$d_X(k) = [d_X(k-1) + q_X^* - \tilde{q}_X(k)]^+, \forall k > 0,$$

where $x^+ := \max\{0, x\}$.

Lemma 15. *A system is fulfilled by a policy η if $\lim_{k\to\infty} d_X(k)/k = 0$ with probability 1.*

Proof. We have $d_X(k) \geq kq_X^* - \sum_{i=1}^{k} \tilde{q}_X(i)$ and $d_X(k)/k \geq q_X^* - \frac{1}{k}\sum_{i=1}^{k} \tilde{q}_X(i)$. Thus, if $\lim_{k\to\infty} d_X(k)/k = 0$, then $q_X = \liminf_{k\to\infty} \frac{\sum_{i=1}^{k} \tilde{q}_X(i)}{k} \geq q_X^*$ and the system is fulfilled. □

We can describe the *state* of the system in the k^{th} frame by the debts of tasks, $[d_X(k)|X \in S]$. Consider a policy that schedules jobs solely based on the requirements and the state of the system. The evolution of the state of the system can then be described by a Markov chain. In each frame k, the Markov chain visits the state $[d_X(k)|X \in S]$. The Markov chain is said to be *positive recurrent* if it visits each reachable state infinitely many times as $k \to \infty$, and the mean time between two consecutive visits to any particular reachable state is finite.

Lemma 16. *Suppose the evolution of the state of a system can be described by a Markov chain under some policy η. The system is fulfilled by η if this Markov chain is positive recurrent.*

Proof. Since the Markov chain is positive recurrent, the state $\{d_X(k) = 0, \forall X \in S\}$ is visited infinitely many times. Further, assuming that the system is in this state at frames $k_1, k_2, k_3, \ldots$, then $[k_{n+1} - k_n]$ is a series of i.i.d. random variables with finite mean. Let I_n be the indicator variable that there exists some $\tilde{k}_n$ between frame k_n and frame k_{n+1} such that $d_X(\tilde{k}_n)/\tilde{k}_n > \delta$ for some $X \in S$, for some arbitrary $\delta > 0$. Let $q_{max}^* = \max_{X\in S} q_X^*$. If $I_n = 1$, we have that $\tilde{k}_n \geq k_n \geq n$ and thus $d_X(\tilde{k}_n) > n\delta$, for some $X \in S$. Since $d_X(k)$ can be incremented by at most q_{max}^* in a frame and $d_X(k_n) = 0$, we have $\tilde{k}_n - k_n > n\delta/q_{max}^*$ and

$k_{n+1} - k_n > \tilde{k}_n - k_n > n\delta/q^*_{max}$. Thus,

$$\begin{aligned} Prob\{I_n = 1\} &< Prob\{k_{n+1} - k_n > n\delta/q^*_{max}\} \\ &= Prob\{k_2 - k_1 > n\delta/q^*_{max}\}, \end{aligned}$$

and

$$\begin{aligned} \textstyle\sum_{n=1}^{\infty} Prob\{I_n = 1\} &< \textstyle\sum_{n=1}^{\infty} Prob\{k_2 - k_1 > n\delta/q^*_{max}\} \\ &\leq E[k_2 - k_1] < \infty, \end{aligned}$$

By the Borel-Cantelli Lemma, the probability that $I_n = 1$ for infinitely many n's is zero, and so is the probability that $d_X(k)/k > \delta$ for infinitely many k's. Thus, $\limsup_{k\to\infty} d_X(k)/k < \delta$ with probability 1, for all $X \in S$ and any arbitrary $\delta > 0$. Finally, we have $\lim_{k\to\infty} d_X(k)/k = 0$ with probability 1 since $d_X(k) \geq 0$ by definition. □

Based on the above lemmas, we determine a sufficient condition for a policy to be a p-approximation policy. The proof is based on the Foster-Lyapunov Theorem:

Theorem 26 (Foster-Lyapunov Theorem)**.** *Consider a Markov chain with state space $\mathcal{D}$. Let $D(k)$ be the state of the Markov chain at the k^{th} step. If there exists a non-negative function $L : \mathcal{D} \to R$, a positive number δ, and a finite subset $\mathcal{D}_0$ of $\mathcal{D}$ such that:*

$$\begin{aligned} E[L(D(k+1)) - L(D(k))|D(k)] &\leq -\delta, \textit{ if } D(k) \notin \mathcal{D}_0, \\ E[L(D(k+1))|D(k)] &< \infty, \textit{ if } D(k) \in \mathcal{D}_0, \end{aligned}$$

then the Markov chain is positive recurrent. ■

Theorem 27. *A policy η is a p-approximation policy, for some $p > 1$, if it schedules jobs solely based on the requirements and the state of a system, and, for each k, the following holds:*

$$\sum_{X\in S} \tilde{q}_X(k) d_X(k) \geq (\max_{[q_X]:\ [q_X] \textit{ is feasible}} \sum_{X\in S} q_X d_X(k))/p.$$

Proof. Consider a system with minimum reward requirements $[q^*_X]$ such that the same system with minimum reward requirements $[pq^*_X]$ is also strictly feasible. By Lemma 16, it suffices to show that under the policy η, the resulting Markov chain is positive recurrent. Consider the Lyapunov function $L(k) := \sum_{X\in S} d^2_X(k)/2$.

The Lyapunov drift function can be written as:

$$\Delta L(k+1) := L(k+1) - L(k) = \frac{1}{2}\sum_{X\in S}[d_X^2(k+1) - d_X^2(k)]$$

$$\leq \sum_{X\in S}(q_X^* - \tilde{q}_X(k))d_X(k) + C,$$

where C is a bounded constant. Since $[pq_X^*]$ is also strictly feasible and $d_X(k) \geq 0$, there exists $\epsilon > 0$ such that $(1+\epsilon)\sum_{X\in S} pq_X^* d_X(k) \leq \max_{[q_X]:\ [q_X] \text{ is feasible}} \sum_{X\in S} q_X d_X(k)$, and hence $(1+\epsilon)\sum_{X\in S} q_X^* d_X(k) \leq \sum_{X\in S} \tilde{q}_X(k) d_X(k)$. Thus, we have

$$\Delta L(k+1) \leq -\epsilon \sum_{X\in S} q_X^* d_X(k) + C. \tag{9.5}$$

Let $\mathcal{D}_0$ be the set of states $[d_X(k)|X \in S]$ with $\sum_{X\in S} q_X^* d_X(k) < (C+\delta)/\epsilon$, for some positive finite number δ. Then, $\mathcal{D}_0$ is a finite set (since $q_X^* > 0$ for all $x \in S$), with $\Delta L(k+1) < -\delta$ when the state of frame k is not in $\mathcal{D}_0$. Further, since $d_X(k)$ can be increased by at most q_X^* in each frame, $L(k+1)$ is finite if the state of frame k is in $\mathcal{D}_0$. By Theorem 26, this Markov chain is positive recurrent and policy η fulfills this system. □

Since a 1-approximation policy is also a feasibility optimal one, a similar proof yields the following:

Theorem 28. *A policy η fulfills a strictly feasible system if it maximizes $\sum_{X\in S} \tilde{q}_X(k) d_X(k)$ among all feasible $[q_X]$ in every frame k. It is a feasibility optimal policy if the above holds for all strictly feasible systems.*

9.4 An On-Line Scheduling Policy

While Section 9.3 has described a sufficient condition for designing feasibility optimal policies, the overhead for computing such a feasibility optimal scheduling policy may be too high to implement. In this section, we introduce a simple on-line policy. We also analyze the performance of this policy under different scenarios.

Theorem 28 has shown that a policy that maximizes $\sum_{X\in S} \tilde{q}_X(k) d_X(k)$ among all feasible $[q_X]$ in every frame k is feasibility optimal. The on-line policy follows this guideline greedily. Assume that, at some time t in frame k, task X has already been scheduled j_X times in its period. The on-line policy then schedules the task Y so that $r_Y^{j_Y+1} d_Y(k)$ is maximized among all $X \in S$. A more detailed description of this policy, which we call the *Greedy Maximizer*, is shown in Algorithm 10.

Next, we evaluate the performance of the Greedy Maximizer. We show that this policy is feasibility optimal if the periods of all tasks are the same, and that it is a 2-approximation policy in general.

Theorem 29. *The Greedy Maximizer fulfills all strictly feasible systems with $\tau_X \equiv \tau$, for all $X \in S$.*

Algorithm 10 Greedy Maximizer

Require: S, $\{\tau_X | X \in S\}$, $\{r_X^i | X \in S, 1 \le i \le \tau_X\}$, $\{q_X^* | X \in S\}$
$T \leftarrow$ least common multiplier of $\{\tau_X | X \in S\}$
for $X \in S$ **do**
  $d_X \leftarrow 0$
$t \leftarrow 0$
loop
  $t \leftarrow t + 1$
  if $t \mod T = 1$ {A new frame} **then**
    for $X \in S$ **do**
      $d_X \leftarrow [d_X + q_X^* - \tilde{q}_X]^+$
      $\tilde{q}_X \leftarrow 0$
      $j_X \leftarrow 0$
  for $X \in S$ **do**
    if $t \mod \tau_X = 1$ {A new period for X} **then**
      $j_X \leftarrow 0$
  $Y \leftarrow \arg\max_{X \in S} r_X^{j_X+1} d_X$
  $j_Y \leftarrow j_Y + 1$
  $\tilde{q}_Y \leftarrow \tilde{q}_Y + r_Y^{i_Y}$
  execute the job of Y at time t

Proof. It suffices to prove that the Greedy Maximizer indeed maximizes $\sum_{X \in S} d_X(k)\tilde{q}_X(k)$ in every frame. Suppose at some frame k, the debts are $\{d_X(k)\}$ and the schedule generated by the Greedy Maximizer is $\eta_{GM}(t)$, $t \in (kT, (k+1)T]$. Let $GM := \sum_{X \in S} d_X(k)\tilde{q}_X(k)$ when η_{GM} is applied. Consider another schedule, $\eta_{OPT}(t)$, that achieves Max$\sum_{X \in S} d_X(k)\tilde{q}_X(k) =: OPT$ in this frame. We need to show that $GM \ge OPT$.

We will modify $\eta_{OPT}(t)$ slot by slot until it is the same as η_{GM}. Let $\eta_{OPT}^i(t)$ be the schedule after we have made sure $\eta_{OPT}^i(t) = \eta_{GM}(t)$ for all t between kT and i, and let $OPT^i := \sum_{X \in S} d_X(k)\tilde{q}_X(k)$ when $\eta_{OPT}^i(t)$ is applied. We then have $\eta_{OPT} \equiv \eta_{OPT}^{kT}$ and $\eta_{GM} \equiv \eta_{OPT}^{(k+1)T}$. The process of modification is as follows: If $\eta_{OPT}^i(i+1) = \eta_{GM}(i+1)$, then we do not need to modify anything and we simply set $\eta_{OPT}^{i+1} \equiv \eta_{OPT}^i$. On the other hand, if $\eta_{OPT}^i(i+1) \ne \eta_{GM}(i+1)$, say, $\eta_{GM}(i+1) = (A, j_A)$ and $\eta_{OPT}^i(i+1) = (B, j_B)$, then we modify $\eta_{OPT}^i(t)$ under two different cases. The first case is that η_{OPT}^i is going to schedule the action (A, j_A) some time after $i+1$ in this frame. In this case η_{OPT}^{i+1} is obtained by switching the two actions (A, j_A) and (B, j_B) in η_{OPT}^i. One such example is shown in Fig. 9.3a. Since interchanging the order of actions does not alter the value of $\sum_{X \in S} d_X(k)\tilde{q}_X(k)$, we have $OPT^{i+1} = OPT^i$ for this case. The second case is that η_{OPT}^i does not schedule the action (A, j_A) in the frame. Then η_{OPT}^{i+1} is obtained by setting $\eta_{OPT}^{i+1}(i+1) = (A, j_A)$ and scheduling the same jobs as η_{OPT}^i for all succeeding time slots. Since the Greedy Maximizer schedules (A, j_A) in this slot, we have $r_A^{j_A} d_A(k) \ge r_B^{j_B} d_B(k)$. Also, for all succeeding time slots, if job B is scheduled, then the reward for that slot is going to be increased since the number of executions of job B has been decreased by 1; if a job C other than A and B is scheduled, then the reward for that slot is not altered by the modification. Fig. 9.3b illustrates one such example. In sum, we have that $OPT^{i+1} \ge OPT^i$.

We have established that $OPT^{i+1} \geq OPT^i$ for all $i \in [kT, (k+1)T]$. Since $OPT = OPT^{kT}$ and $GM = OPT^{(k+1)T}$, we have $GM \geq OPT$ and thus the Greedy Maximizer indeed maximizes $\sum_{X \in S} d_X(k)\tilde{q}_X(k)$.

□

However, when the periods of tasks are not the same, the Greedy Maximizer does not always maximize $\sum_{X \in S} d_X(k)\tilde{q}_X(k)$ and thus may not be feasibility optimal. An example is given below.

Example 4. *Consider a system with two tasks, A and B, with $\tau_A = 6$, $\tau_B = 3$. Assume that $r_A^1 = r_A^2 = r_A^3 = r_A^4 = 100$, $r_A^5 = r_A^6 = 1$, $r_B^1 = 10$, and $r_B^2 = r_B^3 = 0$. Suppose, at some frame k, $d_A(k) = d_B(k) = 1$. The Greedy Maximizer would schedule jobs as in Fig. 9.4a, and yield $d_A(k)\tilde{q}_A(k) + d_B(k)\tilde{q}_B(k) = 411$. On the other hand, a feasibility optimal scheduler would schedule jobs as in Fig. 9.4b, and yield $d_A(k)\tilde{q}_A(k) + d_B(k)\tilde{q}_B(k) = 420$.*

■

Although the Greedy Maximizer is not feasibility optimal, we can still derive an approximation bound for this policy.

Theorem 30. *The Greedy Maximizer is a 2-approximation policy.*

Proof. The proof is similar to that of Theorem 29. Define $\eta_{GM}, \eta_{OPT}, \eta_{OPT}^i, GM, OPT$, and OPT^i in the same way as in the proof of Theorem 29. By Theorem 27, it suffices to show that $GM \geq OPT/2$.

We obtain η_{OPT}^i as follows: If $\eta_{GM}(i+1) = \eta_{OPT}^i(i+1)$, then we set $\eta_{OPT}^{i+1} \equiv \eta_{OPT}^i$. If $\eta_{GM}(i+1) = (A, j_A) \neq (B, j_B) = \eta_{OPT}^i(i)$, then we consider three possible cases. The first case is that the job (A, j_A) is not scheduled by η_{OPT}^i in this period of A. In this case, we set $\eta_{OPT}^{i+1}(i+1) = (A, j_A)$ and use the same schedule as η_{OPT}^i for all succeeding time slots. An example is shown in Fig. 9.5a. The same analysis in Theorem 29 shows that $OPT^{i+1} \geq OPT^i$. The second case is that the job (A, j_A) is scheduled by η_{OPT}^i in this period of A and there is no deadline of B before the deadline of A. In this case, we obtain η_{OPT}^{i+1} by switching the jobs (A, j_A) and (B, j_B) in η_{OPT}^i. An example is shown in Fig. 9.5b. We have $OPT^i = OPT^{i+1}$ for this case. The last case is that the job (A, j_A) is scheduled by η_{OPT}^i in this period of A, and there is a deadline of B before the deadline of A. In this case also, we obtain η_{OPT}^{i+1} by switching the two jobs and renumbering these jobs if necessary. The rewards obtained by all tasks other than B are not altered by this modification. However, as in the example shown in Fig. 9.5c, the job (B, j_B) in η_{OPT}^i may become a job (B, j'_B) in η_{OPT}^{i+1} with $j'_B > j_B$. Thus, the reward obtained by B may be decreased. However, since rewards are non-negative, the amount of loss for B is at most $r_B^{j_B}$. By the design of Greedy Maximizer, we have $r_A^{j_A} d_A(k) \geq r_B^{j_B} d_B(k)$ and thus $OPT^{i+1} \geq OPT^i - r_A^{j_A} d_A(k)$.

In sum, for all i, if the Greedy Maximizer schedules (A, j_A) at time slot $i+1$, we have $OPT^{i+1} \geq OPT^i - r_A^{j_A} d_A(k)$. Thus, $GM = OPT^{(k+1)T} \geq OPT^{kT} - GM = OPT - GM$ and $GM \geq OPT/2$. □

Finally, we discuss the computation overhead of the Greedy Maximizer. We implement the Greedy Maximizer using a max heap where there is one node for each task X and the value for the node is $r_X^{j_X+1} d_X$. At the beginning of a new frame, that is, in steps 7–11 of Algorithm 10, the values for all nodes are updated and we need to reconstruct the max heap, which takes $O(|S| \log |S|)$ time. At the beginning of a period of task X (steps 12–14), we need to update the value for the node representing X. We can do this by first increasing the value to infinity and making it become the root node of the heap. We then remove the root from the heap, update its value, and insert this node back into the heap. These operations take a total of $O(\log |S|)$ time. Finally, in each time slot, the scheduling decisions (steps 15–17) involve finding the root of the max heap, removing the root from the heap, updating the value of the removed node, and inserting it back into the heap. These operations also take a total of $O(\log |S|)$ time. Thus, the Greedy Maximizer can be efficiently implemented.

9.5 Simulation Results

In this section, we present the simulation results. We consider the scenario of a server providing several video streams. For each video stream, there are 12 frames in a GOF, 1 I-frame, 3 P-frames, and 8 B-frames, as suggested in [3]. The server can serve one frame in each time slot.

We first consider a system where all streams have the same frame rate and they all generate one GOF every 30 time slots. We assume that there are two groups of streams, A and B, and each group has 3 different streams. Streams in group A have a higher quality demand, and require that both the I-frames and P-frames are treated as mandatory parts, thus only allowing B-frames to be treated as optional parts. Streams in group B only require I-frames to be mandatory and allow both P-frames and B-frames to be optional. We consider three types of rewards, exponential, logarithmic, and linear. Suppose the optional parts of the k^{th} stream in either group A or B are executed i times in a period. The stream gains a reward of $(5+k)(1-e^{-i/5})$ if the type of reward is exponential, $(5+k)\log(10i+1)$ if it is logarithmic, and $(5+k)i$ if it is linear.

We compare the set of reward requirements of streams that can be fulfilled by the Greedy Maximizer against the set of all feasible requirements. We also compare the policy introduced in [4], which aims to maximize the total per period reward, $\sum_{X \in S} q_X / \frac{T}{\tau_X}$, and thus we call it the MAX policy. To better illustrate the results, we assume that the per period reward requirements for streams in group A are all α and those for streams in group B are all β. We then find all pairs of (α, β) so that the resulting requirements are fulfilled by the evaluated policies and plot the boundaries of all such pairs. We call all pairs of (α, β) that are fulfilled by a policy as the *achievable region* of the policy. We also call the set of all feasible pairs of (α, β) as the *feasible region*.

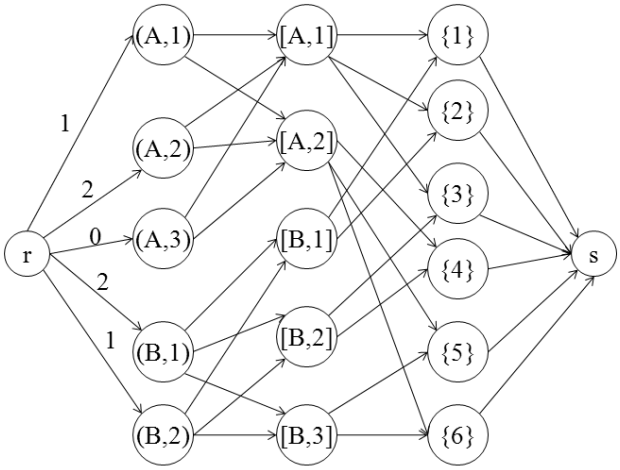
In each simulation of the Greedy Maximizer, we initiate the debt of X to be $M+1$, where we use the Big-M notation introduced in Section 9.1.1, and run the simulation for 20 frames to ensure that it has converged. We then continue to run the simulation for 500 additional frames. The system is considered fulfilled by the Greedy Maximizer if none of the mandatory parts miss their deadlines in the 500 frames, and the total reward obtained by each task is at least 0.996 times its requirement.

The simulation results are shown in Fig. 9.6. For both the cases of exponential and logarithmic functions, the achievable regions of the MAX policy are rectangles. That is because the MAX policy only aims at maximizing the total per-period rewards and does not allow any tradeoff between rewards of different tasks. The achievable regions of the MAX policy are also much smaller than the feasible regions. On the other hand, the achievable regions of the Greedy Maximizer are the same as the feasible regions for both the cases of exponential and logarithmic functions. This shows that the Greedy Maximizer is optimal when all tasks have the same period.

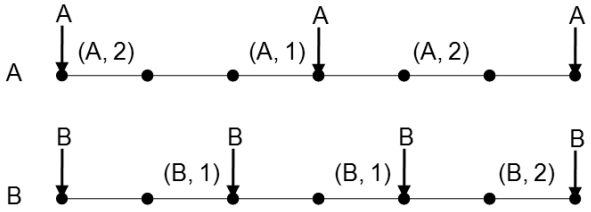
In the linear functions case, the MAX policy fails to fulfill any pairs of (α, β) except $(0, 0)$. A closer examination of the simulation result shows that, besides the mandatory parts, the MAX policy only schedules optional parts of the third tasks in groups A and B. This example shows that, in addition to restricted achievable regions, the MAX policy can also be extremely unfair. Thus, the MAX policy is not desirable when fairness is necessary. On the other hand, the achievable region of the Greedy Maximizer is almost the same as the feasible region.

Next, we simulate a system in which different streams may have different frame rates. We consider the same six streams as in the previous scenarios. The mandatory parts, optional parts, and rewards are the same as in the previous scenario. The only difference is that we assume that the first streams in both group A and B generate one GOF every 40 time slots, the second streams generate one GOF every 30 time slots, and the third streams generate one GOF every 20 time slots.

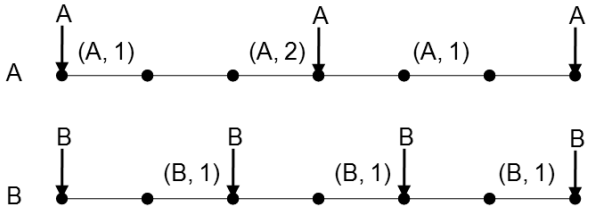
The simulation results are shown in Fig. 9.7. As in the previous simulations, the achievable regions of the Greedy Maximizer are always larger than those of the MAX policy, for all functions. Further, the achievable regions of the Greedy Maximizer are very close to the feasible regions. This suggests that, although we have only proved that the Greedy Maximizer is a 2-approximation policy, this approximation bound is indeed very pessimistic. In most cases, the performance of the Greedy Maximizer is close to a feasibility optimal one.



(a) The resulting flow network. Capacities for edges between the source and $L1$ are marked, while those for other edges are not shown for readability.

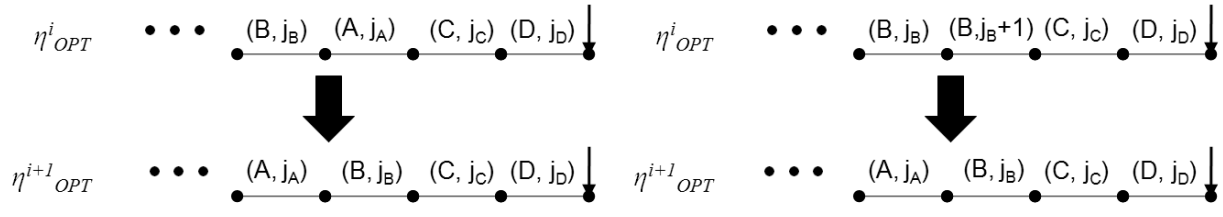


(b) The actions scheduled before renumbering



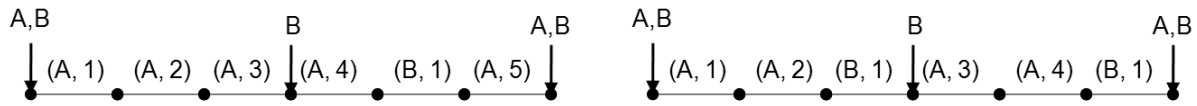
(c) The actions scheduled after renumbering

Figure 9.2: An example of the scheduling policy in Theorem 24.



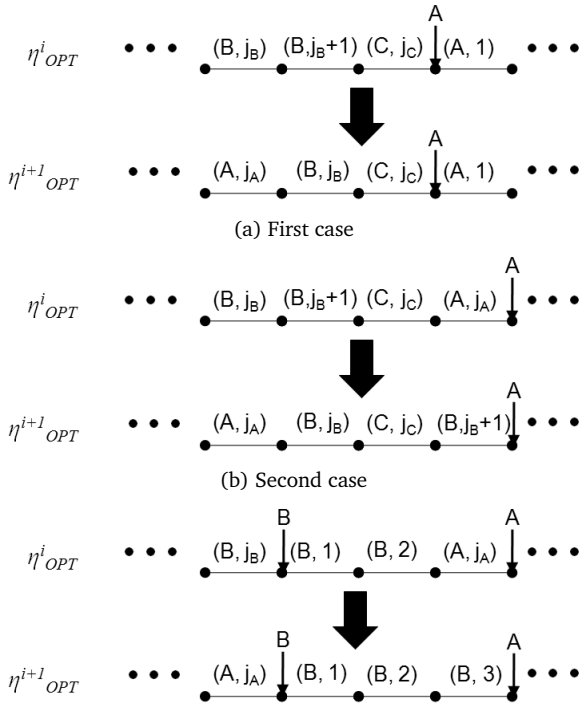
(a) First case (b) Second case

Figure 9.3: Examples of modification in Theorem 29



(a) Greedy Maximizer (b) Feasibility optimal scheduler

Figure 9.4: An example of the resulting schedule from the Greedy Maximizer, and a feasibility optimal scheduler, respectively, in Example 4.

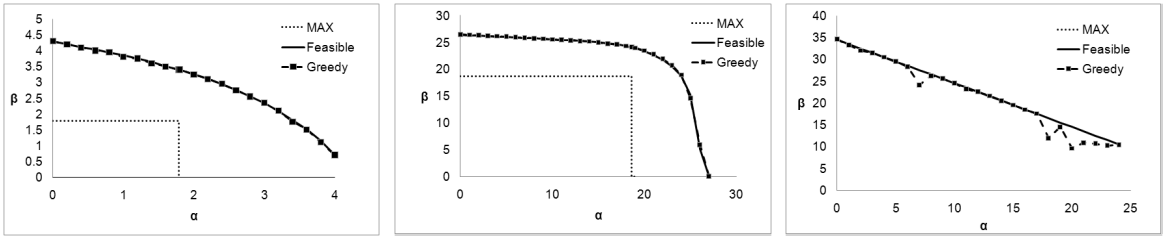


(a) First case

(b) Second case

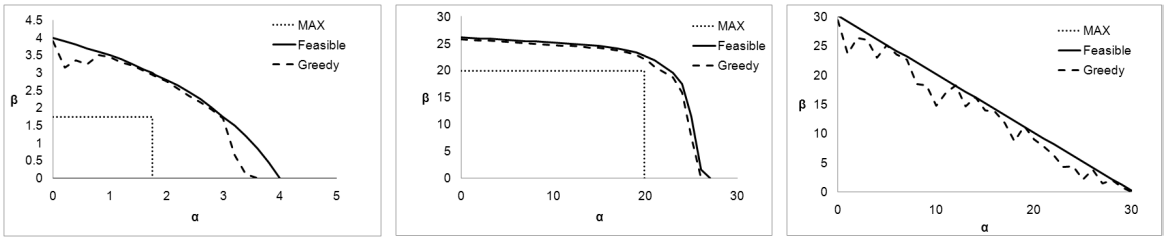
(c) Third case

Figure 9.5: Examples of modification in Theorem 30



(a) Exponential functions (b) Logarithmic functions (c) Linear functions

Figure 9.6: Achievable regions of scheduling policies for the system where all streams have the same frame rate.



(a) Exponential functions (b) Logarithmic functions (c) Linear functions

Figure 9.7: Achievable regions of scheduling policies for the system where different streams have different frame rates.

Chapter 10

Related Work

Providing services for flows with delay constraints over wiereless links is gaining extensive research interest. Stockhammer, Jenkac, and Kuhn [69] have studied the minimum initial delay and the minimum required buffer size for video streaming. Their study considers the case where there is only one wireless client in the system. Kang and Zakhor [39] have focused on improving the quality of video streaming by giving priorities to packets according to the content of the video. Li and Schaar [47] have proposed an adaptive algorithm for tuning the MAC retry limit for layered coded video. None of these works provides theoretical understanding on the three important problems for providing services: scheduling algorithms, admission control, and utility maximization.

Scheduling policies for QoS support on error-prone wireless channels have been increasingly of interest in recent years. Tassiulas and Ephremides [71] have proposed a max weight scheduling policy and proved that it is throughput optimal. Neely [56] has further evaluated this policy and shown that the policy achieves a constant average delay. Shakkottai and Stolyar [66] have evaluated various scheduling policies to support a mixture of real-time and non-real-time traffic. Johnsson and Cox [38] have proposed a policy that aims to achieve both small packet delay and high user throughput. Dua and Bambos [17] have focused on the trade-off between user fairness and system performance and designed a policy for this purpose. However, these works do not provide a thorough theoretical study with provable performance guarantees. Raghunathan et al [64] and Shakkottai and Srikant [65] have developed analytical results on scheduling. However, the goal of their works is to minimize the total number of expired packets among all users, which will inevitably be unfair to clients with poor channel qualities. Stolyar and Ramanan [70] aim at offering QoS guarantees on a per-client basis. Their approach offers asymptotic optimality only when the period is large. Kawata et al [40] have focused more on implementation issues and enhancing QoS for the IEEE 802.11 mechanisms. Their simulations have been conducted in a controlled environment where packet losses are infrequent. Fattah and Leung [21] and Cao and Li [11] have provided extensive surveys on scheduling policies for providing QoS.

Compared to scheduling policies, there are fewer analytical studies on admission control. Xiao et al [75]

and Pong el al [62] have proposed admission control algorithms to guarantee a certain amount of bandwidth for each user but do not take into account any latency constraints. Garg et al [25], Zhai et al [78], and Shin and Schulzrinne [68] have used various metrics to predict system performance statistically but lack a theoretical study. Gao, Cai, and Ngan [24], Niyato and Hossain [60], and Ahmed [1] have surveyed existing admission control algorithms in different types of wireless networks.

There is also research on utility maximization for both wireline and wireless networks. Kelly [41] and Kelly, Maulloo, and Tan [42] have considered the rate control algorithm to achieve maximum utility in a wireline network. Lin and Shroff [50] have studied the same problem with multi-path routing. As for wireless networks, Xiao, Shroff, and Chong [74] have proposed a power-control framework to maximize utility, which is defined as a function of the signal-to-interference ratio and cannot reflect channel unreliability. Cao and Li [10] have proposed a bandwidth allocation policy that also considers channel degradation. Bianchi, Campbell, and Liao [7] have studied utility-fair services in wireless networks. However, all the aforementioned works assume that the utility is only determined by the allocated bandwidth. Thus, they do not consider applications that require delay bounds.

Zhang and Du [79] have proposed a cross-layer design for multimedia broadcast. Raghunathan et al [64] have proposed scheduling policies for broadcasting delay-constrained flows. This work only focuses on minimizing the total number of expired packets and does not consider the different throughput requirements on different flows for each client. Gopala and El Gamal [28] have studied the tradeoff between throughput and delay of broadcasting. They have only studied the scaling laws of average delay, and thus their results are not applicable to scenarios where strict per-packet delay bounds are required. Zhou and Ying [80] have studied the asymptotic capacity of delay-constrained broadcast in mobile ad-hoc networks.

Network coding has emerged as a powerful technique to improve the capacity of wireless networks. Chaporkar and Proutiere [12] have proposed an adaptive network coding policy to improve throughputs of multi-hop unicast flows. Ghaderi , Towsley and Kurose [26] have quantified the reliability gain of network coding for broadcasting in unreliable wireless environments. Nguyen et al [58] have compared the throughputs of broadcast flows in systems employing network coding and those without network coding. Lucani, Medard, and Stojanovic [54] have analyzed the computational overhead of using different network coding schemes. Kozat [45] has studied the throughput capacity when erasure codes are employed. These works focus on throughputs and do not consider delays. Yeow, Hoang, and Tham [76] have focused on minimizing delay for broadcast flows by using network coding. Eryilmaz, Ozdaglar, and Medard [19] have studied the gain in delay performance resulting from network coding. Ying, Yang, and Srikant [77] have demonstrated that coding achieves the optimal delay-throughput tradeoff in mobile ad-hoc networks. These works only consider the performance of average delays and do not address strict per-packet delay bounds. Li, Wang,

and Lin [48] have studied a special case where a basestation is broadcasting delay-constrained flows to two clients. They demonstrate that using opportunistic network coding achieves the maximum asymptotic throughput for this special case. Both Pu el al [63] and Gangammanavar and Eryilmaz [23] have studied optimal coding strategies for broadcasting delay-constrained flows. Their works require the basestation to obtain feedback information from clients frequently, and thus may not be scalable.

Some works in the real-time system literature can be used to consider the problem of serving flows with delay constraints. The imprecise computation models [15, 51] have been proposed to handle applications in which partially completed jobs are useful. In this model, all jobs consist of two parts: a mandatory part and an optional part. The mandatory part needs to be completed before its deadline, or else the system suffers from a timing fault. On the other hand, the optional part is used to further enhance performance by either reducing errors or increasing rewards. The relations between the errors, or rewards, and the time spent on the optional parts, are described through error functions or reward functions. Chung, Liu, and Lin [15] have proposed scheduling policies that aim to minimize the total average error in the system for this model. Their result is optimal only when the error functions are linear and tasks generate jobs with the same period. Shih and Liu [67] have proposed policies that minimize the maximum error among all tasks in the system when error functions are linear. Feiler and Walker [22] have used feedback to opportunistically schedule optional parts when the lengths of mandatory parts may be time-varying. Mejia-Alvarez, Melhem, and Mosse [55] have studied the problem of maximizing total rewards in the system when job generations are dynamic. Chen et al [13] have proposed scheduling policies that defer optional parts so as to provide more timely response for mandatory parts. Zu and Chang [82] have studied the scheduling problem when optional parts are hierarchical. Aydin et al [4] have proposed an off-line scheduling policy that maximizes total rewards when the reward functions are increasing and concave. Most of these works only concern themselves with the maximization of the total reward in a system. Amirijoo, Hansson, and Son [2] have considered the tradeoff between data errors and transaction errors in a real-time database. The IRIS models can be thought of as special cases of the imprecise computation models where the lengths of mandatory parts are zero. Scheduling policies aimed at maximizing total rewards have been studied for such models [9, 16].

We have proposed an analytical model for providing services for flows with delay constraints through wireless networks [29] [35]. Based on this model, we have made progress on both admission control and scheduling algorithms [29] [35] [30]. We have obtained results for utility maximization, both for systems that do not employ rate adaptation [31], and for those that do [32]. We have studied the scheduling problem for broadcasting flows with delay constraints and incorporated the optional usage of network coding [33]. Finally, we have extended the imprecise computation models and the IRIS models in real-time system literature to consider the scheduling problem for a multimedia server [34].

Chapter 11

Conclusion

We have provided a framework for serving flows with delay constraints over unreliable wireless channels. We have proposed an analytical model that jointly consider the requirements of applications and the characteristics of wireless channels. Concerning the requirements of applications, this model allows clients to specify their traffic patterns, delay guarantees, and timely throughput requirements. Concerning the medium, this model allows for the heterogeneous, unreliable, and time-varying nature of wireless channels, as well as systems with rate adaptation and without. Therefore, this model can accommodate a wide range of practical scenarios.

We have studied three important problems: admission control, packet scheduling, and utility maximization. We have provided a sharp characterization of whether it is feasible to fulfill the demands of all clients, as well as a polynomial-time algorithm for admission control under some restrictions. We have established a framework for designing feasibility-optimal scheduling policies, and have derived tractable on-line scheduling policies for a variety of scenarios.

We have also developed solutions to the problem of utility maximization by designing policies that achieve the maximum total utility, for both systems with rate adaptation and without. In addition, we have taken into account the possible selfish behaviors of clients, and have proposed an auction mechanism that prevents clients from gaining additional benefits by lying about their utility functions.

We have also extended our model to consider the scheduling problem for broadcasting flows with delay constraints. This extended model can incorporate various network coding mechanisms. We have proposed a general framework for designing scheduling policies under any network coding mechanism. We have demonstrated the usage of this framework by deriving policies for a number of different network coding mechanisms, including XOR coding and linear coding.

Finally, we have considered the scheduling problem for a multimedia server, where packets of the same flow may not be equally important. We have extended the imprecise computation model and the increasing reward with increasing service model in the real-time systems literature for this context. We have provided a sharp characterization of feasibility. We have also developed a feasibility-optimal scheduling policy, as well

as an on-line scheduling policy that achieves an approximation bound.

References

[1] M.H. Ahmed. Call admission control inwireless networks: A comprehensive survey. *IEEE Communications Surveys*, 7(1):50–69, 2005.

[2] Mehdi Amirijoo, Jorgen Hansson, and Sang Hyuk Son. Specification and management of QoS in real-time databases supporting imprecise computations. *IEEE Transactions on Computers*, 55(3):304–319, March 2006.

[3] Axis Communications. H.264 video compression standard.

[4] Hakan Aydin, Rami Melhem, Daniel Mosse, and Pedro Mejia-Alvarez. Optimal reward-based scheduling for periodic real-time tasks. *IEEE Transactions on Computers*, 50(2):111–130, February 2001.

[5] M. Bazaraa, H. Sherali, and C. Shetty. *Nonlinear programming: theory and algorithms*. Wiley-Interscience, 2006.

[6] P. Bhagwat, P. Bhattacharya, A. Krishma, and S. K. Tripathi. Using channel state dependent packet scheduling to improve TCP throughput over wireless LANs. *Wireless Networks*, 3(1):91–102, 1997.

[7] G. Bianchi, A. Campbell, and R. Liao. On utility-fair adaptive services in wireless networks. In *Proc. of IWQoS*, pages 256–267, 1998.

[8] David Blackwell. An analog of the minimax theorem for vector payoffs. *Pacific J. Math*, 6(1), 1956.

[9] Hasan Cam. An on-line scheduling policy for IRIS real-time composite tasks. *The Journal of Systems and Software*, 52:25–32, 2000.

[10] Y. Cao and V. Li. Utility-oriented adaptive QoS and bandwidth allocation in wireless networks. In *Proc. of ICC*, 2002.

[11] Y. Cao and V.O.K. Li. Scheduling algorithms in broadband wireless networks. *Proceedings of the IEEE*, 89(1):76–87, Jan. 2001.

[12] Prasanna Chaporkar and Alexandre Proutiere. Adaptive network coding and scheduling for maximizing throughput in wireless networks. In *Proceedings of ACM MobiCom*, pages 135–146, 2007.

[13] Jia-Ming Chen, Wan-Chen Lu, Wei-Kuan Shih, and Ming-Chung Tang. Imprecise computations with deferred optional tasks. *Journal of Information Science and Enginnering*, 25(1):185–200, 2009.

[14] Y.S. Chow and H.Teicher. *Probability theory, independence, interchangeability, martingales*. Springer-Verlag, 1988.

[15] Jen-Tao Chung, Jane W. S. Liu, and Kwei-Jay Lin. Scheduling periodic jobs that allow imprecise results. *IEEE Transactions on Computers*, 39(9):1156–1174, September 1990.

[16] Jayanta K. Dey, James Kurose, and Don Towsley. On-line scheduling policies for a class of IRIS (increasing reward with increasing service) real-time tasks. *IEEE Transactions on Computers*, 45(7):802–813, July 1996.

[17] A. Dua and N. Bambos. Deadline constrained packet scheduling for wireless networks. In *62nd IEEE VTC Fall*, 2005.

[18] E. O. Elliot. Estimates of error rates for codes on burst-noise channels. *Bell Syst. Tech. J.*, 42:1977–1997, 1963.

[19] A. Eryilmaz, A. Ozdaglar, and M. Medard. On delay performance gains from network coding. In *Proc. of CISS*, pages 864–870, 2006.

[20] I. V. Martin F., J.J. Alins-Delgado, M. Aguilar-Igartua, and J. Mata-Diaz. Modelling an adaptive-rate video-streaming service using Markov-rewards models. In *Proc. of QSHINE*, pages 92–99, 2004.

[21] H. Fattah and C. Leung. An overview of scheduling algorithms in wireless multimedia networks. *IEEE Wireless Communications*, 9(5):76–83, Oct. 2002.

[22] Peter H. Feiler and John J. Walker. Adaptive feedback scheduling of incremental and design-to-time tasks. In *Proceedings of the 23rd International Conference on Software Engineering*, pages 318–326, 2001.

[23] H. Gangammanavar and A. Eryilmaz. Dynamic coding and rate-control for serving deadline-constrained traffic over fading channels. In *Proc. of IEEE ISIT*, pages 1788–1792, 2010.

[24] D. Gao, J. Cai, and K.N. Ngan. Admission control in IEEE 802.11e wireless LANs. *IEEE Network*, pages 6–13, July/August 2005.

[25] S. Garg and M. Kappes. Admission control for VoIP traffic in IEEE 802.11 networks. In *Proc. of GLOBECOM*, 2003.

[26] M. Ghaderi, D. Towsley, and J. Kurose. Reliability gain of network coding in lossy wireless networks. In *Proc. of IEEE INFOCOM*, pages 2171–2179, 2008.

[27] E. N. Gilbert. Capacity of a burst-noise channel. *Bell Syst. Tech. J.*, 39:1253–1265, 1960.

[28] P.K. Gopala and H. El Gamal. On the throughput-delay tradeoff in cellular multicast. In *Proceedings of the Symposium on Information Theory in WirelessCom*, 2005.

[29] I-H. Hou, V. Borkar, and P. R. Kumar. A theory of QoS for wireless. In *Proc. of IEEE INFOCOM*, 2009.

[30] I-H. Hou and P. R. Kumar. Scheduling heterogeneous real-time traffic over fading wireless channels. In *Proc. of IEEE INFOCOM*, 2010.

[31] I-H. Hou and P. R. Kumar. Utility maximization for delay constrained qos in wireless. In *Proc. of IEEE INFOCOM*, 2010.

[32] I-H. Hou and P. R. Kumar. Utility-optimal scheduling in time-varying wireless networks with delay constraints. In *Proc. of ACM MobiHoc*, 2010.

[33] I-H. Hou and P. R. Kumar. Broadcasting delay-constrained traffic over unreliable wireless links with network coding. In *Proc. of ACM MobiHoc*, 2011.

[34] I-H. Hou and P. R. Kumar. Scheduling periodic real-time tasks with heterogeneous reward requirements. In *Proc. of IEEE RTSS*, 2011.

[35] I-H. Hou and P.R. Kumar. Admission control and scheduling for QoS guarantees for variable-bit-rate applications on wireless channels. In *Proc. of ACM MobiHoc*, pages 175–184, 2009.

[36] ITU-T. Pulse Code Modulation (PCM) of voice frequencies. *ITU-T Recommendations*, 1988.

[37] ITU-T. G.729 based Embedded Variable bit-rate coder: An 8-32 kbit/s scalable wideband coder bitstream interoperable with G.729. *ITU-T Recommendations*, 2006.

[38] K. B. Johnsson and D. C. Cox. An adaptive cross-layer scheduler for improved QoS support of multiclass data services on wireless systems. *IEEE J. on Selected Areas in Communications*, 23(2), 2005.

[39] S. H. Kang and A. Zakhor. Packet scheduling algorithm for wireless video streaming. In *PV*, 2002.

[40] T. Kawata, S. Shin, A. G. Forte, and H. Schulzrinne. Using dynamic PCF to improve the capacity for VoIP traffic in IEEE 802.11 networks. In *Proc. of IEEE WCNC*, 2005.

[41] F. Kelly. Charging and rate control for elastic traffic. *European Trans. on Telecommunications*, 8:33–37, 1997.

[42] F.P. Kelly, A.K. Maulloo, and D.K.H. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49:237–252, 1998.

[43] H. J. Kishner and G. G. Yin. *Stochastic Approximation Algorithms and Applications*. Springer-Verlag, 1997.

[44] Bernhard Korte and Jens Vygen. *Combinatorial Optimization, Theory and Algorithms*. Springer-Verlag Berlin Heifelberg, 2008.

[45] U.C. Kozat. On the throughput capacity of opportunistic multicasting with erasure codes. In *Proceedings of IEEE INFOCOM*, pages 520–528, 2008.

[46] L.J. De la Cruz and J. Mata. Performance of dynamic resources allocation with QoS guarantees for MPEG VBR video traffic transmission over ATM networks. In *Proc. of GLOBECOM*, pages 1483–1489, 1999.

[47] Q. Li and M. van der Schaar. Providing adaptive QoS to layered video over wireless local area networks through real-time retry limit adaptation. *IEEE Trans. on Multimedia*, 6(2):278–290, 2004.

[48] X. Li, C.-C. Wang, and X. Lin. Throughput and delay analysis on uncoded and coded wireless broadcast with hard deadline constraints. In *Proc. of IEEE INFOCOM*, 2010.

[49] X. Lin and N. Shroff. Joint rate control and scheduling in multihop wireless networks. In *Proc. of IEEE CDC*, pages 1484–1489, 2004.

[50] X. Lin and N.B. Shroff. Utility maximization for communication networks with multipath routing. *IEEE Trans. on Automated Control*, 51(5):766–781, 2006.

[51] Jane W.S. Liu, Kwei-Jay Lin, Wei-Kuan Shih, and Albert Chuang-Shi Yu. Algorithms for scheduling imprecise computations. *Computers*, 24(5):58–68, May 1991.

[52] M. Loeve. *Probability Theory*. Litton Educational Publishing, 1963.

[53] M. Loeve. *Probability Theory II*. Springer-Verlag, 1978.

[54] D.E. Lucani, M. Medard, and M. Stojanovic. Systematic network coding for time-division duplexing. In *Proceedings of IEEE ISIT*, pages 2403–2407, 2010.

[55] Pedro Mejia-Alvarez, Rami Melhem, and Daniel Mosse. An incremental approach to scheduling during overloads in real-time systems. In *Proceedings of the 21st IEEE Real-Time Systems Symposium*, pages 283–293, 2000.

[56] M. Neely. Delay analysis for max weight opportunistic scheduling in wireless systems. In *Proc. of Allerton Conf.*, 2008.

[57] J. Neveu. *Discrete parameter maringales*. North-Holland, 1975.

[58] Dong Nguyen, Tuan Tran, Thinh Nguyen, and B. Bose. Wireless broadcast using network coding. *IEEE Transactions on Vehicular Technology*, 58(2):914–925, 2009.

[59] N. Nisan, T. Roughgarden, E. Tardos, and V.V. Vazirani. *Algorithmic game theory*. Cambridge University Press, 2007.

[60] D. Niyato and E. Hossain. Call admission control for QoS provisioning in 4G wireless networks: issues and approaches. *IEEE Network*, pages 5–11, September/October 2005.

[61] D. O'Neill, A. Goldsmith, and S. Boyd. Wireless network utility maximization. In *Proc. of IEEE Milcom*, 2008.

[62] D. Pong and T. Moors. Call admission control for IEEE 802.11 contention access mechanism. In *Proc. of GLOBECOM*, 2003.

[63] W. Pu, C. Luo, F. Wu, and C. W. Chen. QoS-driven network coded wireless multicast. *IEEE Transactions on Wireless Communications*, 8(11):5662–5670, 2009.

[64] V. Raghunathan, V. Borkar, M. Cao, and P.R. Kumar. Index policies for real-time multicast scheduling for wireless broadcast systems. In *Proc. of IEEE INFOCOM*, 2008.

[65] S. Shakkottai and R. Srikant. Scheduling real-time traffic with deadlines over a wireless channel. *Wireless Networks*, 8(1), Jan. 2002.

[66] S. Shakkottai and A. L. Stolyar. Scheduling algorithms for a mixture of real-time and non-real-time data in HDR. In *Proc. of 17th International Teletraffic Congress (ITC-17)*, 2001.

[67] Wei-Kuan Shih and Jane W.S. Liu. Algorithms for scheduling imprecise computations with timing constraints to minimize maximum error. *IEEE Transactions on Computers*, 44(3):466–471, March 1995.

[68] S. Shin and H. Schulzrinne. Call admission control in IEEE 802.11 WLANs using QP-CAT. In *Proc. of INFOCOM*, 2008.

[69] T. Stockhammer, H. Jenkac, and G. Kuhn. Streaming video over variable bit-rate wireless channels. *IEEE Trans. on Multimedia*, 6(2):268–277, April 2004.

[70] A. L. Stolyar and K. Ramanan. Largest weighted delay first scheduling: Large deviations and optimality. *Ann. Appl. Probab.*, 11(1), 2001.

[71] L. Tassiulas and A. Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Trans. on Information Theory*, 39(2):89–103, 1993.

[72] Texas Instruments. Low power advantage of 802.11a/g vs. 802.11b. 2003.

[73] H.S. Wang and N. Moayeri. Finite-state Markov channel – a useful model for radio communication channels. *IEEE Trans. on Vehicular Technology*, 44(1):163–171, 1995.

[74] M. Xiao, N.B. Shroff, and E. Chong. A utility-based power-control scheme in wireless cellular systems. *IEEE/ACM Trans. on Networking*, 11(2):210–221, 2003.

[75] Y. Xiao and H. Li. Evaluation of distributed admission control for the IEEE 802.11e EDCA. *Communications Magazine, IEEE*, 42(9), Sept. 2004.

[76] Wai-Leong Yeow, Anh Tuan Hoang, and Chen-Khong Tham. Minimizing delay for multicast-streaming in wireless networks with network coding. In *Proceedings of IEEE INFOCOM*, pages 190–198, 2009.

[77] L. Ying, S. Yang, and R. Srikant. Coding achieves the optimal delay-throughput trade-off in mobile ad-hoc networks: Two-dimensional i.i.d. mobility model with fast mobiles. In *Proc. of WiOpt*, pages 1–10, 2007.

[78] H. Zhai, X. Chen, and Y. Fang. A call admission and rate control scheme for multimedia support over IEEE 802.11 wireless LANs. *Wireless Networks*, 12(4), August 2006.

[79] X. Zhang and Q. Du. Cross-layer modeling for QoS-driven multimedia multicast/broadcast over fading channels in mobile wireless networks. *IEEE Communications Magazine*, 45(8):62–70, 2007.

[80] S. Zhou and L. Ying. On delay constrained multicast capacity of large-scale mobile ad-hoc networks. In *Proc. of IEEE INFOCOM*, 2010.

[81] X. Zhou, S. Gandhi, S. Suri, and H. Zheng. eBay in the sky: strategy-proof wireless spectrum auctions. In *Proc. of ACM MobiCom*, pages 2–13, 2008.

[82] Ming Zu and Albert M. K. Chang. Real-time scheduling of hierarchical reward-based tasks. In *Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 2–9, 2003.