

© Copyright by Thyagarajan Nandagopal, 2002

SCHEDULING FOR FAIRNESS
AND MINIMAL RESPONSE TIMES
IN WIRELESS DATA NETWORKS

BY

THYAGARAJAN NANDAGOPAL

B.E., Anna University, 1997

M.S., University of Illinois at Urbana-Champaign, 2000

M.S., University of Illinois at Urbana-Champaign, 2002

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2002

Urbana, Illinois

ABSTRACT

The rapid growth of the Internet has led to the emergence of wireless data networks that enable truly nomadic computing. These wireless networks can be categorized into two complementary types: (a) cellular networks, where a base station connects mobile devices to the Internet, and (b) ad hoc networks, where mobile devices form a distributed network among themselves. With the exponential increase in the number of wireless data users, providing quality of service (QoS) in these networks becomes critical. Scheduling at the MAC layer enables us to provide fine-grained QoS to users. In this thesis, we focus on the design of scheduling algorithms to address various QoS issues in cellular and ad hoc networks.

Ad hoc networks are decentralized and do not have the notion of a base station to control channel access. Simultaneous transmissions and location-dependent contention make it difficult to characterize the rates that hosts and flows can obtain in an ad hoc network. We develop a framework to model the nature of wireless channels and obtain a distributed algorithm that can be used to realize a desired fair rate structure in the ad hoc network. A large class of fairness models are applicable to our work.

Next generation 3G/4G cellular data networks allow multiple codes (or channels) to be allocated to a single user, where each code can support multiple data rates. Providing fine-grained QoS to users in such networks poses the two-dimensional challenge of assigning both power (rate) and codes for every user. This gives rise to a new class of parallel scheduling problems. We abstract a communication-theoretic model for multirate wireless channels, and attempt to optimize the maximum response time of jobs, a QoS metric suitable for a stream of user requests. Using a novel approach called *resource augmented*

competitive analysis, we present provable results on the algorithmic complexity of these scheduling problems. In particular, we are able to provide very simple, online algorithms for approximating the optimal response time.

We believe that using our analytical techniques will result in simple, efficient and practical scheduling algorithms for providing QoS in wireless data networks.

ACKNOWLEDGMENTS

I am greatly indebted to my adviser, Professor Vaduvur Bharghavan, for his help, guidance and encouragement during the course of my doctoral research. I am thankful to him for having given me an opportunity to work with him and his other students.

I also thank my mentors at AT&T Labs, Dr. Suhas Diggavi and Dr. S. Muthukrishnan, for enabling an interesting exchange of ideas, and for their help with my dissertation. On the same note, I wish to thank Professor Luca Becchetti, Professor Stefano Leonardi, Professor Alberto Marchetti-Spaccamela, and Andrea Vitaletti at the Department of Systems and Informatics in the University of Rome, Italy, for their valuable help in completing this thesis.

My thanks also goes to my fellow group members at The Illinois Mobile Environments Laboratory (TIMELY), especially, Prasun Sinha, Tae-eun Kim, Kang-won Lee, Jeffrey Monks, Songwu Lu, Jia-ru Li, Narayanan Venkitaraman, Raghupathy Sivakumar and Xia Gao for providing support and motivation during my studies.

I wish to thank Professor Nitin Vaidya and the members of my committee: Professor P. R. Kumar, Professor R. Srikant, and Professor S. Lumetta for their helpful hints and suggestions.

Last but not the least, I would like to thank my parents, my sister, and my fiancée for their love and support.

To Amma, Appa, Anu, and Radhu

TABLE OF CONTENTS

CHAPTER	PAGE
1 INTRODUCTION	1
1.1 Wireless Data Networks	1
1.2 A Framework for Fairness in Ad Hoc Networks	4
1.3 Scheduling in Next Generation Cellular Networks	8
1.4 Contributions and Structure of Dissertation	12
2 ACHIEVING MAC LAYER FAIRNESS IN WIRELESS PACKET DATA NETWORKS	14
2.1 Multiple Access MAC: Contention Resolution	14
2.2 Fairness Characteristics of IEEE 802.11 MAC	17
2.2.1 Asymmetric contention neighborhoods	18
2.2.2 Per-flow versus per-node fairness	19
2.2.3 Randomly generated topology	21
2.3 A General Analytical Framework for Fairness	22
2.3.1 Steps 1, 2, and 3	24
2.3.2 Step 4: Modeling fairness	27
2.4 Proportionally Fair Contention Resolution (PFCR)	30
2.4.1 Proportional fairness and rate control	31
2.4.2 A local mechanism for proportional fairness	31
2.4.2.1 Protocol details	33
2.4.2.2 Practical considerations	35
2.5 Performance of the PFCR Algorithm	35
2.5.1 Example 1	36
2.5.2 Example 2	37
2.5.3 Example 3	38
2.5.4 Example 4	40
2.6 Summary and Future Work	42
3 PARALLEL SCHEDULING PROBLEMS IN NEXT GENERATION WIRELESS NETWORKS	45
3.1 Problem Formulation	45
3.1.1 Communication channel model	45

3.1.2	Abstract scheduling problem	46
3.1.3	Malleable task scheduling	49
3.2	Understanding the Scheduling Problems	50
3.2.1	Some structural observations	50
3.2.2	Computational hardness	51
3.2.3	Offline scheduling problem	55
3.3	Online Heuristics	58
3.3.1	Minimizing the maximum response time	59
3.3.2	The discrete case	65
3.3.3	Other QoS criteria	68
3.4	Simulation Study	70
3.4.1	Online algorithms	70
3.4.2	Simulation setup	72
3.4.2.1	Channel specifications	72
3.4.2.2	Data sets and simulation tools	72
3.4.2.3	Simulation tools	73
3.4.3	Experiments	73
3.4.3.1	Unbounded maximum response time	74
3.4.3.2	Online heuristics	75
3.4.3.3	Resource augmentation	79
3.5	Discussion	83
4	RELATED WORK	85
4.1	Fairness and Scheduling in Ad Hoc Networks	85
4.2	Scheduling in Next Generation Cellular Networks	86
5	CONCLUSIONS AND FUTURE RESEARCH	89
	REFERENCES	92
	VITA	98

LIST OF TABLES

Table	Page
2.1 Example 1: Total number of packets transmitted	36
2.2 Example 2: Total number of packets transmitted	37
2.3 Example 3: Total number of packets transmitted	39
2.4 Example 4: Total number of packets transmitted	40
3.1 Offline scheduling programs	55
3.2 Unbounded nature of $FIFO_{cont}(P/C)$ - one slot OPT max-flow	74
3.3 Unbounded nature of $FIFO_{cont}(P/C)$ - large sizes	75
3.4 Unbounded nature of $FIFO_{cont}(P/C)$ - max-flow for large sizes	75
3.5 Online heuristics: config3	77
3.6 Online heuristics: max-flow	78

LIST OF FIGURES

Figure	Page
1.1 Scheduling scenario for cellular data networks.	11
2.1 Topology: Asymmetric contention.	19
2.2 Packet sending timing: Flow 11 exhibits short-term unfairness.	19
2.3 Topology: Per-flow vs. per-node fairness.	20
2.4 Packet sending timing: Flow 5 receives more bandwidth than other flows. Flows 1-4 exhibit short-term unfairness.	20
2.5 Network topology where all flows share the channel.	21
2.6 Fairness in shared channel: Flows 4 and 11 experience varying throughput at different intervals.	22
2.7 Network graph G : No two neighbors can simultaneously transmit.	24
2.8 Flow-contention graph G' : In every clique, only one node can transmit.	25
2.9 Resource contention graph G'' : Each maximal clique is a server and each flow is a client.	26
2.10 Pseudo code for the PFCR algorithm.	33
2.11 State transition diagram for the PFCR algorithm.	34
2.12 Flow contention graph G' for Example 1.	36
2.13 Example 1 - Relative throughput of various algorithms compared to pro- portional fair allocation.	37
2.14 Flow contention graph G' for Example 2.	38
2.15 Contention clique graph G'' for Example 2.	38
2.16 Example 2 - Relative throughput of various algorithms compared to pro- portional fair allocation.	39
2.17 Flow contention graph G' for Example 3.	40
2.18 Contention clique graph G'' for Example 3.	40
2.19 Example 3 - Relative throughput of various algorithms compared to pro- portional fair allocation.	41
2.20 Flow contention graph G' for Example 4.	42
2.21 Example 4 - Relative throughput of various algorithms compared to pro- portional fair allocation.	43
3.1 Qualcomm HDR rate structure.	47

3.2	Summary of analytical results.	68
3.3	On-line heuristics: Comparison with optimal max-flow.	76
3.4	On-line heuristics: Large traces.	77
3.5	Normalized maximum response time and demand reduction with resource augmentation: <i>FIFO continuous</i>	79
3.6	Normalized maximum response time and demand reduction with resource augmentation: <i>2D-FIFO</i>	80
3.7	Normalized maximum response time and demand reduction with resource augmentation: <i>2D-PIKI</i>	80
3.8	Normalized maximum response time and demand reduction with resource augmentation for large traces: <i>2D-FIFO</i>	82
3.9	Normalized maximum response time and demand reduction with resource augmentation for large traces: <i>2D-PIKI</i>	82

CHAPTER 1

INTRODUCTION

This dissertation focuses on the design, analysis, and evaluation of algorithms for providing quality of service (QoS) through link layer scheduling for various wireless data networks. In this chapter, we present an overview of the various wireless systems to be discussed and identify key problems in such systems. We also present a high-level overview of our solutions and research contributions, and provide an outline for the remainder of this dissertation.

1.1 Wireless Data Networks

The enormous popularity of the Internet and its rapid growth have made the vision of a “wired world” a reality. Apart from large organizations and universities, individual homes have come to occupy a significant portion of the Internet user space. This widespread availability has fueled the extension of the Internet to a totally different domain: *wireless*.

Wireless data connectivity is a very appealing concept that promises to release us from being tied to one place to access data. The promise of “untethered” computing has led to a boom in many areas, from the usage of wireless data enabled gadgets such as laptops, personal digital assistants, and cellular phones, to the design and deployment of various kinds of wireless data networks for an increasingly mobile population. The number of wireless data users is expected to grow at an exponential rate for the next decade, with a fourfold increase within 2 years to about 500 million users worldwide [1]. Modest projections of mobile commerce revenues estimate a growth of 700% to \$4 billion by the year 2005 [2]. It is clear that wireless data is here to stay.

In a wireless system, the physical medium of communication is the space surrounding us. The appeal of wireless computing is in the fact that users can be reached without a fixed communication channel that ties the user to the network. On the other hand, an open medium is prone to disturbances which degrade communications. Fades, blackouts, and random bit errors are typical characteristics of any wireless system. Wireless users move around and also share the channel, experiencing variable signal strength, which in turn affects the quality of their connection. These channel characteristics vary across wireless systems, from low-mobility in-building technologies such as Bluetooth [3] to high-mobility long range wireless networks such as cellular digital packet data (CDPD) [4]. Thus, wireless networking technologies have to address the dual issues of spatial and temporal channel variations, being aware of the fact that it might be impossible to optimize for all wireless systems.

Wireless data networks can be broadly classified into two categories: centralized networks and decentralized networks. Centralized networks are based on a cellular infrastructure, where a base station covers all users within a given geographical region, also known as a *cell*, with a larger area covered by a network of base stations connected by a backbone wired network. Cells could be partially overlapping; however, neighboring cells coordinate on resolving conflicts with resource usage and interference. Each base station handles all requests to and from mobile users within the cell; i.e., a base station handles both uplink (from mobile users) and downlink (to mobile users) requests. Both uplink and downlink channel performances are crucial for the overall system. Our focus here is on the downlink channel performance, which is likely to be a major focus in emerging systems since data traffic is expected to dominate over time, and data traffic typically tends to have asymmetrically large downlink demand. There are two common types of wireless networks that fall under this category, namely wireless local area networks (WLANs) [5] (which include Lucent's WaveLAN and Cisco's Aironet LAN), and wireless wide area networks (WWANs) [6] such as CDPD, High Data Rate (HDR) [7], and the proposed third generation (3G) wireless systems. WLANs have transmission ranges of the order of a few hundred meters and bandwidths of upto 50 Mb/s, whereas WWANs have a

range of the order of a couple of kilometers and bandwidths of less than 1 Mb/s. Due to transmission range constraints, WLANs are used in buildings and inter-building spaces, whereas WWANs provide coverage in large open areas.

Legacy WWANs such as CDPD provide users with raw bandwidths of less than 20 kb/s. The next generation of WWANs under deployment, commonly referred to as 3G networks, support a peak bandwidth of up to 1 Mb/s with the average bandwidth expected to be around 200 kb/s. These systems have unique characteristics that differentiate them from conventional centralized wireless networks, such as the ability for a single mobile host to simultaneously receive on multiple channels and the ability to transmit at multiple rates on any given channel depending on the signal-to-noise ratio (SNR) at the receiver. This gives the base station more flexibility than is available in current systems to manage and modulate the traffic. Designing algorithms that make use of this added flexibility to improve the QoS provided to mobile hosts is an important issue.

Decentralized networks, also known as *ad hoc wireless networks*, lack a fixed infrastructure and centralized control. An ad hoc network is a dynamic multihop wireless network that is established by a group of mobile nodes on a shared wireless channel by virtue of their proximity to each other. Ad hoc networks have widespread applications in military, commercial and public safety environments. Many commercial wireless technologies have been proposed using the ad hoc network paradigm. Bluetooth networks, personal area networks (PANs), home RF networking, and sensor networks are some of the many variants of ad hoc networks that are in the development phase.

Four key properties set ad hoc networks apart from centralized wireless networks:

- *Lack of infrastructure*: These networks do not require an infrastructure, and do not have any notion of base stations or cells.
- *Variable topology*: The existence of a link between any two mobile hosts depends on the distance between them and the transmission range of the radio used. The topology may thus change with mobility of the hosts.

- *Shared channels*: Unlike cellular networks, channel access is decentralized, and the number of neighbors for a mobile host is location-dependent. Therefore, the actual bandwidth realized by a mobile host is typically a fraction of the raw bandwidth.
- *Inaccurate state*: In cellular networks, the base station has knowledge of all the mobile hosts in its cell. In an ad hoc network, this is not possible since the neighborhood set of any mobile host is typically only a subset of the entire network.

Given these characteristics, the throughput that each mobile host receives in the ad hoc network is highly variable even if all mobile hosts run the same applications and talk to a particular host. Such deviation in service will result in dissatisfaction on part of the user. It also affects the performance of higher layer protocols and applications. Thus, there is an urgent need to characterize and quantify the service received by mobile hosts in an ad hoc network.

Based on the discussion of the various wireless data networks above, we observe that several issues need to be studied and analyzed for these networks. In particular, we need to have a way of characterizing the throughput of mobile hosts in an ad hoc network in terms of network parameters. This work is important to further the understanding of ad hoc networks and aid their commercial deployment. We also noted the interesting properties of 3G cellular data networks which point out for the need to develop algorithms that provide improved QoS making use of these features than existing algorithms. In this dissertation, we describe our work in the design, characterization, analysis, and simulation of these algorithms, and provide an overview in the following sections.

1.2 A Framework for Fairness in Ad Hoc Networks

In recent years, wireless ad hoc networks have increasingly received critical attention in the networking research community. Although most current nonmilitary ad hoc network test-beds are experimental in nature, possible future deployment scenarios include deeply networked conglomerations of embedded devices, emergency rescue operations,

“zero conf” meeting setups, and rapidly reconfigurable metropolitan wireless networks. Migrating from experimental environments to commercial environments, ad hoc network designers will need to address critical new challenges, such as “service differentiation” among contending users for the dynamic and scarce channel resources. In a pay-for-use model, the network must provide minimum performance requirements for paying users, at least in relative terms. Since link-layer fairness mechanisms serve as the basis for achieving network-layer QoS (e.g., weighted fair queueing [8] for the IntServ guaranteed QoS service model), wireless medium access control (MAC) protocols in commercial ad hoc networks must support some notion of “weighted fairness,” wherein flows with larger weights receive correspondingly better service in accordance with a system-wide fairness model.

To this end, *we aim to formally investigate the fairness properties that can be achieved by the class of multiple access wireless MAC protocols in shared channel wireless networks in general, and ad hoc networks in particular.* We define a *shared wireless channel* as a communication regime wherein all nodes communicate over the same logical channel using decentralized control, and there is no concept of a base station in the MAC layer. Shared wireless channels thus underlie both ad hoc networks and packet cellular networks, and most wireless multiple access protocols [9, 10], including the basic IEEE 802.11 MAC standard [11], are designed with these channel assumptions.

Naturally, the first question to ask is, “Is there something fundamental about the nature of shared wireless channels that prevents us from reusing the wealth of link-layer fairness techniques that have been developed for wireline and packet cellular environments?” As discussed earlier, it turns out that shared channel wireless networks have three unique characteristics that make it very difficult to achieve, or even consistently define, the notion of fairness:

1. *Spatial (location-dependent) contention for the wireless channel:* Consider a simple channel model where a transmitter has a fixed transmission range, and multiple transmissions in the neighborhood of a receiver will cause a collision at the receiver. Following typical collision avoidance protocols [9, 10, 11], a successful transmission

precludes any station in the neighborhood of either the transmitter or the receiver from engaging in another simultaneous packet transmission/reception. In other words, *transmission of a packet involves contention over the joint neighborhoods of the sender and the receiver*, and the level of contention for the shared wireless channel in a geographical region is *spatially dependent* on the number of contending nodes in the region. This is fundamentally different from wireline and cellular channel models, wherein all flows perceive the same contention.

2. *Trade-off between channel utilization and fairness*: In a shared channel wireless network, *spatial reuse* of the channel bandwidth may be obtained by simultaneously scheduling transmissions whose regions of contention are not in conflict. While spatial reuse is very useful for increasing the utilization of the wireless channel, it introduces a fundamental conflict between optimizing aggregate allocated bandwidth and achieving fairness, because allocating the channel to a flow with a large contention correspondingly reduces the channel reuse. In contrast, wireline and cellular networks do not face this problem because all flows perceive the same contention.
3. *Inaccurate state and decentralized control*: Even if we define the notion of fairness by addressing the two issues above, designing mechanisms for achieving such fairness is a major challenge since there is no centralized control and no station is guaranteed to have accurate knowledge of the contention even in its own neighborhood. Further, contention is really “per-flow” (i.e., a sender-receiver pair) rather than per-node (see point 1 above), which makes its estimation harder at the transmitter before it decides to contend for the channel. Finally, contention resolution must be achieved without assuming any explicit coordination or handshakes among the contenders in order to preserve the robustness of multiple access protocols.

To summarize, wireline and cellular fairness models are inappropriate for our target environment because these channel characteristics are fundamentally unique to shared wireless channels.

In the past decade, there has been a wealth of wireless MAC research focusing on the design and evaluation of multiple access wireless MAC protocols. However, to the best of our knowledge, there are no structured studies devoted to the formal investigation of fairness models in ad hoc wireless networks, evaluation of competing fairness models, or formal fairness characterizations of existing contention resolution mechanisms in wireless MAC protocols. To address these limitations in the state of the art, this dissertation proposes what we believe to be the first structured study of fairness models in shared channel multiple access wireless networks.

Our first contribution is a *general analytical framework that captures the unique characteristics of shared wireless channels* identified above, and allows us to reason about a large class of fairness models. Our starting point is the recognition that link-layer fairness in a shared wireless channel has some commonalities with network-layer fairness in multihop wireline networks in the sense that different “flows” experience different “contention.”¹ The genesis of our framework lies in the fairness framework for network flows proposed by Kelly et. al. [12], which we enhance to address the constraints of link-layer contention resolution in wireless channels. Using this framework, we show that the definition of fairness is equivalent to specifying a “utility function” for the channel allocation for each flow, and that different fairness models can be achieved by enforcing correspondingly different utility functions locally at each contending station without explicit global coordination.

Our second contribution is a *general mechanism for translating a given fairness model into a corresponding backoff-based collision resolution algorithm that probabilistically achieves the fairness objective*. This is a powerful result because it shows that once we model fairness in our framework, the backoff algorithm for achieving this fairness model is automatically derived from the framework! Using this translation, we derive the backoff algorithm for achieving *proportional fairness* [12] in wireless shared channels,

¹A link layer flow is between a pair of neighboring nodes, and has a location-dependent contention for channel allocation. A network layer flow is between a pair of end hosts, and has a path-dependent “contention” for network bandwidth.

as a representative example. We compare the fairness properties of this backoff algorithm to the ideal proportional fairness objective, and fairness properties of IEEE 802.11 MAC, the multiple access carrier sense for wireless (MACAW) algorithm, and a more recent work called connection-based balanced MAC (CB-Fair) [13].

We believe that the two aspects of the proposed framework – (a) the ability to specify arbitrary fairness models (by specifying the corresponding utility function), and (b) the ability to automatically generate local collision resolution mechanisms in response to a given utility function – jointly provide the path for achieving flexible service differentiation in future ad hoc wireless networks.

1.3 Scheduling in Next Generation Cellular Networks

Current wireless cellular data networks are categorized as 2.5G, where data rates are of the order of 56 kb/s. New 3G systems are being deployed to provide much bandwidths up to an order of magnitude more than existing systems. These emerging wireless systems allow multiple codes (channels) to be allocated to users, in each of which traffic can flow in one of multiple rates. This provides them more flexibility than is available in current systems to manage and modulate the traffic. This also gives rise to novel parallel scheduling problems that we study in this chapter.

Our scheduling model abstracts multirate scheduling in many next generation wireless data systems. Here, we will briefly review three examples, namely, code division multiple access (CDMA), wideband orthogonal frequency division multiplexing (OFDM), and multislots time-division multiple access (TDMA) systems.

- **CDMA:** In CDMA systems, all the users share the entire transmission bandwidth and users are distinguished by the use of *signatures* (also called *codes*) assigned to them. There are two main CDMA schemes under implementation in the 3G wireless systems. One is *cdma2000* which is an evolution of Qualcomm's IS-95 second generation (2G) CDMA system and is designed to use a frequency spectral bandwidth of 1.25 MHz. The other is the *wideband CDMA* (WCDMA), which uses a larger

frequency spectral bandwidth of 5 MHz and is being specified in Europe and Japan [14]. Both cdma2000 and WCDMA envision higher rates through assigning multiple codes to users (*code-aggregation*) and variable rates through coding techniques. These two systems are intended for both data and voice applications. Another system of significant interest is a purely data system called the *High Data Rate* (HDR) system designed by Qualcomm [15]. This system is designed with a large range of available data rates and uses sophisticated error-correcting coding schemes with higher latency, making it suitable for non-real-time data traffic. In all CDMA proposals, there is a pilot signal in the broadcast control channel that enables the access terminal to measure the link channel conditions and this is reported back to the base station. Therefore, the base station has access to transmission conditions for users typically at a time-scale of a few milliseconds per measurement.

- **Wideband OFDM:** In wideband OFDM [16], the wideband channel is divided into narrow frequency *tones* in a manner similar to traditional frequency-division multiplexing. However, there are important differences since the transmission frame is assembled using a fast fourier transform (FFT) in OFDMs. As a result, transmissions become less susceptible to multipath propagation effects [16]. Also, the use of FFT allows the system to dynamically assign different sets of tones to different users. Moreover, using powerful error-correcting codes, variable rates can be allocated on the tones. Though this is not among the 3G wireless standards, it is receiving significant research and industry attention. The idea of OFDM is already a part of European digital audio broadcast standards, the US wireline digital subscriber line (DSL) standards [17], the HiperLan wireless LAN standard, and several wireless local loop systems. Systems with number of tones ranging from 16 to 256 are currently being studied [16]. Our algorithms will be applicable to OFDM systems with codes interpreted as tones.
- **Multislot TDMA:** In an evolution of the popular *global system mobilé* (GSM) standard used in Europe [14], a multislot TDMA system called *enhanced data*

rates for GSM evolution (EDGE) is being proposed. This system has a proposed eight time-slots per frame and variable data rates on each slot through adaptive coding schemes. Therefore, it is possible to allocate to a given user both a variable number of slots and varying rates in each slot. The packet data protocol stack being proposed is called the general packet radio service (GPRS), and enhanced GPRS (EGPRS) [14]. We will not go into more details of this system here, but this could potentially be studied in the framework proposed in this dissertation.

In all of the cellular data networks discussed above, data traffic on the downlink (from the base station to the mobile hosts) is expected to dominate data traffic in a cell. Hence, providing consistent QoS to mobile users in the downlink is clearly important. However, wireline scheduling and resource allocation algorithms cannot be directly applied to manage the downlink. Wireless networks have unique characteristics, an example being location-dependent channel errors. Users in different regions of a cell experience different error rates, and hence, they may get different data rates on the downlink. Unlike traditional scheduling scenarios, in a wireless environment, the scheduler must consider channel state in order to provide reasonable QoS, and wireless systems have a variety of built-in capabilities to gather channel condition information for this purpose. In most wireless networks, the base station receives channel state piggybacked on ACKs and other control packets from the users [18], whereas current and proposed CDMA networks obtain channel state by means of an instantaneous power control loop [6].

In addition, random channel errors also result in performance problems for transport protocols such as TCP, since TCP typically interprets channel errors as errors caused by congestion. Link layer retransmissions only aggravate the situation since they interfere with TCP's *rtt* computations [19]. Most solutions to this problem propose intercepting the connection at the base station, creating two logical connections [19, 20]. The base station thus acts as a proxy and interprets the packets up to the transport layer in order to address random channel errors. Proxies are also used in many commercial wireless networks for a different purpose – to address the difference in bandwidth between the wireless and the wired links, which is of at least one order in magnitude. Proxy

servers store and forward data to mobile users and thus have information regarding the various requests in the system. Thus, there is some convergence in the base station and proxy servers aimed at reducing performance problems of transport protocols like TCP, while making use of request sizes, channel state and other information to manage the wireless link more effectively for scheduling resources. A typical proxy server/base station architecture is shown in Figure 1.1. We will assume this context and address the scheduling problems that arise.

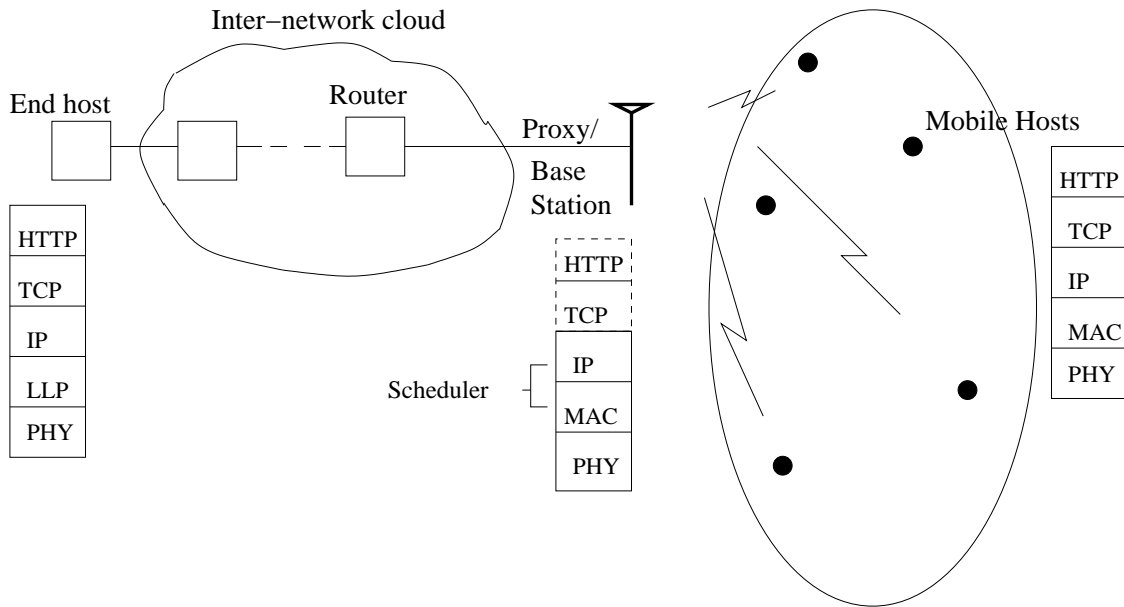


Figure 1.1 Scheduling scenario for cellular data networks.

Our contributions are threefold:

- We abstract a general downlink scheduling problem that has many novelties. For example, we embody channel characteristics guided by communication theoretic considerations, and the properties of these channels are exploited in our scheduling algorithms. Second, we study QoS parameters related to per request behavior, in particular, we focus on optimizing response time per-request. In contrast, prior work in wireless systems scheduling has typically focused on rate optimization metrics.

- The scheduling problems that arise above are novel versions of multidimensional malleable parallel scheduling. Here one can think of the number of channels as the number of processors and the malleability arises because the rates on each channel changes depending on power allocated. We show these problems to be NP-complete and also show that they are also hard to approximate. However, we use resource augmented competitive analysis, to derive simple, online algorithms which not only are practical but also have good performance in approximating the optimal maximum response time of a job.
- We present an experimental study of our algorithms. Using real Web server request logs and realistic 3G/4G system parameters, we show experimentally that our online algorithms perform significantly better than our worst-case analyses indicate. We also infer which resources are most important to augment in a wireless setting.

We believe that resource augmented competitive analysis, which is related to over-provisioning, provides us with a new approach to solving many open problems, and will help in designing interesting scheduling algorithms for next generation cellular networks.

1.4 Contributions and Structure of Dissertation

The key contributions of this dissertation are as follows.

- In our work on ad hoc wireless data networks (Chapter 2), we have proposed an unique general framework that captures the unique characteristics of shared wireless channels and use this model for imposing a structure on the nature of throughput obtained by individual flows in an ad hoc network. We also propose a very practical mechanism that relies on only local state to obtain a given fairness model, using a backoff-based contention resolution algorithm. Our experimental studies verify our analytical results.
- We study a class of job scheduling problems that arise in 3G/4G cellular data networks (Chapter 3), and show that these problems have non-trivial solutions. In

particular, we apply a new type of analytical technique called resource augmented competitive analysis to these wireless systems to design practical online algorithms that perform well when compared to the optimum.

The rest of the dissertation is organized as follows. Chapter 2 presents our fairness model for ad hoc wireless networks. Chapter 3 discusses our work on 3G systems and the novel scheduling problems therein. In Chapter 4, we present related work pertaining to scheduling and fairness for the various wireless systems discussed in this dissertation. Chapter 5 summarizes the results, identifies future directions for exploration, and concludes the dissertation.

CHAPTER 2

ACHIEVING MAC LAYER FAIRNESS IN WIRELESS PACKET DATA NETWORKS

We propose an analytical model to capture the unique characteristics of shared wireless channels and develop a mechanism to translate any fairness model into a corresponding distributed contention resolution algorithm. We illustrate the effectiveness of this mechanism using the proportional fairness model. In this chapter, we describe representative fairness mechanisms from related literature in Section 2.1. Section 2.2 shows inconsistencies in the fairness of IEEE 802.11, a widely used medium access control protocol standard, and Section 2.3 presents a general analytical framework for specifying fairness in shared channel networks. Section 2.4 presents the translation procedure from the fairness model to the specific backoff-based contention resolution algorithm, using proportional fairness as an example. Section 2.5 evaluates the backoff algorithm with reference to the ideal objective as well as related work. Section 2.6 summarizes the chapter.

2.1 Multiple Access MAC: Contention Resolution

In this chapter, our investigations focus on the class of shared channel multiple access collision avoidance protocols in the carrier sense multiple access with collision avoidance (CSMA/CA) family. While we will investigate the fairness properties of multiple access protocols for general shared channel configurations, this work is more applicable to ad hoc networks than cellular networks, wherein other techniques for ensuring fairness, such as centralized scheduling at the base station, may be overlaid on top of the multiple access

MAC protocol [18]. Fairness in multihop wireless networks has also been addressed in a recent related work [21, 22]. This work focuses on maximizing aggregate channel reuse subject to a minimum fairness guarantee, which is very different from the issues we seek to address in this dissertation.

Wireless multiple access protocols typically have two components that work in concert: (a) collision avoidance and (b) contention resolution. Most of the wireless multiple access research in the past decade has focused on achieving perfect collision avoidance [9, 10, 23], and we will not address this component further in this dissertation. For the rest of the work, we will assume some form of efficient collision avoidance that almost completely eliminates non-contention induced collisions [10, 23].

The second component, contention resolution, is the focus of this work. In multiple access protocols, contention resolution has typically been achieved through two mechanisms: (a) *backoff* and (b) *persistence*. In the backoff mechanism, each contending station maintains a “backoff counter” and defers for a random amount of time bounded by the backoff counter prior to a transmission. In the persistence mechanism, each contending station maintains a “persistence probability” and contends for the channel with this probability when it perceives a clear channel. In both cases, multiple simultaneous transmissions in a shared region cause collisions, and the goal is to adjust the backoff counter/ persistence parameters appropriately so that collisions are reduced, and ideally, eliminated. Thus, both the backoff counter and the persistence probability are functions of the estimated contention, and different contention resolution algorithms differ in terms of how they adjust these parameters in response to collisions and successful transmissions. In fact, the key to achieving the desired fairness properties is to map the fairness objective to a corresponding persistence/backoff adjustment algorithm. Deriving this mapping is a challenging exercise, which we will address in Section 2.4.

Let us now briefly consider three representative contention resolution mechanisms. Most of the multiple access protocols in related literature employ mechanisms that are variants of one or more of these three mechanisms.

- *Binary exponential backoff (BEB)*: Following the contention resolution scheme of Ethernet [24], early CSMA/CA protocols used a mechanism wherein the backoff counter of a station is doubled upon every contention loss, and reset to 1 upon a successful completion of the packet handshake. Unfortunately, it has been shown that BEB is highly short-term unfair under high offered loads [24]. The IEEE 802.11 MAC protocol adopts a variant of BEB by starting from a “base backoff counter” value for each new packet, and then using binary exponential backoff for subsequent retransmissions of the packet subject to a maximum upper bound (31 and 1023 respectively for DS PHY) [11]. While this alleviates the short-term unfairness of BEB to some extent, it induces more collisions and unpredictable fairness properties in the presence of heavy contention (Section 3 investigates these issues further).
- *Multiplicative increase/linear decrease (MILD) with backoff copy*: MACAW uses a more sophisticated contention resolution algorithm, wherein it tries to “assign” contention losses to each station and then computes a backoff “per-flow” rather than per-node. The backoff adjustment algorithm at a station maintains a backoff counter per flow, doubles the backoff counter for the flow on each loss assigned to the station, and reduces the backoff counter by 1 for each successful transmission. MACAW also has a “backoff copy” mechanism wherein transmitters advertise their backoff counters in their packets, and idle stations snoop this backoff to keep track of the current contention estimates in their region [9]. The stated goal of MACAW is to set the backoff counter of a flow proportional to the contention experienced by the flow [9]. Unfortunately, this ends up being highly unfair to flows experiencing relatively high contention in asymmetric network configurations because such flows experience contend less aggressively while also experiencing contention from a larger number of flows (Examples 2 and 3 in Section 2.5 illustrate this phenomenon).
- *Combining persistence and backoff*: In CB-Fair [13], the authors design a contention resolution algorithm that combines persistence and backoff. A transmitter contends

with a persistence probability that is proportional to the ratio of the degree of the receiver to the maximum degree in the transmitter’s neighborhood (more generally, the persistence probability is a function of the transmitter’s second neighborhood as well as the receiver’s neighborhood). The backoff adjustment algorithm doubles the backoff counter on each loss and halves the backoff counter on each successful transmission. CB-Fair also uses backoff copying similar to MACAW. It turns out that the multiplicative nature of both the increase and decrease of backoff causes short-term unfairness similar to BEB. Further, the persistence algorithm tries to increase the persistence of highly contending flows, which leads to artificially higher backoffs and highly inconsistent short-term behavior over different time windows (Examples 2 and 3 in Section 2.5 illustrate this phenomenon).

We have only presented brief high-level descriptions of these three algorithms (though we have simulated the algorithms in detail for the purposes of comparison in Section 2.5) because what we really want to show is that, regardless of the details, these mechanisms are based on somewhat ad hoc reasonings about fairness rather than a structured realization of a formally defined fairness model. In the next section, we explore the fairness properties of the IEEE 802.11 MAC protocol standard in more detail, using it as a representative study. We show that because it does not have a clearly defined fairness model, the protocol has inconsistent fairness behavior over different time windows and is, in fact, both short-term and long-term unfair. Both MACAW and CB-Fair also exhibit similar characteristics (see Section 2.5 for examples). This motivates our approach, which starts with an arbitrary fairness model and ends with a contention resolution algorithm design that achieves the fairness model probabilistically.

2.2 Fairness Characteristics of IEEE 802.11 MAC

We now illustrate the fairness properties of IEEE 802.11 through three tests in the *ns-2* simulator [25]. The simulation scenarios consist of both simple hand-configured topologies and more complex randomly generated topologies. The carrier sense/data reception

threshold ratio is set appropriately to eliminate all hidden/exposed sender/receiver problems [23], and no random channel loss is induced. Thus, all losses are due to contention. Each flow has enough packets to individually saturate the channel for the duration of the simulation to simulate the highly loaded case. The wireless channel bandwidth is 2 Mb/s (approximately 180 packets per second for 1 KB packets). The duration of each simulation is 10 s. We will show the channel allocation over small 1- to 2-s time windows as well as over the entire simulation in order to observe both the short-term and long-term fairness characteristics.

We perform three tests. In the first test, we observe the inconsistency of short-term fairness in IEEE 802.11 in a simple scenario where contention in a neighborhood is asymmetric, i.e., nodes experience location-dependent contention. In the second test, we observe the unfairness in per-node versus per-flow fairness in IEEE 802.11 in a simple scenario where one node communicates with many other nodes. In the third test, we illustrate both the short-term and long-term unfairness in IEEE 802.11 in a more complex, randomly generated topology. The conclusion of this study is that the 802.11 MAC protocol is unable to achieve any consistent notion of fairness and is both short-term and long-term unfair.

2.2.1 Asymmetric contention neighborhoods

Consider the topology shown in Figure 2.1. The receiver threshold range is denoted by a solid circle, and the dotted circle denotes the carrier-sensing range. All flows in 1-6 contend with each other and with Flow 11. Flows 9 and 10 contend with flows 9-11. Flow 11 contends with all the flows in the topology. The contention resolution algorithm is the modified binary exponential backoff described in Section 2.1. Figure 2.2 shows the channel allocation for the time window $(2, 4.5]$ for a few flows. The key point to note is the variation in the channel allocation for Flow 11. Specifically, the throughput ratio $R_{1-6} : R_{11} = 6.5$ in $(2, 3.5]$, and $R_{1-6} : R_{11} = 2.5$ in $(3.5, 4.5]$. In more complex simulation scenarios, asymmetric contention can cause larger variations in short-term

and long-term fairness, as seen from the third example as well as the examples in Section 2.5.

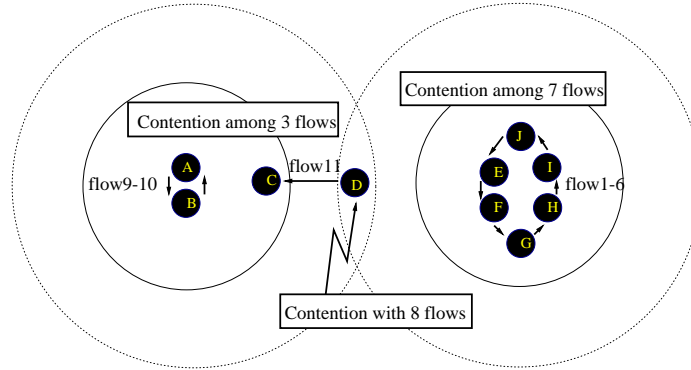


Figure 2.1 Topology: Asymmetric contention.

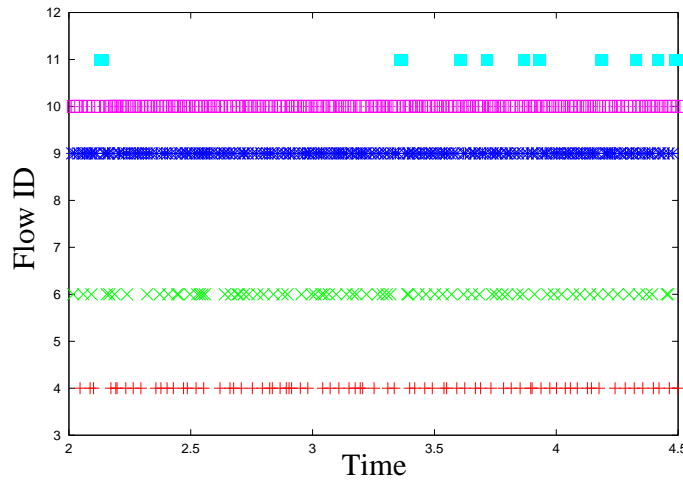


Figure 2.2 Packet sending timing: Flow 11 exhibits short-term unfairness.

2.2.2 Per-flow versus per-node fairness

While protocols such as MACAW and CB-Fair use per-flow queues with per-flow backoffs (thereby effectively treating a node with multiple flows as multiple co-located nodes with one flow each), IEEE 802.11 uses a per-node queue with a per-node backoff. This has two effects: (a) in scenarios with asymmetric contention, a head-of-line packet to a receiver in a high contention neighborhood can block other flow transmissions to

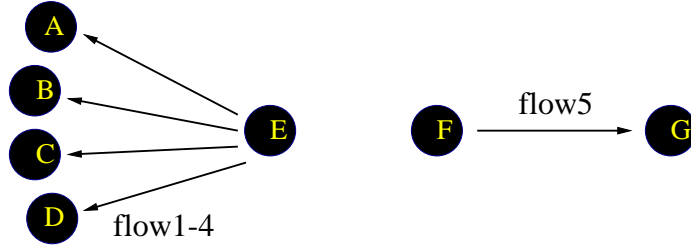


Figure 2.3 Topology: Per-flow vs. per-node fairness.

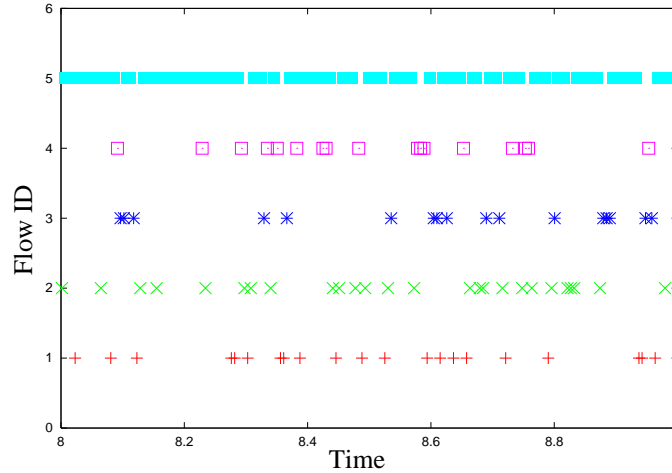


Figure 2.4 Packet sending timing: Flow 5 receives more bandwidth than other flows. Flows 1-4 exhibit short-term unfairness.

lightly loaded neighbors, thereby distorting the fairness characteristics of the network as a whole, and (b) a node with many flows penalizes its flows unfairly. Figure 2.3 shows a simple scenario wherein all flows contend with each other, and Figure 2.4 shows the channel allocation for this scenario. It is important to note that we are not arguing about whether this specific fairness model is desirable or not. It may well be that the system fairness policy mandates per-node rather than per-flow fairness. The issue here is that IEEE 802.11 cannot achieve weighted fairness in a reasonable way, even if we try to weight the backoff increase. In other words, the protocol is implicitly able to provide only per-node fairness, and that too falls apart in the presence of asymmetric contention neighborhoods. This behavior is illustrated in the next example.

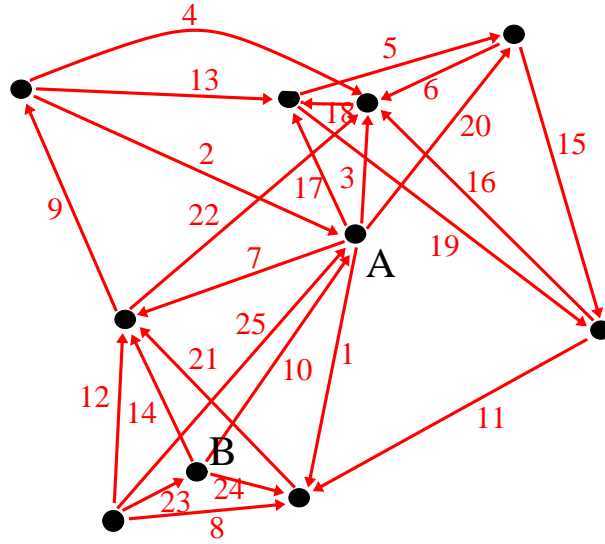


Figure 2.5 Network topology where all flows share the channel.

2.2.3 Randomly generated topology

Figure 2.5 shows a randomly generated topology consisting of 10 nodes spread over a 400 m by 400 m grid in a way that the network is not a clique but only one flow can transmit at a time. There are 25 flows for which the source and receiver are within the receiving threshold range, and all flows start between 0 to 4 s and run for 50 s. Figure 2.6 shows the channel allocation for a few selected flows in this simulation. This example illustrates both short-term and long-term unfairness, both per-node fairness and per-flow fairness. Specifically, simply looking at the throughput evolution of flows 4 and 11 in two time intervals (20, 25] and (30, 35] illustrates the different fairness characteristics observed in different time intervals. Per-flow unfairness in both the short-term and the long-term is obvious from the figure. The allocation also happens to be per-node unfair, because node A receives significantly higher channel allocation than node B, even though every node has the same contention region.

This example illustrates that even for a relatively simple scenario of a few nodes dispersed in a geographical region with relatively small asymmetry in contention, the

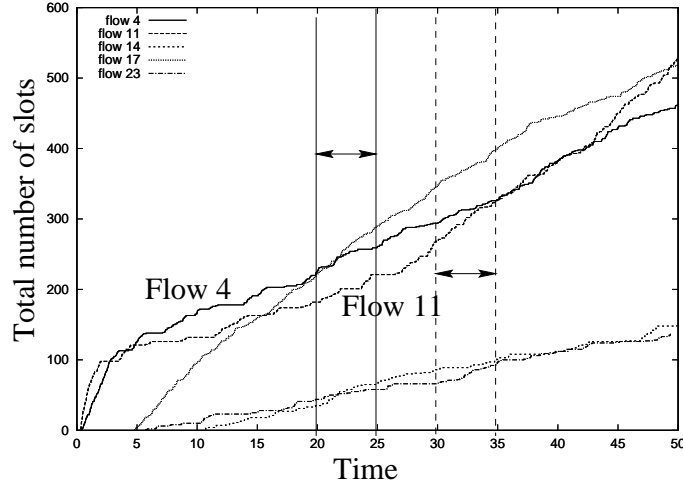


Figure 2.6 Fairness in shared channel: Flows 4 and 11 experience varying throughput at different intervals.

fairness of the IEEE 802.11 MAC protocol starts to fall apart. We show in Section 2.5, that both MACAW and CB-Fair suffer from the same problems. The fundamental issue here is not whether any of these protocols achieves a certain definition of fairness in a given configuration over a given time window. The real problem is that none of the wireless multiple access protocols that we have come across in recent literature have a well defined formal definition of the fairness model that they seek to achieve. Thus, it is difficult to quantify their fairness properties for arbitrary topologies. In the next section, we seek to move the design of fairness mechanisms from arcane art to well defined science.

2.3 A General Analytical Framework for Fairness

In wireline and cellular environments, all flows experience the same contention. Specifying a fairness model is thus fairly straight-forward, e.g., the fairness model of weighted fair queueing (WFQ) [8] states that over any arbitrarily small time window, each backlogged flow receives channel allocation in proportion to its weight. Unfortunately, simply using flow weights to determine the channel allocation in shared wireless channels does not adequately capture the unique characteristics of the shared channel. Recall from

Section 1 that because of spatial location-dependent contention, different flows have different resource utilization for transmitting the same amount of data. Specifically, flows that experience more contention will “shut up” more contending flows while they are transmitting. Consequently, a general framework for modeling fairness for shared wireless channels must provide the ability to capture the location-dependent nature of contention. Within this framework, each fairness model can then trade-off aggregate channel utilization and fairness according to its own optimization criteria.

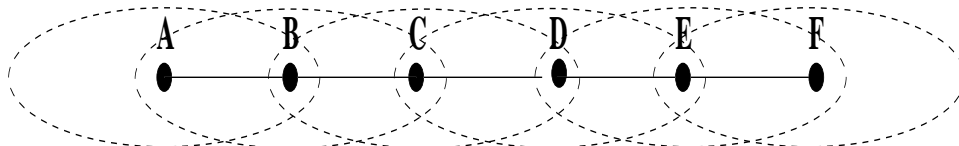
We now present the general analytical framework for modeling MAC layer fairness in shared wireless channels. Modeling fairness in this framework is a four-step process:

1. Step 1: From the network topology, we generate an undirected graph that captures the neighborhood property of nodes (i.e., nodes that are within range of each other are neighbors).
2. Step 2: From the graph and the set of active flows (wherein an active flow is a transmitter-receiver pair that has packets to send), we generate a *flow contention graph* that captures the contention among flows (i.e. each flow is a node in this graph, and two flows that contend for the same channel region are neighbors).
3. Step 3: From the flow contention graph, we generate a *resource constraint graph* that represents each “distinct contention region” as a resource server and each flow as a client. While the precise generation of the graph will become clear later, it is sufficient to know at this point that at most one flow can transmit at any time in a distinct contention region, and that a flow can transmit only if it is the sole transmitter in all the distinct contention regions to which it belongs.
4. Step 4: Given a resource constraint graph, we will show that achieving fairness in the system is equivalent to solving a utility maximization problem subject to the transmission constraints described above in the resource constraint graph. The fairness model of the system is determined by the utility function that must be maximized.

The high level overview of our approach is that Step 3 captures the location-dependent contention constraints of the shared wireless channel, and Step 4 optimizes the fairness model-specific utility function subject to these constraints. It will turn out that the formulation of the optimization problem will naturally lead to the ideal fairness objective, as well as translate automatically to a backoff-based contention resolution algorithm. With this overview, we will now describe the steps leading from modeling of fairness to the design of the backoff algorithm.

2.3.1 Steps 1, 2, and 3

In Step 1, the network is represented as an undirected graph $G = (V, E)$. V is the set of nodes in the network. In G , $e = (u, v)$ is an edge in E iff nodes u and v are within range of each other, where we assume that links are bidirectional, as in collision avoidance based CSMA/CA protocols. Figure 2.7 shows an example, wherein nodes $A - F$ are hosts, and the range of each flow is denoted by the dotted ellipses around the host.



$$G = (V, E)$$

Figure 2.7 Network graph G : No two neighbors can simultaneously transmit.

In Step 2, we consider all the “active flows” in the network, i.e., transmitter-receiver pairs that have packets to send. Let us assume in Figure 2.7 that all links are active flows. Note that when BC is active, AB , CD , and DE are constrained not to transmit or receive simultaneously, because at least one of the two end-points for each of these flows is a neighbor of either B or C . Thus any two active flows that are within a distance 2 in G contend with each other. More generally, if the *carrier sense threshold/packet*

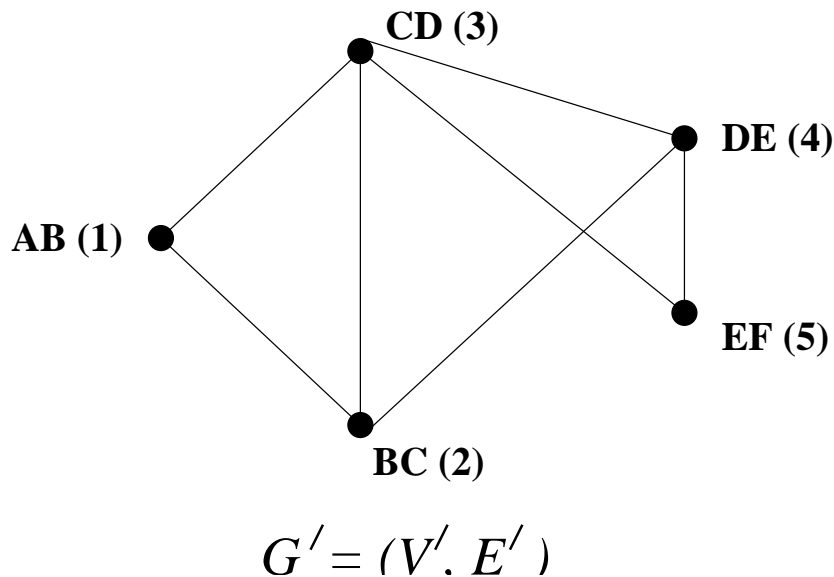


Figure 2.8 Flow-contention graph G' : In every clique, only one node can transmit.

reception threshold = δ , then any two flows that are within a distance of $1 + \delta$ potentially contend with each other.

We generate the flow-contention graph $G' = (V', E')$. $V' \subset E$; i.e., each node in G' is a link with packets to transmit in G . In G' , $e' = (u', v')$ is an edge in E' iff $d_G(u', v') \leq 1 + \delta$, where $d_G(e, e')$ is the shortest distance between the two links e and e' in graph G . Figure 2.8 represents the flow contention graph for the graph in Figure 2.7.

Let us now consider the maximal cliques [26] in G' . A *maximal clique* C in G' is a maximal complete subgraph of G' (e.g., in Figure 2.8, nodes AB, BC , and CD form a maximal clique). A maximal clique in the flow-contention graph represents a “distinct contention region” because at most one node in the clique can transmit at any time, and adding any other node to the maximal clique will enable two non-colliding simultaneous transmissions among the nodes under consideration.

In Step 3, we generate the resource contention graph G'' . $G'' = (V_1, V_2, E'')$ is a bipartite graph such that $V_1 = V'$, and each node in V_2 represents a maximal clique in G' . In G'' , $e'' = (u'', v'')$ is an edge in E'' iff $u'' \in V_1$, $v'' \in V_2$, and u'' belongs to the maximal clique in G' represented by v'' . Figure 2.9 represents the resource contention graph for the flow-contention graph shown in Figure 2.8.

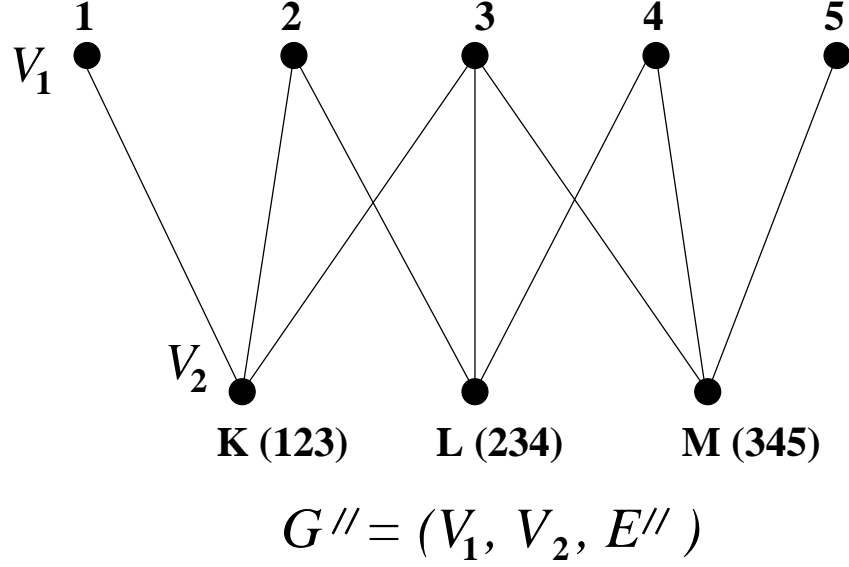


Figure 2.9 Resource contention graph G'' : Each maximal clique is a server and each flow is a client.

Each maximal clique in G' represents a “channel resource” with the nodes in the clique contending for exclusive access to the resource. Let us consider a simple slotted model of channel allocation. We represent each node in V_2 as a server which grants a *token* in each slot to at most one of the edges that is incident on it. Then, the allocation of a token in a clique represents the successful transmission by a node within the clique, and a node in V_1 (i.e., a flow in G) accesses a channel slot successfully if and only if it simultaneously obtains a token from all the edges incident on it.

Let us now consider a time window $[0, T]$. Let $I_{i,j}(t)$ be an indicator function such that $I_{i,j}(t) = 1$ if the node $j \in V_2$ allocates its token in slot t to node $i \in V_1$, and $I_{i,j}(t) = 0$ otherwise. Let $x_i(t)$ be the channel allocation for flow i in time $[0, t]$. Then the channel allocation problem can be represented as a set of the following linear constraints.

$$\begin{aligned} \forall j, \sum_i I_{i,j}(t) &\leq 1, \quad \forall t \\ \forall t, \forall i, x_i(t) &= x_i(t-1) + 1, \quad \text{if } I_{i,j}(t) = 1, \forall (i, j) \in E'' \\ &= x_i(t-1), \quad \text{otherwise.} \end{aligned}$$

Note that this set of constraints captures the location- dependent contention characteristics of shared wireless channels.

2.3.2 Step 4: Modeling fairness

In the rest of this section, we will only deal with unweighted fairness for simplicity of explanation. We will see at the end of the section that we can easily extend our discussions to achieve weighted fairness.

Consider a *utility function* $U(r)$ for a channel allocation rate r that is continuous, differentiable, increasing, and strictly concave over the range $r \geq 0$. If $U(r)$ is concave, then flows with a lower channel allocation rates will have a higher marginal utility than flows with higher channel allocation rates. If the flow contention graph is a complete graph, $\sum_i U(r_i)$ is maximized when $\forall i, j, r_i = r_j$; i.e., aggregate utility is maximized when the channel allocation rate for every flow is the same. For this simple case, it is easy to see that a maximizing the sum of concave utility functions achieves fairness.

It has been formally shown in [27] that there is a general equivalence between maximizing concave utility functions and achieving some systemwide notion of fairness. Specifically, each concave utility function achieves a corresponding fairness model. Thus, any fairness objective in shared wireless channels can be modeled by the following system of equations:

$$\text{Maximize } \sum_i U(r_i) \tag{2.1}$$

where

$$r_i = \Delta x_i(t) / \Delta t$$

subject to

$$\begin{aligned} \forall j, \sum_i I_{i,j}(t) &\leq 1, \quad \forall t \\ \forall t, \forall i, x_i(t) &= x_i(t-1) + 1, \quad \text{if } I_{i,j}(t) = 1, \forall (i,j) \in E'' \\ &= x_i(t-1), \quad \text{otherwise.} \end{aligned}$$

The constraints in the above system are always satisfied for a central channel allocation scheme. However, our goal is to achieve fully decentralized channel allocation. In this case, each flow controls its own rate allocation, which it adjusts in response to success or failure feedback. Now consider that flow i has a locally adjusted channel allocation rate of r_i . In every distinct contention region $j \in V2$, the “loss probability”

$$P_j \leq \left(\frac{\sum_{\{i|(i,j) \in E''\}} r_i - 1}{\sum_{\{i|(i,j) \in E''\}} r_i} \right)^+. \quad (2.2)$$

In other words, if the sum of the channel allocation rates in a maximal clique exceeds 1 slot, then some of the contenders will experience contention loss. Further, the contention loss probability that is experienced by any flow i is $p_i = 1 - \prod_{\{j|(i,j) \in E''\}} (1 - P_j)$. In other words, the probability of a flow successfully accessing a channel slot is the product of its success probabilities over all the distinct contention regions to which it belongs. If the loss probability in each distinct contention region is small, i.e., $P_j \rightarrow 0$, then the flow loss probability becomes $p_i = \sum_{\{j|(i,j) \in E''\}} P_j$. Note that we do not need to explicitly compute the loss probability in each distinct contention region. Each flow simply monitors its own loss probability that is a function of the sum of the loss probabilities over all the contention regions that it belongs to, and the flow’s channel allocation rate as a function of its own success/failure feedback, as we will see later.

For a flow with a channel allocation rate r_i , utility function $U(r_i)$, and perceived contention loss probability p_i , we represent its objective function as

$$\text{maximize } J(r_i) = \alpha U(r_i) - \beta p_i r_i, \quad (2.3)$$

where α and β are system parameters that, respectively, represent the “utility constant” and the “penalty constant” and can be tuned to achieve the desired trade-off between maximizing utility and minimizing loss rate. It has been shown in [28] that for a system of equations of this form, the aggregate utility $\sum_{i \in V1} J(r_i)$ is maximized when each individual flow i maximizes its own objective function $J(r_i)$. Further, as the penalty constant β becomes large, the optimal value of $\sum_{i \in V1} J(r_i)$ converges to $\alpha \sum_{i \in V1} U(r_i)$; i.e., the fully decentralized solution also converges to a channel allocation scheme that maximizes the aggregate utility over all the flows.

Having shown that the distributed scheme converges to the optimal aggregate utility, what remains is to derive the general framework for the adaptation of the channel allocation rate in each flow. Note that the flow objective function $J_i(r_i)$ is maximized when

$$\begin{aligned} \frac{dJ(r_i)}{dr_i} = 0 &\iff \alpha U'(r_i^*) - \beta p_i = 0 \\ &\iff \alpha - \frac{\beta p_i}{U'(r_i^*)} = 0 \end{aligned} \quad (2.4)$$

where the optimal channel allocation rate is denoted by r_i^* . Since $U(x)$ is concave and the penalty function is linear, $J(r_i)$ is a unimodal function. This suggests that the following scheme can be used for oscillating the channel allocation rate about the optimal point:

$$\dot{r}_i = \alpha - \frac{\beta p_i}{U'(r_i^*)} \quad (2.5)$$

where \dot{r}_i is the rate of change of channel allocation rate. It is easy to see that this scheme has an equilibrium point that is equal to the optimal value r_i^* that satisfies $\frac{dJ(r_i)}{dr_i} = 0$. It can be proved that the unique maximizing point is also a stable point of the system as follows. Note that

$$\begin{aligned} \frac{dJ(r_i)}{dt} &= \sum_{i \in V_1} \frac{dJ(r_i)}{dr_i} \frac{dr_i}{dt} \\ &= \sum_{i \in V_1} U'(r_i) \left(\alpha - \frac{\beta p_i}{U'(r_i)} \right)^2 \\ &\geq 0 \end{aligned} \quad (2.6)$$

since the derivative of the strictly concave, increasing function $U(r_i)$ is always positive. Thus, $J(r_i)$ is also a Lyapunov function for the channel allocation scheme given in Equation (2.5). Therefore the unique maximizing value of $J(r_i)$ to which the flow converges is a stable point of the system. This results in the following proposition:

Proposition 2.1 *Let all the flows in the system have the utility function $U(r_i)$. Then, if the channel allocation rate r_i for each flow changes according to the following algorithm*

$$\dot{r}_i = \alpha - \frac{\beta p_i}{U'(r_i)}, \quad (2.7)$$

the network utility and the flow utility functions converge to their optimal value, and the system converges to the optimal point. Further, for weighted fairness, we can replace α by $w_i \cdot \alpha$ which results in an optimal rate allocation that is a function of both the weight and the utility function.

The choice of the utility function for flows determines the fairness model for the system. The general form of flow utility functions is $U(r_i) = -1/r_i^\nu$ for $\nu > 0$. Among these, three fairness models have been of particular interest to the research community. When $\nu = 1$, the utility function $U_i(r_i) = -1/r_i$ characterizes the minimum potential delay fairness model [29]. For the special case when $\nu = 0$, the utility function is given by $U_i(r_i) = \log r_i$, and this is associated with proportional fairness [12]. The third is the max-min fairness model, and it has a very different utility function. For a normalized allocation x_i , the utility function is characterized by [29] $U_i(x_i) = -(-\log x_i)^\alpha$ $0 < x_i < 1$, $\alpha \rightarrow \infty$.

It should be clear at this point that from Proposition 2.1 we can derive the contention resolution algorithm as a function of the utility function. In the next section, we will present the realization of a simple, robust, and local contention resolution algorithm that achieves proportional fairness, as a representative example of the use of our fairness framework for translating an arbitrary fairness model to a corresponding contention resolution mechanism.

2.4 Proportionally Fair Contention Resolution (PFCR)

The general analytical framework for fairness presents a very powerful result – for any arbitrary fairness model that is represented by a concave differentiable utility function, we derive the corresponding channel allocation rate adaptation algorithm from Proposition 2.1. In this section, our goal is to start from this point and end up with a fully decentralized, purely local contention resolution algorithm that requires no explicit coordination among flows. Specifically, we seek to derive the contention resolution algorithm for achieving *proportional fairness*, as an example.

2.4.1 Proportional fairness and rate control

A vector of channel allocation $r = (r_i, i \in V_1)$ is *proportionally fair* if it is feasible (i.e. $r \geq 0$ and $\sum_{i \in j} r_i < 1 + \epsilon$ for every $j \in V_2$)) and if for any other feasible vector r' , the aggregate of the proportional changes is not positive [12]:

$$\sum_{i \in V_1} \frac{r'_i - r_i}{r_i} \leq 0.$$

Specifically, for the proportionally fair rate allocation vector r ,

$$\sum_{i \in V_1} \frac{dr_i}{r_i} = 0$$

which translates to the maximization of the utility function $U(r) = \log(r)$. *Thus proportional fairness is represented by the utility function $\log(r)$.*

Now, substituting for $U'(r_i) = \log'(r_i) = 1/r_i$ in Proposition 2.1 from the previous section, the channel allocation rate adaptation algorithm is given by

$$\dot{r}_i = \alpha - \beta p_i r_i. \tag{2.8}$$

Combining the results in the previous section and this section, we have now shown how to start with a fairness model (e.g., proportional fairness), convert it to a corresponding utility function, then use the framework to generate a channel allocation rate adaptation function. What remains is to use this rate adaptation algorithm to derive a contention resolution mechanism.

2.4.2 A local mechanism for proportional fairness

Recall that our goal is to design contention resolution mechanisms within the commonly followed guidelines of multiple access protocols. We thus have two instruments available to us: (a) persistence, and (b) backoff. Using these two instruments, we need to achieve a channel allocation rate adaptation algorithm of $\dot{r}_i = \alpha - \beta p_i r_i$.

There are two important observations that we make about this adaptation equation. First, the contention loss probability for each distinct contention region is very small

because flows will adapt their channel allocation rate when they observe loss – the penalty constant β can be tuned to make the throttling-upon-loss aggressive. Second, considering a single clique in isolation, every contending flow must observe the same loss probability for the derivation in the previous section to hold. Based on these observations, we obtain two results:

- The first observation implies that the sum of the channel allocation rates in any maximal clique will be close to 1. In other words, *we can approximate the channel allocation rate by a persistence probability, using which each flow decides whether or not to contend for a slot.*
- The second observation implies that *all flows must have the same backoff bound for fair contention loss distribution in a clique.*

In concert, these two results naturally define a contention resolution algorithm wherein a flow contends for a channel with a persistence probability that adapts according to the equation in Proposition 2.1, and a contending flow defers for a random time bounded by a system-wide backoff counter before commencing transmission. Of course, this contention resolution algorithm coexists with standard collision avoidance algorithms which preclude contention if the carrier is busy or if some other flow in the contention region has already acquired the channel.

An interesting feature of this algorithm is that it moves the burden of contention adaptation away from the backoff mechanism and into the persistence mechanism. This feature enables weighted versions of the contention resolution mechanisms to better approximate the ideal model, because weighted persistence is easier to achieve than weighted backoff. Of course, the fact that persistence is highly adaptive to loss implies that in any contention region, there are very few flows (expected value of 1) contending for the channel at any time, and hence a fixed backoff algorithm is both efficient and robust.

```

    contend_and_acquire_slot()
1  for each slot
2    State ← NO_CONTEND;
3    if uniform( 0, 1 ) ≤ xi
4      State ← CONTEND;
5      Bi = uniform( 0, B );
6      wait ( Bi );
7      if carrier_sense( ) == FREE
8        acquire_channel( );
9        if acquire_status( ) == COLLISION
10         xi ← xi ( 1 - β );
11        else State ← ACQUIRE;
12    else
13      xi ← xi ( 1 - β );
14  xi ← xi + α;

```

Figure 2.10 Pseudo code for the PFCR algorithm.

2.4.2.1 Protocol details

There are three states in which a flow can exist:

(a) *NO_CONTEND*, (b) *CONTEND* and (c) *ACQUIRE*. At the beginning of each slot, all flows are in the *NO_CONTEND* state. There are two steps involved acquiring a channel for a particular slot: (a) a flow decides to contend for a slot (*NO_CONTEND* → *CONTEND*), and (b) if it decides to contend, it tries to acquire the channel (*CONTEND* → *ACQUIRE*). The pseudo code and the state transition diagram for the PFCR algorithm are illustrated in Figures 2.10 and 2.11.

Each flow has a transmission probability x_i . When a flow has a packet to transmit and does not sense a carrier, it moves from the *NO_CONTEND* to the *CONTEND* state with a probability of x_i (Lines 2-4). Each contending host chooses a wait time of B_i , where B_i is uniformly distributed in the interval $[0, B]$. B is a system-wide backoff counter. After the waiting time, the flow senses the carrier. If the channel is free, then the flow tries to acquire the channel (Lines 6-8). If either the channel is busy or if there is collision, the

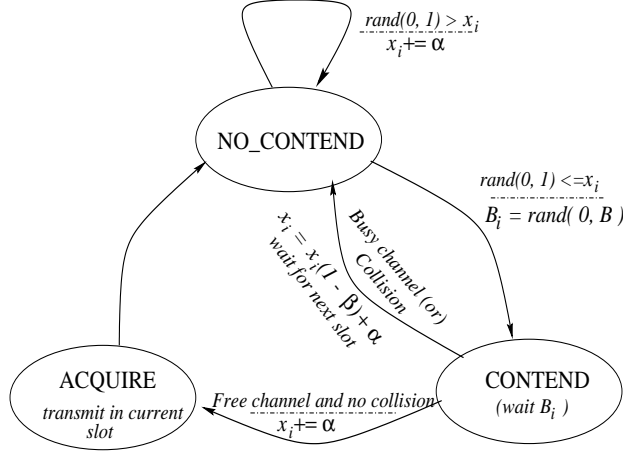


Figure 2.11 State transition diagram for the PFCR algorithm.

flow declares the slot as lost (Lines 9, 10, 12) and reduces its transmission probability by βx_i (Lines 10, 13). Note that the notion of contention loss subsumes collisions in that a contending flow also declares a loss if it did not win the contention round (e.g., if it perceives a busy carrier at the end of the waiting period). This is consistent with the derivation in the analytical framework in Section 4. If the flow successfully acquires the channel, it moves from the *CONTENT* state to the *ACQUIRE* state (Line 11). At the end of the transmission slot, all flows that have packets to send increase their transmission probability by α (Line 14). Since a flow experiences a contention loss with probability p_i , the expected change in the channel allocation rate is $\alpha - \beta p_i x_i$. This exactly reflects the adaptation algorithm in Proposition 2.1.

The algorithm described above is executed independently for each flow and thus is fully decentralized. Flows do not need to have any knowledge of the topology of the network, and adaptation to the dynamics of the flows, channel conditions and topology occurs implicitly (via increased or decreased contention loss feedback). Most importantly, the contention resolution algorithm is derived naturally from the framework, and is very general. For any given concave and differentiable utility function, the contention resolution algorithm that achieves the corresponding fairness property is automatically generated.

2.4.2.2 Practical considerations

While the analytical framework guarantees stability and fairness in a probabilistic sense, a number of practical issues need to be addressed in order to make the contention resolution algorithm robust, efficient, and closely approximate the expected fairness model. The most important of these is the selection of the α and β parameters.

In order to approximate the ideal fairness model perfectly, $\beta \rightarrow 1$ because a large penalty constant ensures that the sum of the transmission probabilities in a distinct contention region never exceeds 1. This has two effects. First, setting a large β value causes large oscillations in short-term fairness due to large fluctuations in the persistence. Second, since the transmission probabilities must never exceed 1, α must be small. However, this reduces efficiency under low loads. Fundamentally, there is a trade-off between achieving higher efficiency ($\alpha \uparrow$, $\beta \downarrow$) and achieving better approximations to the ideal fairness objective ($\alpha \downarrow$, $\beta \uparrow$). In our simulations, we set $\alpha = 0.1$, $\beta = 0.5$, and $B = 32$.

2.5 Performance of the PFCR Algorithm

In this section, we present simulation results to compare the fairness characteristics of the PFCR algorithm, with some existing medium access protocols that were discussed earlier in Section 2.1. In particular, we have chosen three protocols with different contention resolution schemes: IEEE 802.11 [11], MACAW [9] and CB-Fair [13], for comparison. IEEE 802.11 uses binary exponential backoff with backoff reset for each new packet. MACAW uses contention measurement at the sender and the receiver with multiplicative increase and linear decrease, as well as backoff copy. CB-Fair [13] uses multiplicative increase and multiplicative decrease for backoff with copying, and adjusts its persistence based on the characteristics of the second neighborhood of the sender. To show the contention among various flows better, we have chosen to illustrate the topology using the flow-contention graph G' and the resource contention graph G'' for each of the simulation scenarios. For each flow, we present the throughput obtained under different protocols and evaluate the throughput against the ideal proportionally fair rate allocation

for that topology. At the outset, we recognize that proportional fairness is the appropriate benchmark objective only for PFCR. However, none of the other approaches has a well-defined fairness objective, and they do not match any of the well-known fairness objective functions such as max-min, proportional, or min-potential delay fairness.

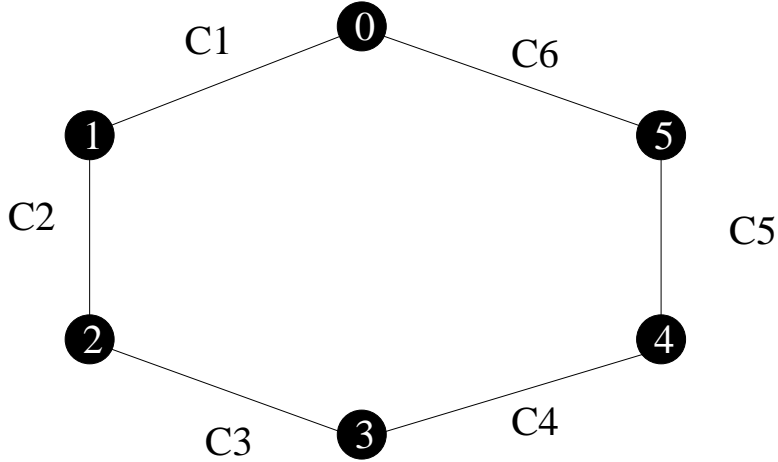


Figure 2.12 Flow contention graph G' for Example 1.

2.5.1 Example 1

We first look at a very simple ring topology, as in Figure 2.12. All flows have the same number of contending flows. We present the throughput results for each protocol in Table 2.1 along with the ideal throughput expected under a proportional fair system. We also show the relative performance of the various protocols with respect to the proportionally fair throughput in Figure 2.13. For this scenario, binary exponential backoff performs worse than the other protocols, although all protocols obtain within 98% of the desired throughput.

Table 2.1 Example 1: Total number of packets transmitted

FlowID	802.11	MACAW	CB-Fair	PFCR	IDEAL
1	22 253	22 550	22 550	22 501	22 460
4	23 009	22 491	22 334	22 525	22 460

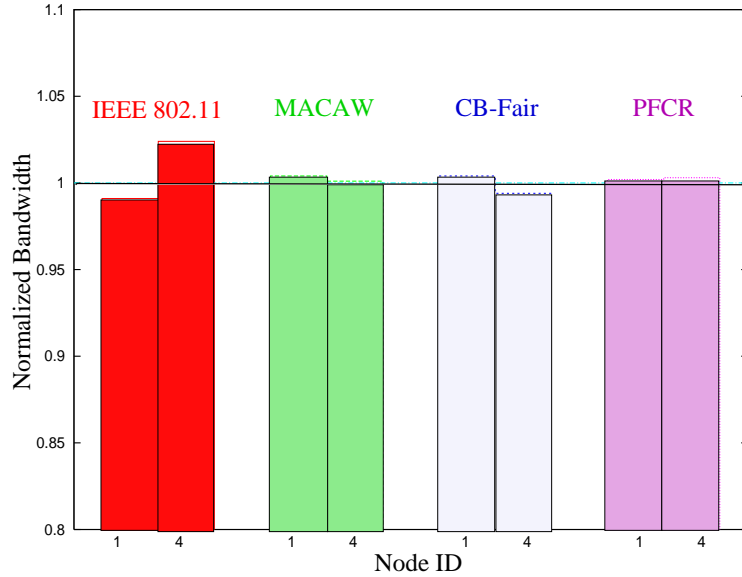


Figure 2.13 Example 1 - Relative throughput of various algorithms compared to proportional fair allocation.

Table 2.2 Example 2: Total number of packets transmitted

FlowID	802.11	MACAW	CB-Fair	PFCR	IDEAL
5	7 645	7 834	9 076	7 865	7 633
6	4 825	2 683	10 322	5 357	5 724
7	23 543	23 979	15 433	22 900	22 898

2.5.2 Example 2

In Example 2, we consider two flow cliques, shown in Figures 2.14 and 2.15. One clique has a small number of flows, and the second clique has a large number of flows. The throughput and relative performance are shown in Table 2.2 and Figure 2.16 respectively. The throughput results show us something very interesting. Flow 6 is part of two cliques, and as a result, it has to win in both the cliques to transmit. In such cases, binary exponential backoff and contention measurement mechanisms cause short-term unfairness as a result of asymmetric contention. Flow 6 receives a rate that is 16% less than the proportional fair share under IEEE 802.11; it receives 54% less than the ideal under MACAW. The bandwidth lost by Flow 6 is used by the other flows in both the cliques,

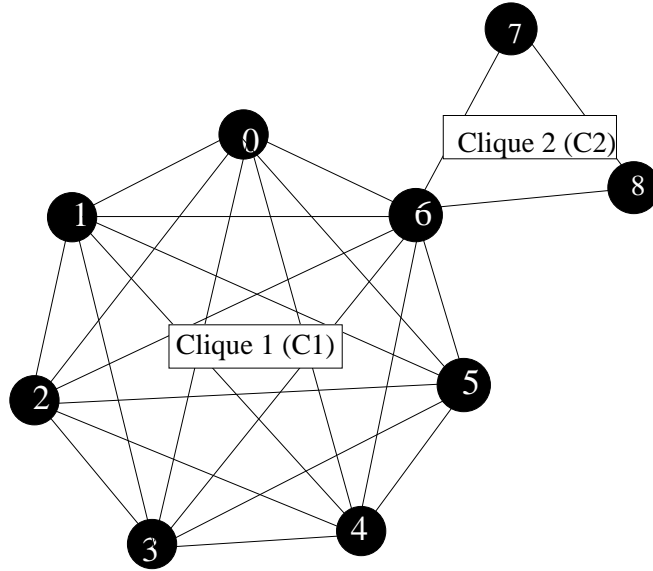


Figure 2.14 Flow contention graph G' for Example 2.

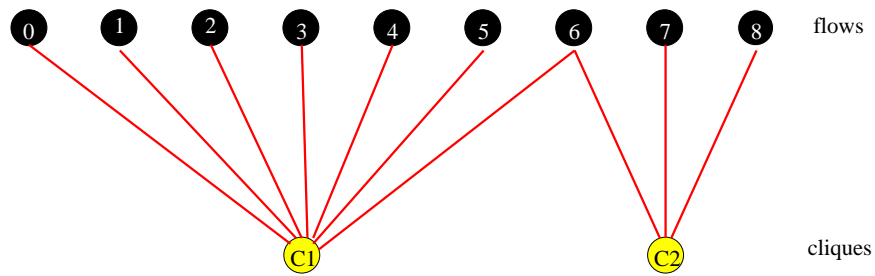


Figure 2.15 Contention clique graph G'' for Example 2.

which obtain above-the-ideal bandwidth. In the case of CB-Fair, however, Flow 6 has the highest degree among all flows, and hence, it receives more than ideal throughput, while other flows receive lesser throughput. However, the PFCR algorithm is able to match the proportionally fair allocation for every node, in particular, the rate for Flow 6 is within 6% of the ideal throughput.

2.5.3 Example 3

In this example, we illustrate the case when there are many flow-cliques in the network and one flow is part of all the cliques, as in Figure 2.17. Figure 2.18 shows that Flow 0 belongs to all the cliques while all other flows are just part of a single clique. The

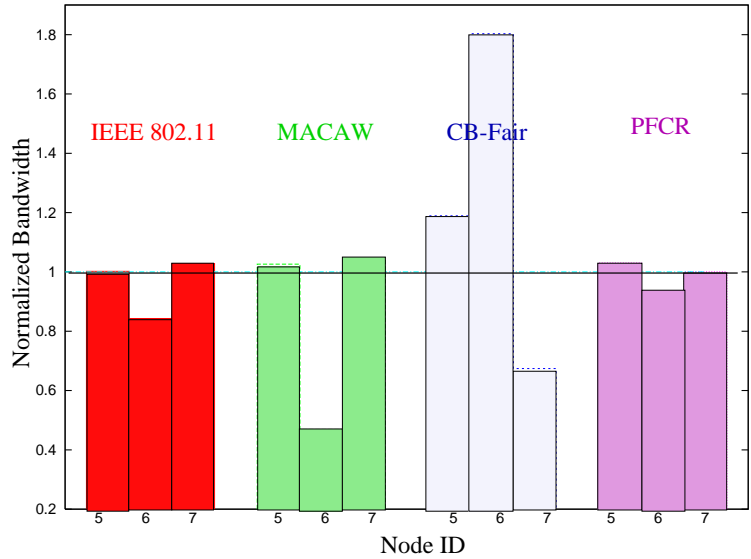


Figure 2.16 Example 2 - Relative throughput of various algorithms compared to proportional fair allocation.

Table 2.3 Example 3: Total number of packets transmitted

FlowID	802.11	MACAW	CB-Fair	PFCR	IDEAL
0	3 571	1 424	27 830	5 764	5 931
4	24 091	24 271	20 298	23 613	23 724

results are presented in Table 2.3 and Figure 2.19. As in the previous case, the PFCR algorithm does not exhibit any short-term unfairness and follows the ideal proportional fair allocation for all flows, especially for Flow 0, which belongs to all the cliques. However, 802.11 gives Flow 0 a rate that is 60% of the proportional fair share and MACAW gives a rate of 20% of the proportional fair share. CB-Fair, on the other hand, gives Flow 0 five times the proportionally fair throughput, since its degree is far higher than any of the other flows. Compared to this, the PFCR algorithm gives a rate that is just 3% less than the proportional fair share.

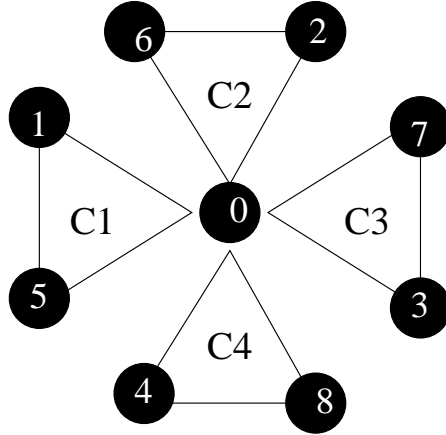


Figure 2.17 Flow contention graph G' for Example 3.

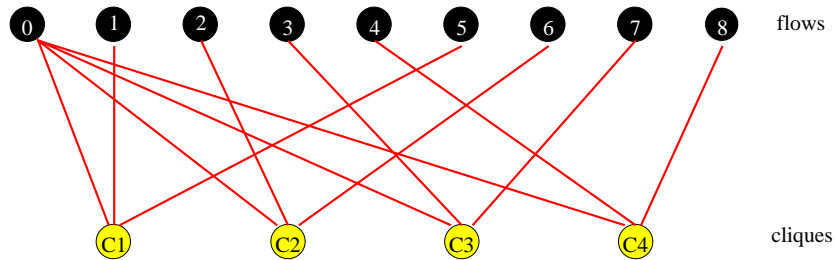


Figure 2.18 Contention clique graph G'' for Example 3.

2.5.4 Example 4

In this example, we will consider a scenario where all flows have identical degrees in the flow-contention graph G' . In this scenario, the CB-Fair algorithm will try to give nearly equal rates for every flow, whereas the proportional fair rate allocation depends on the topology of the graph. Consider the flow-contention graph shown in Figure 2.20. Flows 0, 4, 8, and 12 belong to two cliques, one of size 4 and the other of size 2. Flow 16 is part of four cliques with two flows in each. All the other flows are part of a clique of size

Table 2.4 Example 4: Total number of packets transmitted

FlowID	802.11	MACAW	CB-Fair	PFCR	IDEAL
0	6 208	8 406	11 784	10 686	11 973
1	14 454	14 171	13 934	13 934	12 970
16	42 449	37 351	28 763	28 763	38 910

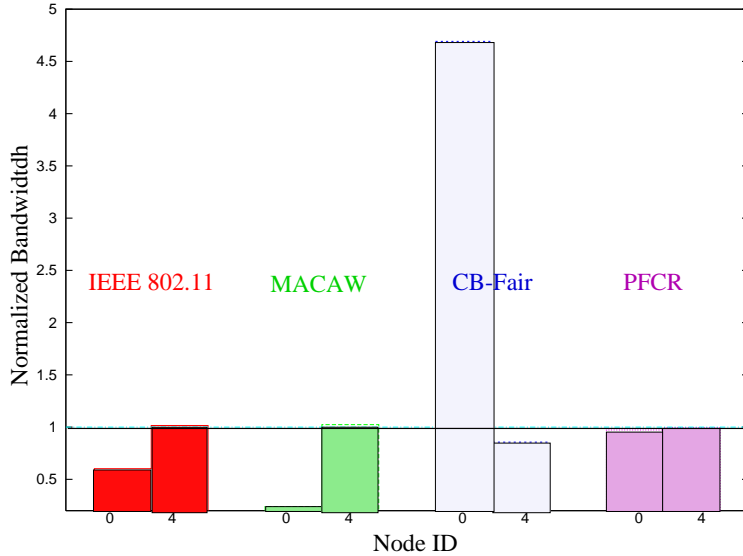


Figure 2.19 Example 3 - Relative throughput of various algorithms compared to proportional fair allocation.

4. When we look at the throughput behavior of the various protocols, shown in Table 2.4 and Figure 2.21 for all the protocols, IEEE 802.11 and MACAW allocate bandwidth in a very unfair manner. Flows that experience asymmetric contention suffer, while flows that experience uniform levels of contention achieve better throughput. In IEEE 802.11, Flows 0, 4, 8, and 12 receive only 1/2 of the ideal proportional fair share, while Flow 16 receives 9% more than its fair share. In MACAW, Flow 16 gets 4% less than its fair share, while Flows 0, 4, 8, and 12 receive just 70% of their fair share. CB-Fair gives a better bandwidth allocation to Flows 0, 4, 8, and 12 while penalizing Flow 16 which gets only 50% of the ideal bandwidth, even though they all have the same degree. This result occurs because flows 0, 4, 8 and 12 are in a region of high contention, and in CB-Fair, higher contention flows win over flows experiencing lower contention. Note that the proportional fairness model does not penalize asymmetric contention flows severely, and the PFCR algorithm, which is modeled to achieve proportional fairness, closely follows the ideal allocation of rates for all flows. Using PFCR, Flows 0, 4, 8, and 12 receive 91% of their fair share, and Flow 16 gets just 1% more than its fair share.

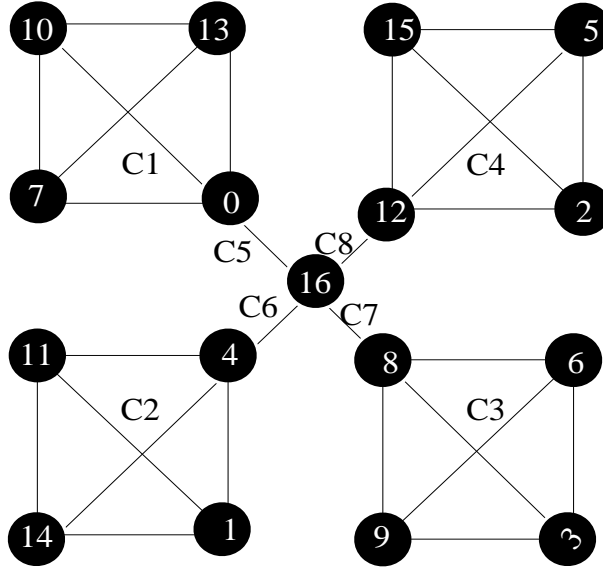


Figure 2.20 Flow contention graph G' for Example 4.

Of course, as we pointed out in the previous section, the closeness to which PFCR approximates proportional fairness is dependent on the α and β values chosen for the simulations. While it is of course possible to tune these parameters according to each simulation configuration to achieve best performance, we chose to set them a priori and make them nonadaptive because, in practice, nodes do not have nonlocal state, and there is no reliable way to achieve coordination of parameter adaptation among the nodes. We found that the results typically did not change significantly for the range of values $\alpha \in [0.01, 0.15]$ and $\beta \in [0.3, 0.7]$. We thus used $\alpha = 0.1$ and $\beta = 0.5$ consistently for all our simulations. In all our experiments, it turns out that PFCR approximates the ideal proportional fairness objective quite well.

2.6 Summary and Future Work

State-of-the-art multiple access protocols for shared channel wireless networks often lack a well-defined notion of fairness and provide somewhat ad hoc mechanisms for achieving fairness among contending flows. However, as shared channel wireless networks move from the academic domain to the commercial domain, these environments must support

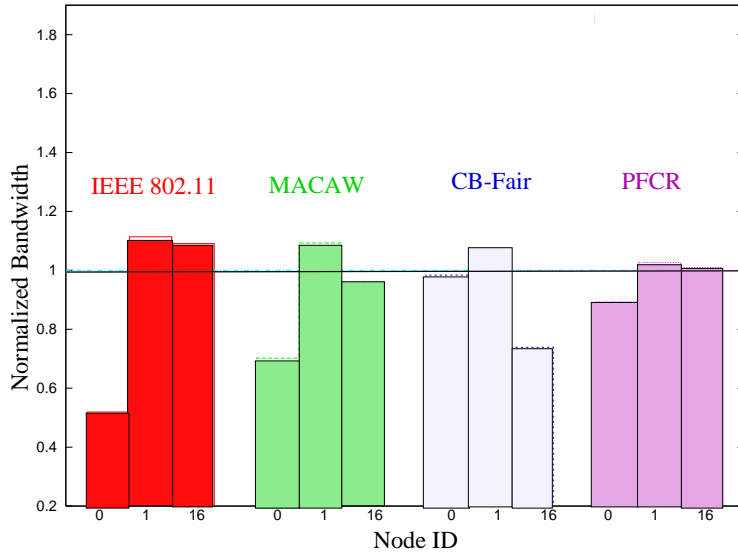


Figure 2.21 Example 4 - Relative throughput of various algorithms compared to proportional fair allocation.

network-level quality of service and service differentiation, which translates to MAC layer weighted fairness. While there is a wealth of work on achieving fairness in wireline and cellular networks, these approaches are inappropriate for shared wireless channels because of the unique location-dependent nature of contention in such networks. Given the spatial distribution of contention, different systems may choose to enforce different fairness models depending on their requirements, goals, and deployment scenarios.

Based on these requirements, we perceived the need to develop a general analytical framework in which to reason about fairness in shared channel wireless networks. This chapter makes two fundamental contributions: (a) we propose perhaps the first general framework in which a large class of fairness objectives (represented by concave differentiable utility functions) can be modeled, and (b) we propose the first translation mechanism for taking a fairness specification and automatically generating a corresponding contention resolution algorithm. Together, these two techniques are very powerful because they allow future wireless network designers to deploy different service differentiation models as well as alter fairness/service models on the fly.

We demonstrated the techniques in this thesis by starting with the proportional fairness model, generating the corresponding utility function, and then plugging the derivative of this function into the generic channel allocation rate adaptation algorithm to obtain a fully decentralized, purely local contention resolution algorithm. We showed via simulation experiments that the automatically generated contention resolution algorithm does in fact approximate the ideal fairness objective closely.

CHAPTER 3

PARALLEL SCHEDULING PROBLEMS IN NEXT GENERATION WIRELESS NETWORKS

In this chapter, we consider a class of scheduling problems in next generation multi-rate cellular networks, and propose our solutions for the same. We present the communication channel model, and abstract our scheduling problem in Section 3.1. In Section 3.2, we present a theoretical study showing the structure and the complexity of these problems. In Section 3.3, we present our main algorithmic results. In particular, in Sections 3.3.1 and 3.3.2 we consider simple online algorithms for minimizing the maximum response time and present our augmented resource based analyses. In Section 3.3.3 we present preliminary algorithmic results for optimizing other per-request QoS metrics such as average response times. In Section 3.4, we present our experimental results. We conclude the chapter in Section 3.5.

3.1 Problem Formulation

3.1.1 Communication channel model

In wireless systems, channels have variable attenuation, depending on the geographic location of the users. This is mainly due to multipath impairments and radio propagation losses. Say the base station (BS) is communicating with n mobile users. The physical channel attenuations of the users are denoted by $\bar{g}_1, \bar{g}_2, \dots, \bar{g}_n$, respectively; each \bar{g}_i is a scalar factor which we call the *physical gain*. If the BS transmits power p_i to a user i , the signal-to-interference-plus-noise ratio (SINR) is given by $SINR = \frac{\bar{g}_i p_i}{\sigma^2}$, where σ^2 is the total noise power (including interference) [30]. SINR determines the rate of transmission

of packets to the user¹. In particular, the rate $r_{bps}(\cdot)$ as a function of the SINR is a concave logarithmic function [31]

$$r_{bps}(x) = \bar{W} \log_2\left(1 + \frac{x}{\Gamma}\right), \quad (3.1)$$

where $r_{bps}(\cdot)$ is the rate in bits per second, \bar{W} is the spectral bandwidth used, and Γ is dependent on the coding gain from the physical layer error-correcting code [31]. Both Γ and \bar{W} are system parameters, which for our purposes will be constants. Therefore, for a particular user, the service received over a period of time τ , obtained as a function of the power p_i allocated to the user on a single channel (code), is given by

$$r_i(p_i) = \bar{W}\tau \log_2\left(1 + \frac{\bar{g}_i p_i}{\sigma^2 \Gamma}\right). \quad (3.2)$$

The rate versus SINR curves for next-generation wireless systems closely approximate the convex function described by this equation (see for example [15]). This rate function already embodies the effect of variable rate error-correcting coding schemes in the physical layer, as is typical in next generation wireless systems [14, 15, 31]. For example, Figure 3.1 shows the actual rates used in the Qualcomm HDR system against the analytical model. Therefore, we will use this equation for rate calculations in our scheduling problems. For notational convenience we will denote $g_i = \frac{\bar{g}_i}{\Gamma \sigma^2}$ as the *channel gain*, and we will set $W = \tau \bar{W}$ yielding $r_i(p_i) = W \log_2(1 + p_i g_i)$.

3.1.2 Abstract scheduling problem

In this section, we abstract a parallel scheduling problem that arises in multirate, multicode next generation wireless data systems such as the ones above. Our focus is on downlink and non-real-time traffic (such as browsing, downloads, etc.). In the following discussion, we will use the terms *requests*, *jobs* and *users* interchangeably.

¹The SINR is an important parameter for two reasons. First, it determines the probability of error in transmission of packets. Second, for a given error probability, we can transmit at higher rates dependent on the SINR. For example, when we have a higher SINR, we can transmit at a higher information rate for the same error probability [30].

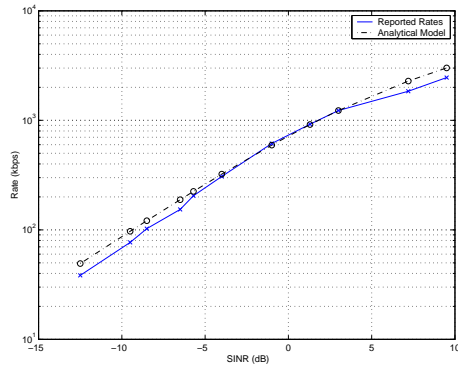


Figure 3.1 Qualcomm HDR rate structure.

The base station has a total power P to transmit. Time is assumed to be partitioned into equal width windows called *time slots*² (or frames), whose width is τ . We also assume that there are a total of C codes which can be assigned to users in a time slot. If user u makes a request i , then we say that the gain of request i is the same as the gain of the user u , $g_i = g_u$. Requests arrive in the system over time at the beginning of time slots. The size s_i (in bits) and the channel gain g_i of the user who made the i th request are known when the request arrives at time a_i . The arrival time is also known as *release time*. We will assume that the channel conditions of the users are constant over the scheduling period. Although this is a simplification, it holds in significant cases; i.e., if the time scale of the scheduler is several seconds and if the users do not have very high mobility, then the channel conditions will be static over this time scale [30]. Additionally, this already proves challenging. We address the more general problem of time-varying user gains later in this chapter.

The scheduling problem is to determine an assignment of power and codes to each user in each time slot. Hence, the scheduling problem is:

Inputs: Online job arrivals which specifies job size and time of arrival.

Outputs: The set $\mathcal{C}_u(t)$, the set of codes assigned to user u at time t , and $p_u^{(i)}(t)$, the power assigned to user u at time t to each code $i \in \mathcal{C}_u(t)$.

²We will use *time* and *time slot* interchangeably when no confusion arises.

This assignment translates to effective rate per code as given by Equation (3.2). The assignment must satisfy the following conditions:

- *Discrete rate set:* Only a discrete set of rates (equivalently, minimum power per discrete rate) is allowed. These rates are denoted $R(1), R(2), \dots$ and have the property that $\frac{R(i)}{R(i-1)} \leq \theta$. When $\theta = 2$, this relationship holds for existing next generation wireless data system proposals like cdma2000, WCDMA, and HDR, which has a rate set of $\{38.4, 76.8, 102.6, 153.6, 204.8, 307.2, 614.4, 921.6, 1228.8, 1843.2, 2457.6\}$ kb/s. We also make a regularity assumption that the user gains are such that the lowest discrete rate $R(1) \leq W \log(1 + gP)$, i.e., by allocating all the resources to the user there exists a feasible discrete rate. In practice, error-correcting codes can be used over a group of codes to increase the dynamic range of user gains that fall into the feasible region.
- *Request completion:* All requests get the requested data size; that is, we need

$$s_u = \sum_t R_u(t) = \sum_t \sum_{i \in \mathcal{C}_{u(t)}} r_i(p_i) \quad (3.3)$$

where $\mathcal{C}_{u(t)}$ is the set of codes assigned to user u in time slot t , and $r_i(p_i)$ is calculated using (3.2) for the continuous or discrete cases as needed.

QoS metric Various QoS metrics could be optimized. We focus on one metric, namely response time and our criterion is to *minimize the maximum response time* or *max-flow* [32], where response time for request i is $c_i - a_i$, if request i is completed by time c_i . We remark that this metric is also sometimes called *flow time* in literature. Although our focus is on minimizing maximum response time, we have also studied other metrics, such as minimizing average response time and minimizing total weighted response time.

We assume requests may be served over several time slots with different sets of codes at each time slot. In standard scheduling terminology [32], this corresponds to requests being *preempted* (i.e., stop processing a request, process other requests, and resume the

original request) or being *migrated* (i.e., assign sets of codes to a user that differ from one time slot to another).³

Goal: *To come up with a schedule that minimizes the maximum response time, given an instance of jobs.*

There are two basic variants of our problems: namely *offline* or *online*.

- **Offline problem:** All request arrivals are known ahead of time. The offline case is of theoretical interest and is mainly useful to quantify the benefit to be accrued from scheduling.
- **Online problem:** Requests arrive over time and the scheduling algorithms have to take their decisions without knowledge of future requests. The performance of the online algorithms are measured in comparison to the offline case.

3.1.3 Malleable task scheduling

The above scheduling problem has some similarities to scheduling malleable tasks on parallel machines.

Given a set of n tasks and m identical machines we assume that every task j has a processing specified by m positive integers, $p_{j,1}, \dots, p_{j,m}$: $p_{j,q}$ ($1 \leq q \leq m$) denotes the processing time of task j when it is executed in parallel on q processors. The following assumptions are normally done in the literature [33, 34, 35]: (i) the number of machines assigned to any task j cannot change over time; (ii) $p_{j,q}$ is a nonincreasing function of the number q of processors executing the tasks; and (iii) the total work $w_{j,q} = q \cdot p_{j,q}$ done on malleable task j is a nondecreasing function of the number of processors executing j .

Note that the above formulation differs from the scheduling problem considered in this dissertation. In fact, in our case, increasing the number of codes results in increasing transmission efficiency, thereby decreasing total power consumed to schedule a request.

³A more detailed model may distinguish some codes to be more preferable than the others from one time slot to another to insure intercell interference avoidance, but we do not consider such issues in this dissertation.

This is due to Equation (3.2) which shows the concavity of the rate function with respect to power allocated.

Moreover, the main difference is that our wireless scheduling problem is in essence a two-dimensional packing problem. In fact, in our scheduling problem the processing time of a job depends on two variables (the number of codes and the assigned power per slot), whereas in the malleable case the processing time only depends on the number of processors assigned to the task. Also, we do allow the assignment to a request of a different set of codes in different slots.

Finally, we observe that (a) for malleable scheduling the problem of optimizing response-time-related metrics has not been addressed, and (b) attention has focused on minimizing (weighted) completion time.

3.2 Understanding the Scheduling Problems

3.2.1 Some structural observations

Here, we state and prove some properties of the communication channel. They will be invoked later in proving our main results.

Continuous power (rate) case First, we consider the case where the rates are not discrete and are a monotonic function of power, as in Equation (3.2). Concavity of the rate with respect to p in (3.2) implies that if we assign $c \geq 1$ codes to a user u , then *it is optimal to divide the total power p allocated to that user equally among the codes assigned* as summarized below.

Lemma 3.1 (*Equipartition of power*) *Given c codes and power p to a user, the rate r is maximized for $p_i = p/c$, $i = 1, \dots, c$.*

Using Lemma 3.1 we can write the rate obtained by a user given c codes and p total power as,

$$R(p, c) = Wc \log\left(1 + \frac{gp}{c}\right). \quad (3.4)$$

Discrete Power (Rate) Case When we have a discrete rate set, the actual rate obtained on each code is given by the highest discrete rate which is below $W \log(1 + \frac{p}{c})$. Therefore, the difference between the continuous rate and the discrete rate case, depends on the discrete rate set available. Therefore, we can easily see the following fact.

Fact 3.1 *If the discrete rate set $\{R(j)\}$ is such that $\frac{R(j)}{R(j-1)} \leq \theta$ then for any continuous power allocation p there exists a discrete rate $R(l)$ such that $R(l) \geq \frac{1}{\theta} r_i(p)$, where $r_i(p)$ is given by (3.2).*

3.2.2 Computational hardness

In order to understand the challenge of the problem further, let us consider the offline complexity of the scheduling problems. If power (rate) values are required to be drawn from a discrete set, the problem in its simplest instance is the bin packing problem and hence it is NP-complete [36]. We focus on the more challenging case when code is discrete, as usual, but the power (and hence rate) is allowed to take any continuous value. In order to prove the hardness of this problem, we will consider the version of the problem in which i th request has arrival (release) time a_i , deadline d_i and size s_i in bytes. Using this, we can state the following theorem.

Theorem 3.1 : *If the number of codes assigned to each user is integral, then it is NP-complete to compute a feasible schedule for the problem of meeting deadlines even if all users have the same channel gain, a common release time, a common deadline and the power assigned to each code is not restricted to a discrete set of values.*

Proof: We reduce the NP-complete problem *3-partition* defined as follows [36]:

Instance: A set A of $3m$ elements, a_1, a_2, \dots, a_{3m} , a bound B and a size $s(a_j)$, $a_j \in A$, such that $\sum_{j=1}^{3m} s(a_j) = mB$.

Question: Can A be partitioned into m disjoint sets A_1, A_2, \dots, A_m such that for $1 \leq i \leq m$, A_i has three elements and $\sum_{a_j \in A_i} s(a_j) = B$?

The reduction is as follows: given an instance \mathcal{I} of 3-partition, we define an instance \mathcal{J} of the combinatorial problem of meeting deadlines with $3m$ users. The request of user j ,

$1 \leq j \leq 3m$, has size $s_j = s(a_j)$; all requests are released at time 0 and have a common deadline $D = m$. All users have the same channel gain g , and therefore, the power that users need to get desired rates depends only on the number of codes they are assigned and on the size of the request. Let p_j denote the power assigned to user j if only one code is assigned to j over all frames. We assume that there are *three* codes per frame and that the maximum power of the base station is P where $\sum_{j=1}^{3m} p_j = mP$.

We now show that there is an assignment of users to frames that meets the common deadline D if and only if \mathcal{I} has a feasible solution.

Assume that A_1, A_2, \dots, A_m is a feasible solution of \mathcal{I} . We define a solution of \mathcal{J} as follows: if element a_j , $j = 1, 2, \dots, 3m$, is assigned to set A_i then user j is assigned to frame i with the minimum power that is needed to satisfy the user's request in one code in any one frame. It is easy to see that this is a feasible solution since $\sum_{a_j \in A_i} s(a_j) = B$ implies that P is the total power required by users assigned to frame i .

Similarly, it is possible to show that given a feasible solution of \mathcal{J} that satisfy all users' request within m frames, it is possible to obtain a feasible solution of \mathcal{I} . Namely, it is sufficient to assign to set A_i , $i = 1, 2, \dots, m$, the elements that correspond to users assigned to frame i ; since the total power assigned to frame i is P , it follows that $\sum_{a_j \in A_i} s(a_j) = B$. That completes the proof. ■

The result above in fact shows the problem to be NP-complete in the strong sense (see [36] for definition and significance). Although we have shown this hardness result only for the deadlines problem, it is easy to see that this immediately gives the hardness of the other scheduling problem we have, namely, minimizing the maximum response times. Finally, notice that the result holds independently of how rates are affected by the use of multiple codes, since the hardness is proved even for the restrictive case when each request gets only one code over all time slots.

We can also show that this problem is hard to approximate. This is summarized in the following theorem.

Theorem 3.2 *For every $c > 0$, it is NP-hard to compute a c -approximation of the maximum response time.*

Proof: We apply the gap technique for proving inapproximability results ([37]) showing a reduction from three-dimensional matching to the problem of minimizing the maximum response time in the discrete case.

An instance of three-dimensional matching is specified by a set of $3n$ rationals x_i , $0 < x_i < 1$, $i = 1, 2, \dots, 3n$ such that $\sum_{i=1}^{3n} x_i = n$; the problem requires the partitioning of the given numbers in n sets F_j , $j = 1, 2, \dots, n$, such that for each j , $j = 1, 2, \dots, n$, $|F_j| = 3$ and $\sum_{x_i \in F_j} x_i = 1$.

Given c and an instance $\mathcal{I} = \{x_1, x_2, \dots, x_{3n}\}$ of three-dimensional matching we define an instance \mathcal{I}' of the scheduling problem of minimizing the max response time and we show that if \mathcal{I} has a solution then the max response time of \mathcal{I}' is n ; otherwise the max response time of \mathcal{I}' is bigger than cn . Therefore, by the gap technique, if there exists a polynomial time algorithm for solving the problem of minimizing the maximum response time within a factor c approximation of the optimum, then it is possible to solve three-dimensional matching in polynomial time. Since three-dimensional matching is NP-complete, the theorem follows.

Given c and $\mathcal{I} = \{x_1, x_2, \dots, x_{3n}\}$ we define an instance of the scheduling problem with $(cn + 1)n(c + 3)$ jobs and we assume that each slot has total power equal to 10 and that there are three codes per slot.

The jobs are released in $cn + 1$ phases; in each phase $n(c + 3)$ jobs are released. Phase $k = 0, \dots, cn$ is formed by two stages as follows:

- **Stage 1:** At time $t = kn(c + 1)$, $3n$ requests J_1, J_2, \dots, J_{3n} arrive. For each k and i , the gain function of request J_i is such that by assigning power greater or equal to $3 + x_i$ one code is sufficient to complete the job in a slot and $3 + x_i$ is the minimum power requirement to process request J_i . Jobs released in Stage 1 of a phase are called J-jobs in the following.
- **Stage 2:** At time $t = kn(c + 1) + n + j$, $j = 0, 1, \dots, nc - 1$ a job A_j is released. For each k and j , the gain function of this job is such that by assigning power ≥ 8 , one code is sufficient to complete it in a slot and 8 is the minimum power requirement

to process the job. Jobs released in Stage 2 of a phase are called A-jobs in the following.

The easy proof of the following fact is omitted.

Fact 3.2 *All requests released in Stage 1 of a phase can be completed using n slots if and only if \mathcal{I} has a solution; otherwise there are at least $n + 1$ slots containing requests released in Stage 1 of the phase.*

We first prove that if instance \mathcal{I} of three-dimensional matching has a solution, then the maximum response time is equal to n . This follows since the $3n$ jobs released at time $kn(c + 1)$ can be scheduled in n slots and therefore completed by time $kn(c + 1) + n$. Every job of Stage 2, released at time $t = kn(c + 1) + n + j$, $j = 0, 1, \dots, nc - 1$, is scheduled at time t itself.

We are left to prove that if instance \mathcal{I} of the three-dimensional matching problem has no solution, then the maximum response time is bigger than cn . This follows from the following induction on the number of phases k :

Fact 3.3 *Assume that instance \mathcal{I} does not have a solution. For every $k = 1, 2, \dots, cn$, if no J-job released in a phase $j < k$ has response time bigger than ' cn ', then $k - 1$ A-jobs released in phases previous to k are not scheduled by time $kn(c + 1)$.*

Proof: We prove the claim by induction. For the basis of the induction, we prove the claim for $k = 1$. If instance \mathcal{I} has no solution, then, by Fact 3.2, at least one J-job is not completed by slot $n - 1$. If the response time of this job is less than $n(c + 1)$, then it is scheduled in some time slot in $[n, n(c + 1) - 1]$, and therefore, one A-job will be scheduled after time $n(c + 1)$. Assume the claim is true until phase $k - 1$. We therefore have $k - 1$ A-jobs released before time $(k - 1)n(c + 1)$ not completed by time slot $kn(c + 1)$. All J-jobs released in Stage 1 of phase k are scheduled in at least $n + 1$ different slots before time $(k + 1)n(c + 1)$. Since a time slot cannot accommodate an A-job together with a J-job, it then follows that at least k of the A-jobs released by time $(k + 1)n(c + 1)$ will be scheduled not earlier than $(k + 1)n(c + 1)$. ■

We therefore have $cn + 1$ A-jobs released by time $(cn + 1)n(c + 1)$ scheduled after this time. The max response time of one A-job is therefore at least $cn + 1$. ■

This result shows that the parallel scheduling problem posed is computationally hard to approximate as well. Therefore, in order to prove approximation results we require to use the techniques of resource augmentation as done in Section 3.3.

3.2.3 Offline scheduling problem

We will now study the offline version i.e., when all arrival times are known apriori. Using this, we will get lower bounds on optimum values of certain QoS metrics which will be a benchmark to compare against online algorithms.

Deadlines scheduling problem We will focus on a particular variant of the problem, namely, that of meeting of deadlines. Here, each request j has an arrival time a_j as well as a deadline d_j . As before, for each request j , at time a_j we know its size s_j and the channel gain g_j . The goal is to merely test feasibility, i.e., determine if there is a valid schedule that meets all deadlines. This problem is the technical core of many other scheduling problems.

We can write the solution to the deadlines scheduling problem as a combinatorial optimization program as shown in Table 3.2.3.

Table 3.1 Offline scheduling programs

Time-Indexed Program (integral)	Interval-Indexed Program (fractional)
<p style="text-align: center;">maximize 1 subject to</p> $\sum_{j=1}^n c(j, t) \leq C \quad \forall t \quad \sum_{j=1}^n p(j, t) \leq P, \quad \forall t$ $\sum_{t=a_j}^{d_j-1} W c(j, t) \log \left(1 + \frac{p(j, t) g_j}{c(j, t)} \right) \geq s_j, \quad \forall j$ $\sum_{t < a_j, t \geq d_j} c(j, t) = 0, \quad \forall j$ $\sum_{t < a_j, t \geq d_j} p(j, t) = 0, \quad \forall j$ $c(j, t), p(j, t) \text{ discrete values}$	<p style="text-align: center;">maximize 1 subject to</p> $\sum_{j=1}^n c(j, k) \leq C \quad \forall k \quad \sum_{j=1}^n p(j, k) \leq P, \quad \forall k$ $\sum_{t=a_j}^{d_j-1} W \tau_k c(j, k) \log \left(1 + \frac{p(j, k) g_j}{c(j, k)} \right) \geq s_j, \quad \forall j$ $\sum_{k < a_j^{-1}, k > d_j^{-1}} c(j, k) = 0, \quad \forall j$ $\sum_{k < a_j^{-1}, k > d_j^{-1}} p(j, k) = 0, \quad \forall j$ $c(j, k), p(j, k) \geq 0, \quad \forall j, k$

Here, $c(j, t)$ denotes the number of codes assigned to user j in time slot t and let $p(j, t)$ be the total power assigned to user j in time slot t over all the codes. This is called the *time-indexed program* in the table.

It is easy to see the following result.

Lemma 3.2 *The integral time-indexed program has a feasible solution if and only if there exists a valid schedule for the deadlines problem in which all deadlines are met.*

This is an NP-hard problem as proved earlier, since $c()$ and $p()$ take on only discrete values. Hence, we relax the variables to be continuous and then show that the relaxed problem is tractable.

Fractional time-indexed program We relax the integer program by allowing $c(j, t)$ and $p(j, t)$ to be fractional.

Theorem 3.3 *There exists a pseudo-polynomial time algorithm to solve the above problem.*

Proof: We first need to show that the constraint set is convex. The Hessian \mathbf{H} for the rate function $R_u(p, c) = Wc \log(1 + \frac{gp}{c})$ is negative semidefinite, and therefore, the function is concave in (p, c) (albeit not *strictly* concave) [38]. The other constraints are linear and hence the program is convex. There exist polynomial time solutions for convex programming problems that test feasibility [38]. For this program, the running time is polynomial not in input size $(n, \log s_i, \text{etc.})$ but rather is polynomial in the number of variables which is bounded by the size of the numbers in the input (i.e., s_j). ■

Interval-indexed program Though we have relaxed the integer programming problem to the pseudo-polynomial time algorithm, the problem size is still too big. In particular, the number of variables is $O(nT)$ where n is the number of requests, and T is the total length of the schedule that could be large depending on request sizes (ideally, we would like to have the number of variables depend only on n , the number of requests). Next we

consider decreasing the number of variables used in the convex program. We will define a new program below called the *interval-indexed convex program*.

An *event* is either the arrival or the deadline of a request in the system. Consider the sorted list of the events t_1, \dots, t_K . We divide the time scale into *intervals* where an interval is the time period between any two consecutive events, that is interval I_k contains $[t_k, t_{k+1})$. For n requests, the total number of intervals is at most $2n$. We will look for *sliver* solutions, that is, ones in which for each interval I , each user j gets power $p(j, t)$ and $c(j, t)$ for $t \in I$ that remains constant for all $t \in I$; that is, $p(j, t_1) = p(j, t_2)$ for $t_1, t_2 \in I$ and likewise for $c()$. Let $c(j, k)$ be the fractional number of codes and $p(j, k)$ be the fractional power assigned to job j in interval k *per time slot*. Let a_j^{-1} denote the interval at the beginning of which job j arrives in the system, and d_j^{-1} be the interval at the end of which its deadline lies.

The convex programming formulation for solving the scheduling problem with slivers is given in Table 3.2.3.

Theorem 3.4 *The time-indexed convex program has a feasible solution if and only if the interval-indexed convex program has a feasible solution. It can be solved in time polynomial in n, C using the convex program above.*

Proof: We will show that if the time indexed convex program has a feasible solution, so does the interval-indexed convex program; the other direction is trivial.

Let $\overline{p(j, t)}$ and $\overline{c(j, t)}$ be the power and code assignments, respectively, at time frame t to user j in the time-indexed convex program. Therefore, these values satisfy all the constraints of the time-indexed convex program. We now claim that $p(j, k) = \frac{\sum_{t \in k} \overline{p(j, t)}}{\tau_k}$ and $c(j, k) = \frac{\sum_{t \in k} \overline{c(j, t)}}{\tau_k}$ are feasible values in the interval-indexed convex program for user j in interval k , for all users and intervals. That is, these values would satisfy the constraints of the interval-indexed convex program. Clearly we can bound $\sum_j p(j, k)$ as

$$\sum_j \frac{\sum_{t \in k} \overline{p(j, t)}}{\tau_k} = \sum_{t \in k} \frac{\sum_j \overline{p(j, t)}}{\tau_k} \leq \sum_{t \in k} \frac{P}{\tau_k} \leq P.$$

So the power constraint is satisfied; similarly, the code constraint is satisfied too. We have

$$\frac{1}{\tau_k} \sum_{t \in k} \overline{c(j, t)} \log\left(1 + \frac{\overline{g_j p(j, t)}}{c(j, t)}\right) \leq c(j, k) \log\left(1 + \frac{g_j p(j, k)}{c(j, k)}\right)$$

due to Jensen's inequality for multidimensional functions which states that $\mathbb{E}[f(\mathbf{X})] \leq f(\mathbb{E}[\mathbf{X}])$ for a concave function $f(\cdot)$ [39]. Therefore, the demand constraint is satisfied which completes the proof. ■

The result above exposes an interesting structural property of the interval indexed convex program; i.e., the structure that sliver assignment of power and code to requests is optimal in the fractional case.

Using the deadlines scheduling problem: Using the deadlines scheduling problem, other scheduling problems can be solved near optimally. For minimizing the maximum response time (max-flow), we would start by guessing a target response time F , and checking if there is a schedule in which the maximum response time for any request is at most F . This can be reformulated as the deadlines scheduling problem, because in order for a job i to have response time at most F , it must have deadline $a_i + F$, which is the bound on the completion time and, therefore, is a deadline. Now our deadline scheduling problem can be used to check the feasibility of target F . If the target cannot be met, we choose a larger value of F and continue. Otherwise, we decrease the estimate F and continue. An efficient solution is to perform a binary search with the target response time value. That gives an efficient (polynomial time) algorithm to optimize the maximum response time. We will call this algorithm OPT_{frac} , because it assigns fractional values of codes using the convex program described earlier. Indeed, the same approach works for optimizing quality of service criteria such as $\max_i f(c_i - a_i)$ for any monotonic increasing function f .

3.3 Online Heuristics

In this section we present our main algorithmic results, namely, a set of online algorithms for optimizing the metrics of our interest. Recall that, at any instant, an online

algorithm has to take all its scheduling decisions without any assumption on the requests that will be presented in the future. We measure the performance of an online algorithm using the ratio of the value of the objective function (here, *max response time*) achieved by the online algorithm and the optimal value, which can be computed offline.

In our analysis we also use *resource augmentation* [40]. That is, we compare the optimum (i.e., the solution found by adversary) with the value of the solution found by the online algorithm when it is provided with more resources than an adversary who can serve it optimally. Formally, we say that algorithm A for our scheduling problem is an $(\alpha, \beta, \gamma, \delta)$ approximation if it provides a β approximation of the optimum when the sizes of user requests are scaled down by a factor α and the number of codes (the power) used by the algorithm is at most γ (δ , respectively) times the number of codes (the power, respectively) used by the optimum solution to serve the original input sequence.

Before proceeding with our results we provide a general rule that, an $(\alpha, \beta, \gamma, \delta)$ approximation algorithm A , provides a $(1, \beta, \alpha\gamma, \alpha\delta)$ approximation algorithm A' . Algorithm A' is as follows: first apply A to the given instance and let S be the obtained solution that allocates power and codes to users. For each slot $x \in S$, A' uses α copies of x and allocates these slots in the same way as x . Clearly, the set of new slots allows to answer α times the demand satisfied by x . Notice that A' uses α times more codes and total power than A . This implies that all results stated for algorithms working on requests of reduced size can be transformed into results for algorithms working on the original instance of the problem at the expense of some extra codes and power allocated by the system. Therefore, we will not be concerned with directly forcing α to be 1 in our algorithms and their analysis, since no generality is lost in the process.

3.3.1 Minimizing the maximum response time

It is well known in processor scheduling literature [32] that the online algorithm earliest release time (ERT) or first in first out (FIFO) is optimal for minimizing the maximum response time. Therefore, it is natural to ask how FIFO would perform in our case. The simple FIFO strategy in our case allocates all the codes and power to one

user at a time till the user completes the job. Using the “equipartition of power” lemma (Lemma 3.1), this translates into giving the user a power per code of P/C in consecutive time slots that the user completely occupies. Therefore, we study the online scheme where each user is given a power P/C per code and is served by a FIFO scheduling discipline. We call this scheduling discipline $FIFO_{cont}(\frac{P}{C})$. We compare this algorithm against an optimal algorithm OPT_{cont} , that gives a continuous rate/power allocation on an integral number of codes. We first we show a negative result which demonstrates that the $FIFO_{cont}(\frac{P}{C})$ strategy could be arbitrarily worse than the optimal strategy.

Theorem 3.5 *If the maximum response time of the scheduling discipline $FIFO_{cont}(\frac{P}{C})$, on an instance \mathcal{I} of continuous job arrivals, is denoted by $f^{FIFO_{cont}}(\mathcal{I})$ and the optimal discipline has a maximum response time of $f^{OPT_{cont}}(\mathcal{I})$, then*

$$\max_{\mathcal{I}} (f^{FIFO_{cont}}(\mathcal{I})/f^{OPT_{cont}}(\mathcal{I})) > M, \quad \forall M.$$

Proof: The maximum power and codes available in a slot are P and C , respectively. Let $C - 1$ jobs arrive in a batch every time slot such that they need $\frac{P+\delta}{C}$ power each and one code to complete and such that $0 < \delta \leq \frac{P}{C-1}$, which implies that the jobs can be scheduled in one time slot. Such a sequence would be scheduled in two time slots by $FIFO_{cont}(\frac{P}{C})$, since it assigns two codes for each job, and therefore would need $2C - 2$ codes for every $C - 1$ jobs. Hence, the job batch that arrives at the M th time slot is scheduled in the same time slot by the optimum and therefore has a response time of 1, whereas the online $FIFO_{cont}(\frac{P}{C})$ serves this job set only after $2(M - 1)$ time slots have elapsed. Hence, this job set has a response time $2 + 2(M - 1) - M = M$, for any M . ■

In spite of the negative results above, we can show that $FIFO_{cont}(\frac{P}{C})$ can achieve the optimum if every request is reduced to 50% of its original size. To do this we need the following lemma.

Lemma 3.3 *Let the optimal discipline on an online job arrival instance \mathcal{I} give a power assignment $p_j^{OPT_{cont}}(\mathcal{I})$ and code assignment $k_j^{OPT_{cont}}(\mathcal{I})$ to the j th job. Let us denote by \mathcal{I}' the instance where each job size s_j in instance \mathcal{I} is reduced to $s_j/2$, and denote by*

$k_j(\mathcal{I}')$ the code assignment to job j by the scheduling discipline $FIFO_{cont}(\frac{P}{C})$ applied to \mathcal{I}' . Then

$$\frac{k_j(\mathcal{I}')}{C} \leq \max \left\{ \frac{p_j^{OPT_{cont}}(\mathcal{I})}{P}, \frac{k_j^{OPT_{cont}}(\mathcal{I})}{C} \right\}. \quad (3.5)$$

Proof: For each job j of size s_j on the original instance \mathcal{I} , let the pair p_j^{GS}, k_j^{GS} be such that

$$(p_j^{GS}, k_j^{GS}) = \operatorname{argmin}_{(p_j^c, k_j^c): s_j \leq Wk_j^c \log(1 + \frac{g_j p_j^c}{k_j^c})} \left[\frac{p_j^c}{P} + \frac{k_j^c}{C} \right], \quad (3.6)$$

where p_j^{GS}, k_j^{GS} are the total power and codes assigned to user j . This solution allocates a power per code $p_j^s = p_j^{GS}/k_j^{GS}$ to the j th job/user. We now show that allocating power P/C per code is not much worse in terms of this criterion. For each code that uses power p_j^s we allocate $r_j^s = \lceil \frac{p_j^s}{P/C} \rceil$ codes with power $\frac{P}{C}$ in each code. Since $r_j^s \geq \frac{p_j^s}{P/C}$,

$$\begin{aligned} s_j &\stackrel{(a)}{\leq} W \log(1 + p_j^s g_j) \leq W \log \left(1 + r_j^s \frac{P}{C} g_j \right) \\ &\leq W r_j^s \log \left(1 + \frac{P}{C} g_j \right), \end{aligned} \quad (3.7)$$

where (a) is due to (3.6). Hence, by using this allocation the demand s_j of each user j is satisfied. As a result we can give $k_j^{alloc} \stackrel{def}{=} k_j^{GS} \lceil \frac{p_j^{GS}/k_j^{GS}}{P/C} \rceil$ codes of power $\frac{P}{C}$ and still complete the job. Therefore, for the P/C allocation, if we use k_j^{alloc} codes for the job j , we obtain,

$$\frac{k_j^{alloc}}{C} = \frac{k_j^{GS}}{C} \left\lceil \frac{p_j^{GS}/k_j^{GS}}{P/C} \right\rceil \leq \left(\frac{p_j^{GS}}{P} + \frac{k_j^{GS}}{C} \right). \quad (3.8)$$

But we have $p_j^{alloc} = k_j^{alloc} P/C$, and hence,

$$\frac{1}{2} \left(\frac{p_j^{alloc}}{P} + \frac{k_j^{alloc}}{C} \right) = \frac{k_j^{alloc}}{C} \leq \left(\frac{p_j^{GS}}{P} + \frac{k_j^{GS}}{C} \right). \quad (3.9)$$

We now relate this to the optimal solution on the instance \mathcal{I} . Let us denote by $p_j(\mathcal{I})$ and $k_j(\mathcal{I})$, respectively, the total power and the number of codes allocated by the P/C allocation when applied to the instance \mathcal{I} . Therefore,

$$\begin{aligned}
\frac{k_j(\mathcal{I}')}{C} &\leq \frac{1}{2} \frac{k_j(\mathcal{I})}{C} & (3.10) \\
&\leq \frac{1}{2} \left(\frac{p_j^{GS}(\mathcal{I})}{P} + \frac{k_j^{GS}(\mathcal{I})}{C} \right) \\
&\leq \frac{1}{2} \left(\frac{p_j^{OPT_{cont}}(\mathcal{I})}{P} + \frac{k_j^{OPT_{cont}}(\mathcal{I})}{C} \right) \\
&\leq \max \left\{ \frac{p_j^{OPT_{cont}}(\mathcal{I})}{P}, \frac{k_j^{OPT_{cont}}(\mathcal{I})}{C} \right\}
\end{aligned}$$

■

The optimal assignment need not necessarily assign equal power per code across time slots it schedules the j th user. However, due to the joint concavity of the rate in terms of power and code assignment (see proof of Theorem 3.3), the rate for a given user can only increase by giving equal power assignment per code across time slots, provided the power constraint is satisfied. Therefore, the optimal solution has a tighter constraint than the minimization in (3.6), and hence, the third inequality in (3.10) is satisfied.

The algorithm schedules the jobs in order of release time, assigning a number of codes k_j^{red} with power allocation P/C until 50% of the original demand is met. Its operation can be summarized as follows:

1. When a request j is presented, find the number of codes k_j^{red} needed to complete demand s_j^r reduced by 50% with $\frac{P}{C}$ power per code.
2. When a job is completed select the pending user request, if any, with earliest release time a_j .

Theorem 3.6 *Let the optimal discipline on an online job arrival instance \mathcal{I} have a maximum response time of $f^{OPT_{cont}}(\mathcal{I})$. Let us denote by \mathcal{I}' the instance where each job size s_i in instance \mathcal{I} is reduced to $s_i/2$, and denote the maximum response time of the scheduling discipline $FIFO_{cont}(\frac{P}{C})$ applied to \mathcal{I}' by $f^{FIFO_{cont}}(\mathcal{I}')$. Then*

$$f^{FIFO_{cont}}(\mathcal{I}') \leq f^{OPT_{cont}}(\mathcal{I}) + 2, \forall \mathcal{I}. \quad (3.11)$$

Proof: Consider the request r achieving the maximum response time for the scheduling discipline $FIFO_{cont}(\frac{P}{C})$ applied to instance \mathcal{I}' . Without loss of generality, we assume that request r is the last request presented to the algorithm. Request r has been released in slot t_r and completed in slot $C_r(\mathcal{I}')$ in the algorithm's solution. Denote by t (the renewal time) the last slot in which all requests that have been presented before time t have been completed by slot t . We can restrict our attention to the subset of user requests, denoted by $\mathcal{J}' = \{j \in \mathcal{J} | a_j \geq t\}$, that have been presented at or after slot t , since these are the only requests that contribute to the response time of request r . The completion time for request r using the $FIFO_{cont}(\frac{P}{C})$ discipline on instance \mathcal{I}' is at most $C_r(\mathcal{I}') \leq t + \left\lceil \frac{\sum_j k_j(\mathcal{I}')}{C} \right\rceil$, where $k_j(\mathcal{I}')$ is the number of codes required to complete job j on instance \mathcal{I}' . Denote by s the user request completed last in the solution of the optimum on instance \mathcal{I} . Request s has been released at some time $t_s \leq t_r$, and therefore, $C_s^{OPT_{cont}}(\mathcal{I}) - t_s \geq t - t_s + \Psi$, where

$$\Psi = \max \left\{ \left\lceil \frac{\sum_j p_j^{OPT_{cont}}(\mathcal{I})}{P} \right\rceil, \left\lceil \frac{\sum_j k_j^{OPT_{cont}}(\mathcal{I})}{C} \right\rceil \right\}.$$

Now, we have

$$\begin{aligned} f^{FIFO_{cont}}(\mathcal{I}') &= C_r(\mathcal{I}') - t_r & (3.12) \\ &\leq t - t_r + \left\lceil \frac{\sum_j k_j(\mathcal{I}')}{C} \right\rceil \stackrel{(a)}{\leq} t - t_s + \Psi \\ &\leq C_s^{OPT_{cont}}(\mathcal{I}) - t_s + 2 \leq f^{OPT_{cont}}(\mathcal{I}) + 2, \end{aligned}$$

where (a) follows from Lemma 3.3 giving us the result. ■

This result shows that by reducing the demand, we can prove a positive result on $FIFO_{cont}(\frac{P}{C})$. Thus, $FIFO_{cont}(\frac{P}{C})$ is a $(1/2, 1, 1, 1)$ approximation algorithm of OPT_{cont} . In terms of augmentating power and codes, this translates to a $(1, 1, 2, 2)$ approximation algorithm of OPT_{cont} . We can generalize this by allowing for varying levels of resource augmentation to show another positive result on $FIFO_{cont}(\frac{P}{C})$.

Theorem 3.7 *Let the optimal discipline on an online job arrivals instance \mathcal{I} have a maximum response time of $f^{OPT_{cont}}(\mathcal{I})$. Denote by $f^{FIFO'_{cont}}(\mathcal{I})$ the maximum response*

time of the scheduling discipline $FIFO_{cont}(\frac{P'}{C'})$ applied to \mathcal{I} with power $P' \leq \frac{\lceil \kappa C \rceil}{(\kappa-1)C}P$ and $C' \leq \lceil \kappa C \rceil$ codes. Then

$$f^{FIFO'_{cont}}(\mathcal{I}) \leq f^{OPT_{cont}}(\mathcal{I}), \quad \forall \mathcal{I}. \quad (3.13)$$

Proof: Let the optimal scheme on instance \mathcal{I} allocate power $p_j^l(i)$ to user j , on the i th code in time slot l . Let us assign $r_j^l(i) = \lceil \frac{(\kappa-1)p_j^l(i)}{P/C} \rceil$ codes with power $\frac{P}{(\kappa-1)C}$ per code. Due to the power constraint we have for the l th time slot: $\sum_j \sum_{i=1}^C p_j^l(i) \leq P$. Using the fact that $(1+x)^a \geq 1+ax$, $a \geq 0$, and since $r_j^l(i) \geq \frac{(\kappa-1)p_j^l(i)}{P/C}$, we have

$$\begin{aligned} W r_j^l(i) \log \left(1 + \frac{P}{(\kappa-1)C} g_j \right) &\geq W \log \left(1 + r_j^l(i) \frac{P}{(\kappa-1)C} g_j \right) \\ &\geq W \log(1 + p_j^l(i) g_j). \end{aligned} \quad (3.14)$$

Thus, the rate achieved by $r_j^l(i)$ codes with power $\frac{P}{(\kappa-1)C}$ per code is greater than or equal to the rate achieved by the optimal scheme with power $p_j^l(i)$. For every slot l , we can now easily give an upper bound to the total amount of power number of codes needed with this allocation:

$$\sum_j \sum_{i=1}^C r_j^l(i) \leq \sum_j \sum_{i=1}^C \left\lceil \frac{(\kappa-1)p_j^l(i)}{P/C} \right\rceil + C \leq \kappa C \leq \lceil \kappa C \rceil, \quad (3.15)$$

where the second inequality is due to the power constraint. Therefore, there exists a P'/C' allocation which achieves the same schedule as the optimal using power $P' \leq \frac{\lceil \kappa C \rceil}{(\kappa-1)C}P$ and codes $C' \leq \lceil \kappa C \rceil$. The problem of scheduling jobs with P'/C' power per code is like a single processor scheduling problem [32], and FIFO is optimal for this problem with respect to maximum response time, proving the result. \blacksquare

For the rest of this dissertation, we will consider the case where $\kappa = 2$, resulting in an allocation of power P/C per code for the $FIFO_{cont}(\frac{P'}{C'})$ algorithm.

Corollary 3.1 *Let the optimal discipline on an online job arrivals instance \mathcal{I} have a maximum response time of $f^{OPT_{cont}}(\mathcal{I})$. Denote by $f^{FIFO'_{cont}}(\mathcal{I})$ the maximum response time of the scheduling discipline $FIFO_{cont}(\frac{P'}{C'})$ applied to \mathcal{I} with power $P' \leq 2P$ and $C' \leq 2C$ codes. Then*

$$f^{FIFO'_{cont}}(\mathcal{I}) \leq f^{OPT_{cont}}(\mathcal{I}), \quad \forall \mathcal{I}. \quad (3.16)$$

3.3.2 The discrete case

In this section, we show how to transform our algorithms for the continuous case into algorithms for the discrete case. Our transformation preserves the approximation of the algorithms in the continuous case at the expense of some extra codes and some extra power allocated in each slot.

In the continuous case, we assign power P/C to every code, but this may correspond to a non feasible transmission rate at the receiver for some specific user. To move from the continuous to the discrete case, we need to round the power assignment to a value that sustains one of the discrete transmission rates. For this allocation, we impose a further regularity condition that the lowest discrete rate $R(1) \leq W \log(1 + \frac{gP}{C})$; i.e., there is a feasible discrete rate below this power allocation. Though this is perhaps a little more stringent than required, it makes the analysis simpler. As before, this restriction can be removed in practice by using error-correcting codes on a group of codes, so that the combined rate is a feasible discrete rate.

We will implement a rounding scheme that allows us to turn a solution for the continuous case into a solution for the discrete case. We perform two different kinds of rounding of a power z assigned to a code:

1. *Round up*: If there exists a power $\bar{z}_1 \in (z, mz]$ corresponding to a discrete rate, then assign power \bar{z}_1 to the code. The value m is a system parameter, and is a constant.
2. *Round down*: If there exists a power $\bar{z}_2 \leq z$ corresponding to a discrete rate, then assign power \bar{z}_2 per code.

Then for each user we choose the rounding that gives the higher rate if the user were given all the resources, i.e., all the power and codes. In the continuous rate allocation scheme described in Subsection 3.3.1, all codes allocated to user j are assigned with power $x = P/C$. Now, we show that the approximations shown for the continuous case can be translated to the discrete rate by additional resource augmentation. The idea is

to show that additional power and codes needed, for the rounding procedure above to be as good as the continuous case, is bounded.

Theorem 3.8 *Let the $FIFO_{cont}(\frac{P}{C})$ discipline on online job arrival instance \mathcal{I} have a maximum response time of $f^{FIFO_{cont}}(\mathcal{I})$. Let the scheduling discipline $FIFO_{disc}(\frac{P}{C})$ be obtained by taking the discipline $FIFO_{cont}(\frac{P}{C})$ and applying the above rounding procedure. Denote by $f^{FIFO_{disc}}(\mathcal{I})$ the maximum response time of $FIFO_{disc}(\frac{P'}{C'})$ applied to \mathcal{I} , where the power and number of codes per time slot have been augmented to P' and C' respectively. Then there exist $P' \leq \max(mP, (2 - \frac{1}{\theta})P)$, $C' \leq 2C$ such that,*

$$f^{FIFO_{disc}}(\mathcal{I}) \leq f^{FIFO_{cont}}(\mathcal{I}), \quad \forall \mathcal{I}. \quad (3.17)$$

Proof: Let the continuous rate allocation scheme allocate a code i with power $x(i)$, where $\sum_i x(i) \leq P$. Note that $x(i) = P/C$ for the $FIFO_{cont}(\frac{P}{C})$ algorithm. Let \mathcal{K}_1 (\mathcal{K}_2) denote the number of codes whose associated power was rounded up (respectively down) in a particular time slot l . Clearly, $|\mathcal{K}_1| + |\mathcal{K}_2| \leq C$. Let the power allocated on each code i after rounding be denoted by $p^{rnd}(i)$, and clearly $p^{rnd}(i) \leq mx(i)$, $i \in \mathcal{K}_1$ and $p^{rnd}(i) \leq x(i)$, $i \in \mathcal{K}_2$. Recall that the discrete rates have the constraint $\frac{R(j+1)}{R(j)} \leq \theta$, and from Fact 3.1, we have the rounded-down rate on code i , $R^{rnd}(i) \geq \frac{1}{\theta}r(i) = \frac{1}{\theta}W \log_2(1 + x(i)g)$. Now, suppose in each time slot, for every code $i \in \mathcal{K}_2$ we assign two codes, one with power $p^{rnd}(i)$ and the other with power $p^{extra}(i)$, where $p^{extra}(i)$ is just enough to give a rate of $r(i) - R^{rnd}(i)$. We have,

$$\begin{aligned} W \log_2(1 + p^{extra}(i)g) &\leq (1 - \frac{1}{\theta})r(i) = (1 - \frac{1}{\theta})W \log_2(1 + x(i)g) & (3.18) \\ \Rightarrow 1 + p^{extra}(i)g &\leq (1 + x(i)g)^{(1 - \frac{1}{\theta})} \\ &< 1 + (1 - \frac{1}{\theta})x(i)g, \quad \theta > 1, \\ \Rightarrow p^{extra}(i) &< (1 - \frac{1}{\theta})x(i) \\ \Rightarrow p^{rnd}(i) + p^{extra}(i) &< (2 - \frac{1}{\theta})x(i) \end{aligned}$$

Also, for each code $i \in \mathcal{K}_1$ (whose power was rounded up) we assign one code. Clearly such an allocation will meet the same demand as the continuous rate $FIFO_{cont}(\frac{P}{C})$ schedule. Hence the schedule is equivalent to the the continuous rate $FIFO_{cont}(\frac{P}{C})$ schedule.

Therefore the result (3.17) is obtained. The only question that remains is how much resource augmentation was done to obtain this. In the new allocation we have used $P' = \sum_{i \in \mathcal{K}_1} p^{rnd}(i) + \sum_{i \in \mathcal{K}_2} (p^{rnd}(i) + p^{extra}(i))$ total power and $C' = |\mathcal{K}_1| + 2|\mathcal{K}_2|$ total codes. But we have,

$$\begin{aligned}
P' &\leq m \sum_{i \in \mathcal{K}_1} x(i) + (2 - \frac{1}{\theta}) \sum_{i \in \mathcal{K}_2} x(i) \\
&\leq \max(mP, (2 - \frac{1}{\theta})P) \\
C' &= |\mathcal{K}_1| + 2|\mathcal{K}_2| \leq 2C.
\end{aligned} \tag{3.19}$$

Hence the new allocation uses at most a power $P' \leq \max(mP, (2 - \frac{1}{\theta})P)$ and a number of codes $C' \leq 2C$. ■

We can also extend this result to the optimal algorithm with discrete rates, giving us the following result.

Corollary 3.2 *Let the optimal discipline (with continuous rate assignments) on a continuous job arrivals instance \mathcal{I} have a maximum response time of $f^{OPT_{cont}}(\mathcal{I})$. Denote by $f^{OPT_{disc}}(\mathcal{I})$ the maximum response time of the optimal scheduling discipline OPT_{disc} , with discrete rate assignments, when applied to \mathcal{I} with power $P' \leq 2P$ and $C' \leq 2C$ codes. Then*

$$f^{OPT_{disc}}(\mathcal{I}) \leq f^{OPT_{cont}}(\mathcal{I}), \quad \forall \mathcal{I}. \tag{3.20}$$

Proof: Apply the same rounding as described for $FIFO_{disc}$ algorithm to transform the OPT_{cont} solution to the OPT_{disc} solution. ■

Combining the results of this section with the results of the previous section, we can relate the $FIFO_{disc}$ algorithm to the OPT_{cont} algorithm in terms of resource augmentation needed.

Theorem 3.9 *Let the optimal discipline on a continuous job arrivals instance \mathcal{I} have a maximum response time of $f^{OPT_{cont}}(\mathcal{I})$. Denote by $f^{FIFO_{disc}}(\mathcal{I})$ the maximum response time of the scheduling discipline $FIFO_{disc}(\frac{P'}{C'})$ when applied to \mathcal{I} with power*

$P' \leq \max(m, (2 - 1/\theta)) \frac{\lceil \kappa C \rceil}{(\kappa - 1)C} P$ and $C' \leq 2 \lceil \kappa C \rceil$ codes. Then

$$f^{FIFO_{disc}}(\mathcal{I}) \leq f^{OPT_{cont}}(\mathcal{I}), \forall \mathcal{I}. \quad (3.21)$$

The results for the various *FIFO* and *OPT* algorithms discussed in this section are summarized in Figure 3.2. The results are shown in the form of $(1, 1, \gamma, \delta)$ resource augmentation, i.e., γP power and δC codes required by an algorithm to meet the maximum response time performance of the latter algorithm, for any given instance of jobs. Later in this dissertation, we assume that $m = \theta = \kappa = 2$ for evaluation purposes.

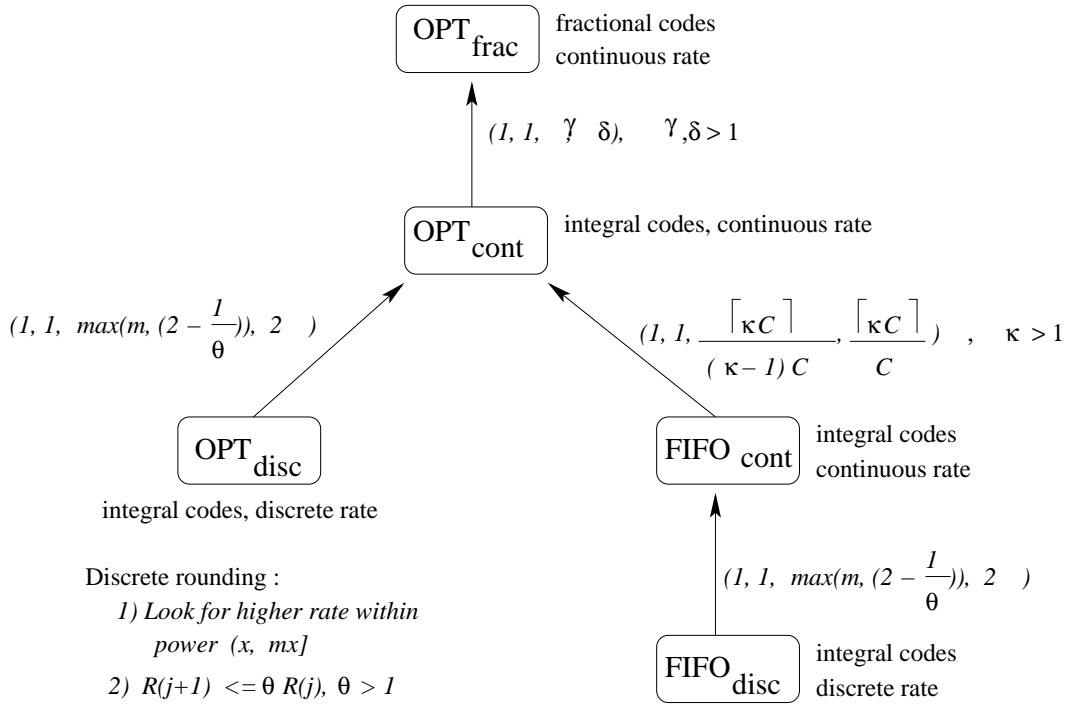


Figure 3.2 Summary of analytical results.

3.3.3 Other QoS criteria

Although we focussed our attention on minimizing the maximum response time in this dissertation, several of our ideas could be extended to other optimization criteria such as *minimizing weighted response time*, $\sum_i w_i (c_1 - a_i)$, where arbitrary weights w_i

are specified for each request i . If $w_i \propto 1/t_i$, where t_i is the time it takes to service the i th request when all codes and power are assigned to request i , the corresponding metric, i.e., $\frac{c_i - a_i}{t_i}$, is known as *stretch* of job i . Stretch has been used in Web server scheduling context for heterogeneous load [32, 41, 42]. While response time is skewed towards large jobs (because jobs with large service times also tend to have large response time), the relative response metric is independent of size, resulting in more fairness for all job classes. Since data requests in the emerging data systems and applications would very likely be heterogeneous, relative response is an attractive metric to investigate. Other weight functions may also be useful, although the two above are most common. Here, we will focus on minimizing average response time.

For average response time, we analyze the shortest remaining processing time (SRPT) algorithm, which is an optimal algorithm for the scheduling problem of minimizing the average response time on a single machine [43]. We will establish the worst-case performance of SRPT when the demand of every user is reduced to $\frac{1}{2(1+\epsilon)}$ of the original demand. In particular, we will show that under this condition SRPT achieves the optimum average response time.

Theorem 3.10 *Algorithm SRPT achieves the optimum average response time if every user demand is reduced to $\frac{1}{2(1+\epsilon)}$ of the original demand.*

Proof: Consider user j and denote by f_j^{red} and by $f_j^{OPT_{cont}}$ the number of frames used by the algorithm working with reduced demands for user j and a lower bound on the number of frames used by the optimum.

From (3.10) it follows

$$f_j^{red} \leq \frac{p_j^r}{P} + \frac{k_j^r}{C} \leq \max\left\{\frac{p_j^{OPT_{cont}}}{P}, \frac{k_j^{OPT_{cont}}}{C}\right\} \leq f_j^{OPT_{cont}}.$$

We know that SRPT on allocation $f_j^{OPT_{cont}}$ gives an optimal solution.

Consider the schedule produced by SRPT on $f_j^{OPT_{cont}}$ and stop processing request j when request j is allocated for f_j^{red} frames. This new schedule has an average response time certainly smaller than SRPT on $f_j^{OPT_{cont}}$. On the other hand, this schedule has an

average response time that is certainly higher than the result of applying SRPT on f_j^{red} for which the theorem follows. ■

The results obtained for minimizing maximum response times in the previous subsections are applicable to SRPT too. If we denote the optimal average response time algorithm with continuous rates and integral number of codes by OPT_{cont} , then we can replace FIFO by SRPT in Theorems 3.8 and 3.9 to obtain similar bounds on resource augmentation.

3.4 Simulation Study

In this section, we experimentally study the performance of our online and offline algorithms for minimizing maximum response time. The results presented in this section are derived from real datasets from Web logs and from synthetic datasets designed to explore certain features of the algorithms. In the rest of this section, we will use the terms maximum response time and max-flow interchangeably.

3.4.1 Online algorithms

The $FIFO_{cont}(\frac{P}{C})$ algorithm was described in Section 3.3. We call this algorithm *FIFO-continuous*. Essentially, this algorithm allots P/C power to each code, and job requests are then scheduled in the order of their arrival.

The rounding procedure for converting the continuous power (rate) algorithm to a discrete power (rate) algorithm was described earlier in Section 3.3.2. Rounding the rates results in a different power per code for each job than in the continuous case. As a result, when codes are assigned to a job in a frame/slot, the packing may not be tight. In other words, some power or codes or both might be unused in a slot. The goal of a discrete-rate online algorithm is to minimize this potential waste of resources in order to reduce the maximum response time.

With this goal in mind, we have developed three online discrete-rate algorithms, which we call *FIFO*, *2D-FIFO*, and *2D-PIKI*. Given a job, the power per code corresponding

to the discrete bit rate is the same for all of these algorithms. They differ only in the way the jobs are selected for receiving service:

- **FIFO:** This is the traditional FIFO algorithm. The request i currently in the system that has the earliest release time a_i is always selected. No other job in the system is scheduled until this job is completed.
- **2D-FIFO:** The request i currently in the system that has the earliest release time a_i has higher priority over other job requests is selected. However, if this job i leaves power/codes unused in that time slot, other jobs j in the system are considered in the non-decreasing order of their release times a_j .
- **2D-PIKI:** The request i currently in the system that has the highest value of power per code p_i is selected. If this job leaves power/codes unused in that time slot, other jobs j in the system are considered in the nonincreasing order of the power per code p_j . This scheme aims to achieve a better packing in each time slot, in order to reduce the completion time.

Due to the discrete nature of the rate set, in certain slots the scheduler may have some codes k^{extra} and some power p^{extra} that cannot be assigned to any job in the system, since the power per code $p_i > p^{extra}$ for all jobs i . In such a situation, the scheduler will choose the first flow that received service in the slot and give it the best possible discrete rate with the remaining power and codes. This is applicable to all the three algorithms described above.

Note that the three discrete algorithms proposed here, no algorithm guarantees that all the power and codes will be used in every slot. Therefore, we expect to see differences between the *FIFO-continuous* algorithm and the three discrete-rate algorithms. In the remainder of this section, we will quantify the differences through simulations.

3.4.2 Simulation setup

In this section, we describe the parameters chosen for our simulations and the tools and data sets used for the same.

3.4.2.1 Channel specifications

We adopt the channel specifications similar to 3G system proposals [15, 14] for our channel model. We would like to emphasize that our algorithms are applicable to all systems that support multiple channels and multiple rates. Such systems include the various next-generation wireless data networks.

We perform experiments on a single cell and abstract the effect of out-of-cell interferers into a decrease in SINR values. The peak power available at the base station was chosen to be $P = 40W$, while the maximum number of channels was chosen as $C = 16$. The power attenuation factor \bar{g}_u for user u is modeled with two components: (a) shadow loss component S , which is a log-normal shadowing variable, and (b) path loss components $P = 1/d^\alpha$, where d is the distance between the base station and the user and α is the distance loss exponent. We chose $\alpha = 3$, giving $\bar{g}_u \propto S/d_u^3$.

The parameters to be used for the rate calculation given in Equation (3.2) were chosen as follows: slot length $\tau = 1.67$ milliseconds, channel bandwidth $\bar{W} = 76.8$ kHz and the coding gain $\Gamma = 4.7$ dB. We operated over an SINR range from -15 dB/Hz to 15 dB/Hz. The discrete rate set used is a set of 15 rates: $\{ 2.4, 4.8, 9.6, 19.2, 38.4, 76.8, 102.6, 153.6, 204.8, 307.2, 614.4, 921.6, 1228.8, 1843.2, 2457.6 \}$ kb/s. Under these restrictions, the maximum data rate for a mobile user in the cell will range from 10 kb/s to 2 Mb/s. The parameters chosen for our simulations are representative of the 3G systems currently under deployment.

3.4.2.2 Data sets and simulation tools

The traces used in the experiments are derived from a single Web-proxy server [44]. For comparing the online algorithms with the offline optimum, we used traces that consist

of up to 100 jobs that arrive over a period of 1 min, where the small size of these traces was primarily due to the computational restrictions on finding the optimal max-flow, i.e. the optimal maximum response time. To evaluate the performance of various online algorithms under heavy demand, we use traces consisting of 4000 jobs arriving over an average period of 30 min, where the requests are generated by 100 users in the cell. In all of the traces used, the minimum request size was 40 bytes, the maximum request size was 500 kbytes, with mean request sizes ranging from 20 to 34 kbytes. The average interarrival time of requests in the traces is 200-400 ms. We also generated some synthetic workloads to explore specific aspects of our algorithms as needed.

3.4.2.3 Simulation tools

In order to compute the optimum max-flow in the offline case (i.e., the OPT_{frac} solution), we used an optimization tool called *LOQO* [45], along with a front-end tool named *AMPL* [46] developed at Bell Labs. *LOQO* is a program for solving smooth optimization problems, and uses an infeasible primal-dual interior-point method applied to a sequence of quadratic approximations to a given problem. *AMPL* is a popular tool used as an interface description tool for many linear/nonlinear optimization programs. Convex programming is a fairly expensive operation, and our experiments were often limited by the program running out of memory for moderate amounts of variables (in the order of several thousands). We used a dual Pentium III processor workstation with 1 GB RAM, for solving the convex problems. We used these runs only for benchmarking purposes. Our online algorithms were evaluated using a custom-built simulator.

3.4.3 Experiments

We performed two kinds of experiments to evaluate our algorithms. The first set of experiments validated our theoretical results and demonstrated some interesting properties of the online algorithms, while the second experiment was designed to measure the average-case performance of our algorithms. In all these experiments, the optimum

solution is the maximum response time obtained by the OPT_{frac} algorithm, which is always better than the OPT_{cont} and the OPT_{disc} solution.

3.4.3.1 Unbounded maximum response time

In Section 3.3, we claimed that $FIFO_{cont}(\frac{P}{C})$ can have unbounded max-flow in certain cases. In this example, we present such a scenario.

Consider a job i of size $s_i = 130$ bits with a maximum possible data rate of $R_i^{max} = 1.227$ Mb/s. In order to complete this job, we need one code and power $p_i = 2.552 > P/C$. Hence, $FIFO_{cont}(\frac{P}{C})$ will allocate $c_i = 2$ and $p_i = 5$ in the continuous case. When rounded to the nearest discrete rate (76.8 kb/s), the discrete algorithm allocates $c_i = 2$ and $p_i = 5.01$.

Thus, if 15 such jobs arrive in every time slot, the optimal algorithm will serve all the jobs within a time slot, while the continuous and discrete versions will need 30 codes for every 15 jobs, and hence will use two time slots to serve them. As a result, the max-flow becomes unbounded, as shown in Table 3.2 for the trace *config1*. In this example, the results of the discrete-rate algorithms are identical, and hence, we elect to show the results for 2D-FIFO only. We denote the optimal maximum response time by OPT .

Table 3.2 Unbounded nature of $FIFO_{cont}(P/C)$ - one slot OPT max-flow

Trace	OPT Max-flow (in slots)	Time	FIFO (cont.)	2D-FIFO
config1	1	10	2	2
		500	14	14

We also present an example where the jobs are such that they cannot be completed in one time slot, even if all the power and codes are assigned to them. This serves to show that the unbounded nature of the online algorithms is not a special case.

Consider a job set as shown in the trace *config2* in Table 3.3. Each job needs at least two time slots for completion if it is the only job in the system. The jobs have been selected such that in the discrete case, their $p_i/k_i = 4.001$, which ensures that 4 units of

power and 7 codes are unused in each slot. Thus, the optimal max-flow for this trace is 28 slots, while the online algorithms produce higher max-flow. We can construct a trace where this set of jobs arrive every 28 slots. Thus, the optimal max-flow will remain at 28 slots, while the max-flow for the online algorithms will grow unbounded. This is shown in Table 3.4. All values are in terms of slots.

Table 3.3 Unbounded nature of $FIFO_{cont}(P/C)$ - large sizes

Trace	Id	Size (in bits)	Max data rate (in kb/s)	p_i/k_i (discrete)
config2	1	400	24.091	4.001
	2	650	48.376	4.001
	3	650	97.524	4.001
	4	650	198.095	4.001
	5	1 200	408.048	4.001
	6	2 000	860.529	4.001
	7	35 000	9 000.895	4.001

Table 3.4 Unbounded nature of $FIFO_{cont}(P/C)$ - max-flow for large sizes

Trace	OPT Max-flow (in slots)	Time	FIFO (cont.)	2D-FIFO
config2	28	7	30	32
		210	79	134

The above example serves to show that it is very difficult to produce deterministic approximation algorithms for minimizing max-flow.

3.4.3.2 Online heuristics

In this section, we will evaluate the different online algorithms and compare their performance against the offline optimal algorithm. We used small Web traces, with 100 jobs arriving over an average period of 1 min, for computing the convex programming lower bound denoted by OPT, for max-flow. The job requests are for users who are

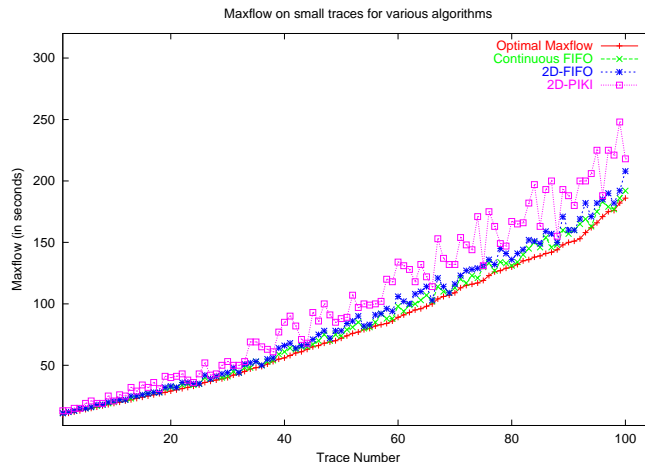


Figure 3.3 On-line heuristics: Comparison with optimal max-flow.

distributed uniformly in the cell. We tested over 1 000 such traces, and present the results for a randomly selected set of 100 traces in Figure 3.3. Since the discrete *FIFO* algorithm performs always worse than the *2D-FIFO* algorithm, we have omitted its results in the figure for purposes of clarity.

It can be seen that the online algorithms perform very close to the optimal, on the average. We also observed that *2D-FIFO* performs the best among the three discrete-rate algorithms and also that *2D-PIKI* performs the worst. In addition, the discrete algorithms always appear to perform worse than the continuous version.

We repeated the same experiment on much larger traces. We selected 36 Web traces with 4000 jobs each, requested by 100 users, arriving over an average period of 40 min. This data represents 24 hours of Web traffic, taken over different periods of time. Due to the large size of the traces, we could not compute the optimal maximum response time, and hence, we present only the results for the *FIFO-continuous* and the *2D-FIFO* and *2D-PIKI* algorithms in Figure 3.4. The results also follow the earlier trend, with the discrete algorithms performing close to the *FIFO-continuous* algorithm.

While these inferences continue to hold true in most instances, they are not always true. Consider the set of jobs shown in Table 3.5. It is very similar to Table 3.3, except

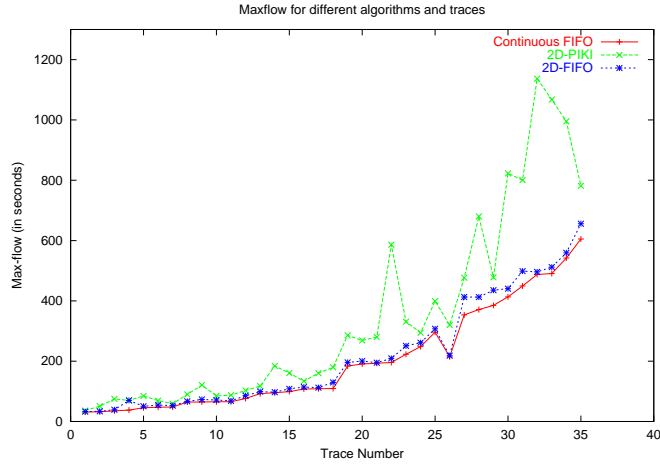


Figure 3.4 On-line heuristics: Large traces.

that there is an additional job: job 1. The trace *config3* consists of this set of 8 jobs arriving at the same time, every 35 slots, for 1040 time slots. The max-flow results for this trace are presented in Table 3.6. There are some interesting observations to be made from this example. The first observation is that *2D-PIKI* performs as well as the optimal algorithm, deviating from its usual poor behavior. Moreover, while one might expect the discrete-rate algorithms to do worse than the continuous case algorithms in all cases, in this particular example, the converse is true. Both *2D-FIFO* and *2D-PIKI* perform much better than the *FIFO-continuous* algorithm.

Table 3.5 Online heuristics: *config3*

Trace	Id	Size (in bits)	Max data rate (in kb/s)	p_i/k_i (discrete)
config3	1	124 000	7 502.72	0.550 1
	2	400	24.091	4.001
	3	650	48.376	4.001
	4	650	97.524	4.001
	5	650	198.095	4.001
	6	1 200	408.048	4.001
	7	2 000	860.529	4.001
	8	35 000	9 000.895	4.001

Based on our experiments, the following observations can be made.

Table 3.6 Online heuristics: max-flow

Trace	OPT Max-flow (in slots)	Continuous FIFO	Discrete		
			FIFO	2D-FIFO	2D-PIKI
config3	35	637	2 595	69	35

- All three discrete algorithms perform very close to the *FIFO-continuous* algorithm, which in turn performs near optimal solution OPT_{frac} , on the average. However, this does not give any guarantees on the worst-case behavior, which can be unbounded. It is also important to note that the optimal maximum response time is computed assuming that codes and powers can be assigned in infinitesimally small units, while the other algorithms use codes in integral units. We have not been able to compute the optimal maximum response time for the integral channel/code case due to the computational complexity of finding such a solution. However, we believe that the optimum in the discrete case will differ significantly from the OPT maximum response time. Hence, the performance of our discrete algorithms will be much better when compared to the discrete optimum, on the average.
- *FIFO* will always perform worse than *2D-FIFO* because the amount of code and power wasted in every slot is greater in the *FIFO* algorithm. On the other hand, *2D-PIKI* performs well only when there is a correct mix of jobs with high and low p_i/c_i ratios. Its performance is therefore highly susceptible to user gains and the pattern of requests.

In the rest of this section, we want to investigate the behavior of the algorithms when they are provided with augmented resources. In particular, our goals are to ensure that our theoretical bounds are obeyed and to measure the actual levels of resource augmentation needed in the practical case. We ignore the discrete *FIFO* algorithm from now on and focus on the other two discrete algorithms.

3.4.3.3 Resource augmentation

In this set of experiments, we will examine the amount of resource augmentation needed for a discrete-rate algorithm to achieve the same max-flow as *FIFO-continuous* and compare it to theoretical bounds given in Theorems 3.8 and 3.9.

We performed our experiments on two sets of traces, as in Section 3.4.3.2. We used several small traces with 100 jobs each, arriving over a period of 1 min. The scheduling algorithms were provided with augmented power in steps $\{P, 1.25P, 1.5P, 1.75P, 2P, 2.5P, 3P, 4P\}$ and augmented number of codes in steps $\{C, 1.5C, 2C, 2.5C, 3P, 4P\}$. For each combination of augmented power and codes, we measured the max-flow at 100% of the demand. The demand was reduced in steps $\{s_i, 0.95s_i, 0.9s_i, 0.85s_i, 0.8s_i, 0.75s_i, 0.7s_i, 0.6s_i, 0.5s_i, 0.4s_i, 0.3s_i, 0.25s_i\}$ until the max-flow for the reduced demand in the discrete code case was lesser than or equal to the optimal max-flow (*OPT*). The results are illustrated in Figures 3.5, 3.6, and 3.7

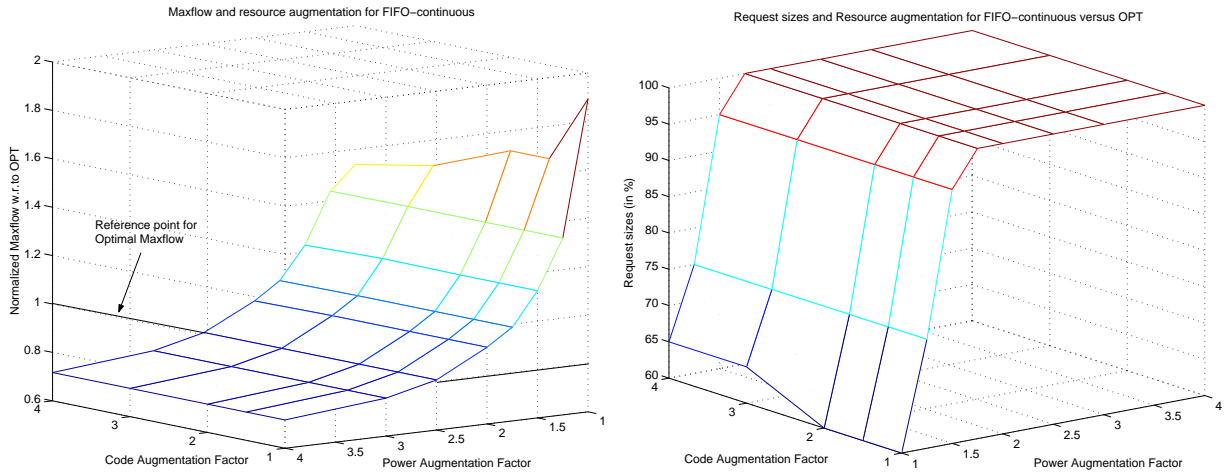


Figure 3.5 Normalized maximum response time and demand reduction with resource augmentation: *FIFO continuous*.

Each figure contains two graphs. The graph on the left-hand side shows the maximum response time for a given algorithm normalized with respect to the optimal maximum response time (*OPT*), for a given combination of augmented resources. The worst-case ratio observed over all of the traces is plotted in this graph. The graph on the right-hand side of each figure represents the combinations of demand reduction and resource

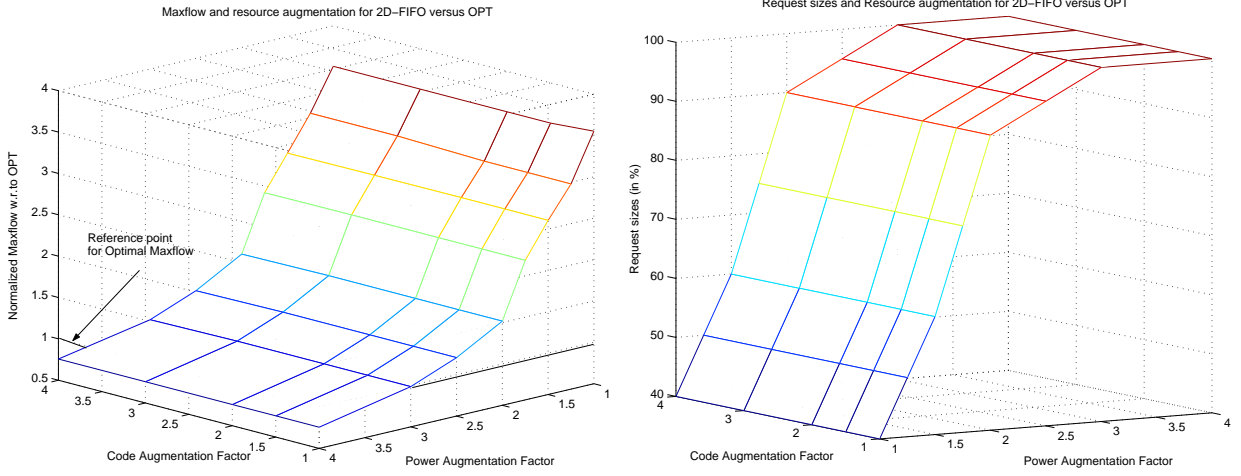


Figure 3.6 Normalized maximum response time and demand reduction with resource augmentation: *2D-FIFO*.

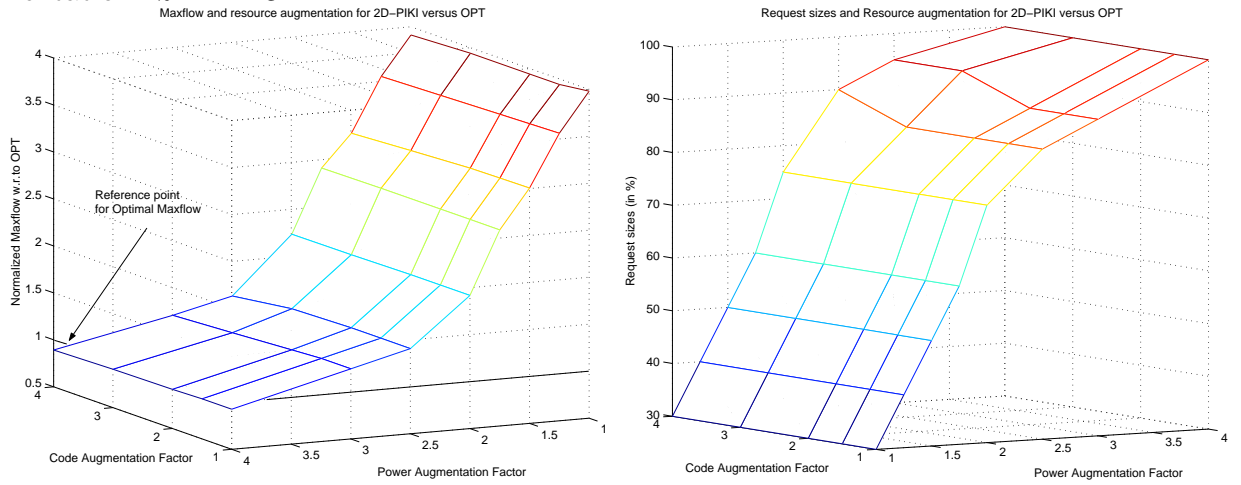


Figure 3.7 Normalized maximum response time and demand reduction with resource augmentation: *2D-PIKI*.

augmentation that give the same value of the maximum response time for the chosen algorithm as the optimal maximum response time. Note that the optimal maximum response time is found without augmenting power or codes.

The results bring out a very interesting fact:

- *Code augmentation and power augmentation are not the same*: The asymmetric nature of the graphs tell us that over-provisioning codes is not very efficient compared to over-provisioning of power.

This is a very important result because power is more readily available than codes in a wireless system. The number of channels/codes in a system depends on the coding and modulation schemes and also has to be standardized. However, the peak power value is limited only by interference constraints and, hence, is easier to augment than the number of codes.

This interesting behavior turns out to be a result of the concave nature of SINR-rate function in the system. Recall that the rates are a function of the SNR at the receiver, and can be approximated by Equation 3.1. We let the gains and other parameters be constants, and we vary the power and codes by multiplicative factors. For a given augmentation factor x , we compute the rate when the power is augmented x times ($r_p(x)$), and the rate when code is augmented x times ($r_c(x)$):

$$\begin{aligned} \lim_{x \rightarrow \infty} r_p(x) &= \lim_{x \rightarrow \infty} W \log_2 \left(1 + \frac{xP}{C} g \right) \rightarrow \infty \\ \lim_{x \rightarrow \infty} r_c(x) &= \lim_{x \rightarrow \infty} Wx \log_2 \left(1 + \frac{P}{xC} g \right) = WPg \end{aligned} \quad (3.22)$$

It is clear that the rate gain from code augmentation diminishes as the augmentation factor increases. The gain is substantial when there are fewer than four codes in the system. However, the rates do not increase by a large factor for higher number of codes. This explains the effectiveness of power augmentation over code augmentation. In addition, code augmentation will help to improve the rate only if the current operating point is at the flat region of the log curve, implying very high gain users who receive few codes and a lot of power. Our algorithms avoid such an operating point since they try to spread out the power among all the codes, resulting in a low operating point.

In Figures 3.8 and 3.9, we present the results of similar experiments with larger traces, where we compared the results for the discretized algorithms against the *FIFO-continuous* algorithm. These traces had 4000 jobs each, requested by 100 users, arriving over a period of 30 min. The users requesting the jobs were uniformly distributed over the region of the cell. The scheduling algorithms were provided with augmented power in steps $\{P, 1.25P, 1.5P, 1.75P, 2P\}$ and augmented number of codes in steps $\{C, 1.5C, 2C, 2.5C\}$.

For each combination of augmented power and codes, we measured the max-flow at 100% of the demand. The demand was reduced in steps $\{s_i, 0.95s_i, 0.9s_i, 0.85s_i, 0.8s_i, 0.75s_i, 0.7s_i, 0.6s_i\}$ until the max-flow for the reduced demand in the discrete case was less than or equal to the max-flow in the continuous case with power P and codes C at 100% of the demand.

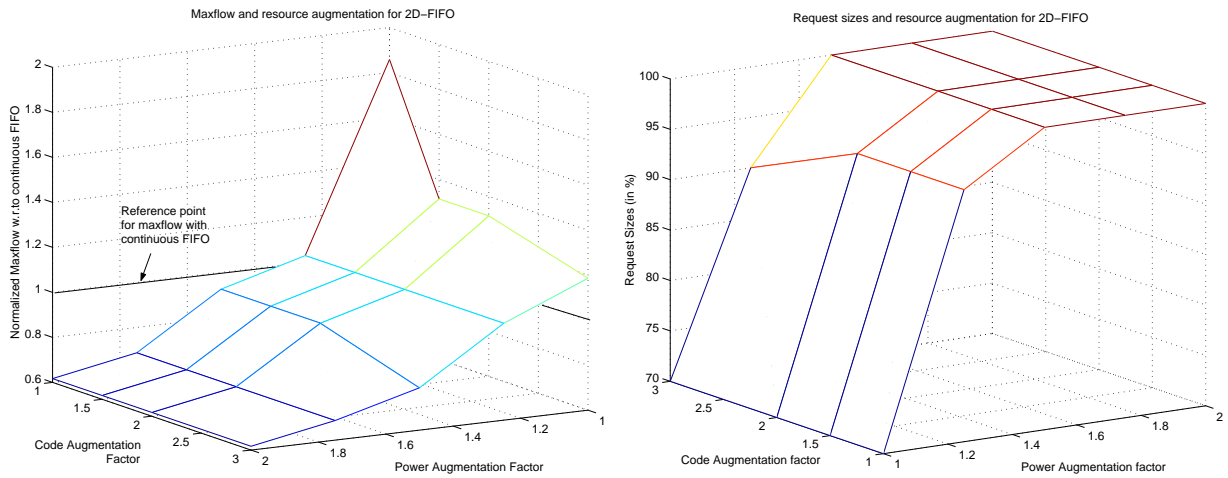


Figure 3.8 Normalized maximum response time and demand reduction with resource augmentation for large traces: *2D-FIFO*.

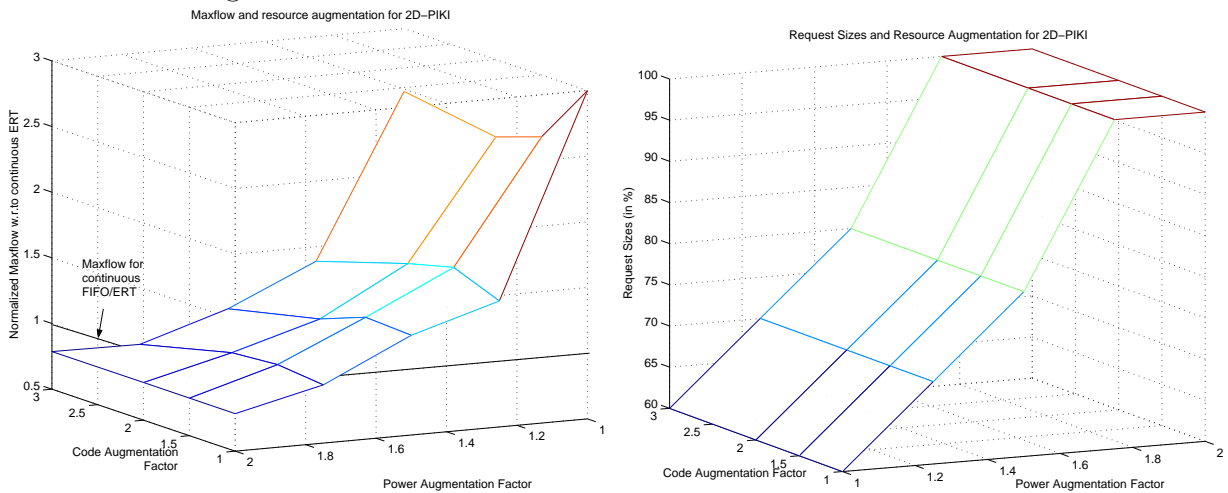


Figure 3.9 Normalized maximum response time and demand reduction with resource augmentation for large traces: *2D-PIKI*.

The results in Figures 3.8 and 3.9 are normalized compared with respect to the maximum response time for the *FIFO-continuous* algorithm without resource augmentation.

The figures reiterate the two conclusions that we inferred in the previous subsection.

- *The average case is vastly better than the worst case.* Our theoretical bounds indicate that maximum response time in the discrete case will equal that of the continuous case with $P' \leq 2P$ and $C' \leq 2C$. From our experimental results, we see that if power is augmented 1.5 times, then the max-flow in the discrete case equals that of the continuous case with power P , without any need for code augmentation. Thus, a system designed for meeting the QoS needs in the average case needs to over-provision its resources by an amount much lesser than that implied by the theoretical bounds.
- *2D-FIFO outperforms 2D-PIKI.* 2D-PIKI is not able to obtain a right mix of jobs with the correct packing ratios in a practical scenario. As a result, its performance suffers when compared to 2D-FIFO.

We would like to state again that even though the discrete optimum was not measured explicitly for these traces, we expect it to be quite close to the *FIFO-continuous* case, and therefore the performance bounds of our algorithms will be much tighter when compared to the discrete optimum.

In conclusion, our experimental results demonstrate that the discrete-rate algorithms proposed in this section perform close to the optimal in the average case, even though the performance ratio is theoretically unbounded in the unaugmented case. We also show that resource augmentation, in particular, power augmentation, will enhance the performance of discrete algorithms.

3.5 Discussion

In this chapter, we have formulated new scheduling problems related to multiple rate, multiple code wireless networks. We focussed our attention on the maximum response time criterion for packet scheduling. However, the formulation and approach can be extended to other criteria such as (weighted) average flow. We proposed online algorithms

that utilize the multicode, multirate feature of 3G/4G wireless networks by effectively assigning power and codes to different users and jobs. We performed experimental results to show that for several cases of practical interest the proposed algorithms perform much better than our worst case analysis shows. Our analyses are valid in the presence of time-varying gains, since we consider resource augmentation for the online algorithm within a given time slot, where the user gains and other parameters are the same for both the optimal solution and the online algorithm. In summary, we have proposed simple online scheduling algorithms that effectively provide fine-grained QoS to the users by utilizing the advanced features of 3G/4G wireless networks.

CHAPTER 4

RELATED WORK

In this chapter, we present a discussion on prior literature related to the topics discussed in this dissertation. We first focus on the fairness and scheduling problem in ad hoc networks, and later look at related work in the area of scheduling for cellular networks.

4.1 Fairness and Scheduling in Ad Hoc Networks

Research on scheduling for ad hoc networks has been in progress for nearly two decades, with initial studies on Aloha packet radio networks [47]. Many studies focused on channel assignment problems in multihop wireless networks, where the channels could be time slots, frequencies, or codes. In [48, 49, 50], TDMA scheduling of broadcasts is considered, whereas in [51, 52, 53], TDMA link scheduling is investigated in detail. Many problems in this domain are NP-complete [54], and hence, these solutions try to come up with efficient approximation algorithms. The issue of distributed scheduling for channel assignment is addressed in [48, 49, 55, 56]. The above work does not consider the impact of mobility on scheduling, which has been addressed in later work [57, 58, 59, 60]. In general, these schemes try to perform graph coloring on the nodes of a multihop radio network, to obtain some minimum throughput and delay bounds. The key features of our work that are distinct from the above are that we consider fairness at the level of flows instead of individual hosts, and that we provide a way of achieving a wide range of fairness functions in a packet radio network regardless of the topology. In [21, 22], a distributed mechanism for achieving max-min fairness in an ad hoc network has been

proposed. The proposed network model is very similar to ours, and they consider a static network with one hop flows as in this dissertation.

In the case of nonstatic ad hoc networks, mobility impacts fairness greatly, since the system of rates will not settle down to an equilibrium, thereby obscuring the meaning of fairness. However, minimum throughput bounds can be obtained if the level of contention is bounded. Hence, we have not considered mobility, whereas [57, 58, 59] have obtained delay and minimum per-node throughput bounds in for a mobile network based on the maximum degree of the nodes.

The capacity of wireless multihop networks determines the best achievable throughput for flows in the system and thus impacts the performance of the scheduling algorithms. In a fundamental work, [61] models the achievable capacity region of a multihop wireless network and shows that average throughput for a node is $\Theta(1/\sqrt{n \log n})$. However, in practice, per-node throughput goes down by a larger factor with the increasing number of nodes, due to a large number of collisions and associated reservation mechanisms. In [62], the authors try to address this issue by proposing a randomized protocol called *SEEDEX* to avoid reserving the channel for every packet and silencing a two-hop neighborhood. They have also described a scheme where a node can control the amount of throughput that it can receive relative to the nodes in its two-hop neighborhood. However, the structure of rates in the entire network is not defined, which is the key issue addressed by this dissertation.

4.2 Scheduling in Next Generation Cellular Networks

There has been a significant amount of work on scheduling problems over cellular networks. In this dissertation, we have studied the downlink scheduling problem. The uplink scheduling problem is a complementary problem where the fundamental issues are quite different. See [63] and references therein for more details.

Typically resource allocation problems study per-user rate throughput. The rate optimization problem has been extensively studied for various wireless system with focus

varying from maximizing overall throughput to providing a minimum throughput guarantee for all users [18, 64, 65, 66, 67, 68]. A good discussion of related work about throughput optimization and fairness in wireless data networks can be found in [69, 70].

Job scheduling is very popular in the context of processor scheduling, and various algorithms have been proposed for different QoS metrics such as completion time, maximum response time, and weighted response time [71]. In the parallel scheduling literature, there has been significant work on scheduling of *malleable* tasks, see for example [33, 34, 35, 72, 73]. There are several differences between the scheduling problem posed in this chapter with the malleable task case, and details are given in Section 3.1.3. In a nutshell, the difference primarily arises because the time required to process a request depends on both the power and number of codes allocated (see Section 3.1.1). Message scheduling in packet switched networks is a problem considered in [74], where the messages are jobs, and are served in units of packets. In this work, two algorithms that are similar to shortest job first (SJF) and shortest remaining processing time first (SRPT) are shown to provide better average response times than FIFO.

In wireless networks, job scheduling has been addressed in the context of downlink *broadcast* scheduling [75, 76]. In [75], a dynamic priority function is proposed as the basis of scheduling messages/jobs, where the priority of a job is dependent on the size of the job, as well as its waiting time. This shortest message preempt (SMP) scheduling algorithm results in a larger average delay than SRPT, but is shown to provide lesser average delays for long messages than SRPT. The authors of [76] look at stretch metrics, i.e. delay per byte transferred. They have proposed the highest density first (HDF) heuristic, which is shown to be a $\frac{1+\epsilon}{\epsilon}$ approximation of the optimal maximum stretch. In a recent work, downlink unicast scheduling in multirate CDMA systems was studied [77]; this is close to our work in spirit. However, they assume a linear rate model for the physical layer which is not accurate. Also, they do not have any upper bound on the number of available codes; hence, they study the problem of allocating power only. We have studied the nuances of allocating both power and codes in this dissertation.

In [78], a scheduling scheme is presented to provide rate guarantees to users in a multi-rate cellular data network. This work has been generalized in [79] to maximize the aggregate utilities of users for a wide class of functions where the utility functions do not have to be concave. The network model is very similar to the one that we have considered in this dissertation, with the rate dependent on the SNR of the user, and power is allotted subject to the peak power constraint. However, the number of codes is assumed to be unlimited in both these schemes, making them impractical. While their approach can be further generalized to include finite codes, their solutions will no longer be optimal. In this dissertation, we have considered a more practical problem where the power and code are both constrained and, we have provided a thorough competitive analysis of the online algorithms, in particular, using resource augmented analysis; this is the first provable result known for any of the online scheduling problems, including the ones in [77].

Rate fairness in multirate CDMA networks has also been studied in [80, 81]. In [80], the throughput $T(i)$ of a user i is compared against the requested rate $R(i)$, and the user with the highest value of $R(i)/T(i)$ is served first. This scheme is shown in [81] to achieve the *proportional fair* allocation [12] in the system. The algorithm does not depend on the existence of multiple rates in the system, and is applicable to single rate systems as well.

CHAPTER 5

CONCLUSIONS AND FUTURE RESEARCH

The growth of the Internet has extended to the wireless domain thereby enabling truly nomadic computing. Wireless devices and applications are becoming increasingly popular, and this is reflected by the widespread adoption of cellular phones, personal digital assistants, and laptops in our society. To provide connectivity to these systems, different kinds of networks are being devised. These can be categorized into two types: (a) cellular networks and (b) ad hoc networks. These two kinds of networks are complementary, with the cellular networks enabling access to the Internet by means of a central controller, and ad hoc networks linking various mobile devices in a distributed manner through a multi-hop wireless network. For any technology to sustain its growth rates, user satisfaction is paramount, and this is achieved by providing QoS assurances to users. The unique characteristics of the wireless domain such as spatially and temporally varying channel conditions along with location dependent contention have made it a challenge to enable the levels of QoS associated with wired networks. QoS mechanisms need basic support at the MAC layer through packet scheduling schemes. Hence, we have focused on the design of novel scheduling schemes for wireless data networks in this dissertation.

In ad hoc networks, each host has a different view of the network than its neighbors or other nodes in the network. As a result, when a host tries to obtain its “fair” share of the network bandwidth, the other hosts might think of it as unfair. A fundamental notion of fairness is hard to achieve in such systems due to the problem of channel reuse and location dependent contention. In this dissertation, we have proposed a framework to capture the characteristics of the shared wireless channel and the contention levels experienced by various flows in the network. Using this framework, a general scheme

to apply different fairness models is described. An instantiation of this scheme for the *proportional fairness* criterion is presented. Our work is the first in this area to identify this problem, and we have presented an elegant solution for the same.

Cellular networks have greater control over assigning bandwidth than ad hoc networks. The next generation 3G/4G cellular networks are migrating towards higher bandwidths and novel coding/modulation schemes designed to increase spectral efficiency and reuse of the wireless channel. In addition, users can receive on multiple channels and at different rates that are dependent on the quality of their channel. This leads to interesting resource allocation problems for optimizing various QoS metrics. It is possible to apply existing algorithms for job scheduling and fair rate scheduling to these systems, but due to the nonlinear nature of the resource allocation problem, such solutions will be far from optimal. In this dissertation, we have considered one type of a job scheduling problem, minimizing maximum response time. We have proved that it is not possible to generate approximation algorithms with constant performance ratios in these systems. Consequently, we have applied a new analysis technique called *resource augmented analysis* to design practical algorithms. Our analysis technique can be used to quantify the amount of resource overprovisioning necessary for a system to satisfy certain QoS constraints. We show through simulations that our algorithms work very well in practical instances and obey theoretical limits.

While our contributions in this dissertation are both novel and practical, the research areas considered here are new and relatively unexplored. We present some directions for future research below.

- *Fairness for multihop flows in ad hoc networks:* In this dissertation, we considered only single-hop flows, and used purely MAC layer mechanisms to achieve fairness. For multihop flows, throughput fairness has to involve end-to-end mechanisms such as congestion control and rate adaptation on an end to end basis. Thus, any study of fairness at this level has to incorporate the transport layer and the MAC layer, along with the notion of channel capacity for the entire ad hoc network. This

is a nontrivial problem, albeit one that is needed to improve the QoS for ad hoc networks.

- *Rate scheduling in next generation cellular networks:* Most research on multirate cellular systems has focused on improving the spectral efficiency. Paying users eventually expect rate service guarantees from service providers. Therefore, it is important to study rate scheduling in this context using a practical network model, such as the one presented in this dissertation. It will also be of interest to investigate job scheduling strategies for various QoS measures other than max-flow. The analytical tools and models presented in this dissertation will prove useful in this regard.

REFERENCES

- [1] Jupiter Communications, April 2002, <http://www.jup.com/>.
- [2] IDG Data Consulting, April 2002, <http://www.idc.com/>.
- [3] Bluetooth, August 2001, <http://www.bluetooth.com/>.
- [4] Wireless data forum, June 1999, <http://www.wirelessdata.org/>.
- [5] Wireless LAN Association, April 2000, <http://www.wlana.com/>.
- [6] D. Knisely, S. Kumar, S. Laha, and S. Nanda, "Evolution of wireless data services: IS-95 to CDMA2000," *IEEE Communications Magazine*, vol. 36, pp. 140–149, October 1998.
- [7] The Qualcomm high data rate wireless network, February 2002, <http://www.qualcomm.com/hdr/>.
- [8] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *ACM SIGCOMM*, Austin, TX, August 1989, pp. 1–12.
- [9] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A media access protocol for wireless LANs," in *ACM SIGCOMM*, London, UK, September 1994, pp. 212–225.
- [10] J. Garcia-Luna-Aceves and C. Fullmer, "Floor acquisition multiple access (FAMA) in single-channel wireless networks," *Mobile Networks and Applications*, vol. 4, pp. 157–174, October 1999.
- [11] IEEE, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," IEEE Standard 802.11, June 1997.
- [12] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: Shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, March 1998.
- [13] T. Ozugur, M. Naghshineh, P. Kermani, C. Olsen, B. Rezvani, and J. Copeland, "Balanced media access methods for wireless networks," in *ACM MOBICOM*, Dallas, TX, October 1998, pp. 21–32.
- [14] S. Nanda, K. Balachandran, and S. Kumar, "Adaptation techniques in wireless packet data services," *IEEE Communications Magazine*, vol. 38, pp. 54–65, January 2000.

- [15] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushayana, and A. Viterbi, "CDMA/HDR: A bandwidth-efficient high-speed wireless data service for nomadic users," *IEEE Communications Magazine*, vol. 38, pp. 70–77, July 2000.
- [16] J. Chuang and N. Sollenberger, "Beyond 3G: Wideband data access based on OFDM and dynamic packet assignment," *IEEE Communications Magazine*, vol. 38, pp. 78–87, July 2000.
- [17] T. J. J. Starr and J. Cioffi, *Understanding Digital Subscriber Line Technology*. Upper Saddle River, NJ: Prentice Hall, Inc., 1999.
- [18] S. Lu, T. Nandagopal, and V. Bharghavan, "Fair scheduling in wireless packet networks," in *ACM MOBICOM*, Dallas, TX, October 1998, pp. 10–20.
- [19] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, "WTCP: A reliable transport protocol for wireless wide-area networks," in *ACM MOBICOM*, Seattle, WA, August 1999, pp. 231–241.
- [20] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 756–769, June 1997.
- [21] H. Luo, S. Lu, and V. Bharghavan, "A new model for packet scheduling in multihop wireless networks," in *ACM MOBICOM*, Boston, MA, August 2000, pp. 76–86.
- [22] H. Luo and S. Lu, "A self-coordinating approach to distributed fair queueing in ad-hoc wireless networks," in *IEEE INFOCOM*, Anchorage, AK, March 2001, pp. 1370–1379.
- [23] V. Bharghavan, "Performance analysis of a medium access protocol for wireless packet networks," in *IEEE Performance and Dependability Symposium*, Durham, NC, September 1998, pp. 86–95.
- [24] S. Shenker, "Some conjectures on the behavior of acknowledgement-based transmission control of random access communication channels," in *ACM SIGMETRICS*, Alberta, Canada, September 1987, pp. 245–255.
- [25] "ns-2 Network Simulator," 1998, <http://www-mash.cs.berkeley.edu/ns/>.
- [26] D. B. West, *Introduction to Graph Theory*, 2nd ed., Englewood Cliffs, NJ: Prentice Hall, 2001.
- [27] S. Shenker, "Fundamental design issues for the future internet," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 13, pp. 1176–1188, September 1995.
- [28] S. Kunniyur and R. Srikant, "End-to-end congestion control schemes: Utility functions, random losses and ECN marks," in *IEEE INFOCOM*, Tel Aviv, Israel, March 2000, pp. 1323–1332.
- [29] L. Massoulié and J. Roberts, "Bandwidth sharing: Objectives and algorithms," in *IEEE INFOCOM*, New York, NY, March 1999, pp. 1395–1403.

- [30] T. S. Rappaport, *Wireless Communications, Principles & Practice*. Upper Saddle River, NJ: Prentice Hall, Inc., 1996.
- [31] J. M. Cioffi, G. P. Dudevoir, M. V. Eyuboglu, and G. D. Forney, “MMSE decision-feedback equalizers and coding: Parts I and II,” *IEEE Transactions on Communications*, vol. 43, pp. 2582–2604, October 1995.
- [32] M. Bender, S. Chakraborti, and S. Muthukrishnan, “Flow and stretch metrics for scheduling continuous job streams,” in *ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA, January 1998, pp. 270–279.
- [33] J. Turek, J. L. Wolf, and P. S. Yu, “Approximate algorithms for scheduling parallel tasks,” in *ACM Symposium on Parallel Algorithms and Architectures*, San Diego, CA, June 1992, pp. 323–332.
- [34] W. Ludwig and P. Tiwari, “Scheduling malleable and nonmalleable parallel tasks,” in *ACM-SIAM Symposium on Discrete Algorithms*, Arlington, VA, January 1994, pp. 167–176.
- [35] K. Jansen and L. Porkolab, “Linear-time approximation schemes for scheduling malleable parallel tasks,” in *ACM-SIAM Symposium on Discrete Algorithms*, Baltimore, MD, January 1999, pp. 490–498.
- [36] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W. H. Freeman and Co., 1979.
- [37] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Berlin: Springer - Verlag, 1999.
- [38] D. G. Luenberger, *Linear and Nonlinear programming*, 2nd ed., Massachusetts: Addison-Wesley, Reading, 1984.
- [39] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: John Wiley and Sons, 1991.
- [40] B. Kalyanasundaram and K. Pruhs, “Speed is as powerful as clairvoyance,” in *IEEE Symposium on Foundations of Computer Science*, Milwaukee, WI, October 1995, pp. 214–221.
- [41] M. Crovella, M. Harchol-Balter, and C. Murta, “Task assignment in a distributed system: Improving performance by unbalancing load,” Boston University, Technical Report 97-018, 1997.
- [42] M. Crovella, R. Frangioso, and M. Harchol-Balter, “Connection scheduling in web servers,” Boston University, Technical Report 99-003, 1999.
- [43] K. R. Baker, *Introduction to Sequencing and Scheduling*. New York: Wiley, 1974.
- [44] “Internet traffic archive,” 2000, <http://ita.ee.lbl.gov>.
- [45] “LOQO Optimization Toolkit,” 2000, <http://www.orfe.princeton.edu/~loqo>.

- [46] “AMPL modeling language for mathematical programming,” 2002, <http://www.ampl.com/>.
- [47] A. S. Tannenbaum, *Computer Networks*, 3rd ed., Englewood Cliffs, NJ: Prentice Hall, 1996.
- [48] I. Chlamtac and S. Kutten, “Tree-based broadcasting in multihop radio networks,” *IEEE Transactions on Computers*, vol. 36, pp. 1209–1223, October 1987.
- [49] R. Ramaswami and K. K. Parhi, “Distributed scheduling of broadcasts in a radio network,” in *IEEE INFOCOM*, Ottawa, Canada, April 1989, pp. 497–504.
- [50] A. Ephremides and T. Truong, “Scheduling broadcasts in multihop radio networks,” *IEEE Transactions on Communications*, vol. 38, pp. 456–460, April 1990.
- [51] I. Chlamtac and A. Lerner, “Fair algorithms for maximal link activation in multihop radio networks,” *IEEE Transactions on Communications*, vol. 35, pp. 739–746, July 1987.
- [52] A. Ephremides, J. E. Wieselthier, and D. J. Baker, “A design concept for reliable mobile radio networks with frequency hopping signalling,” *Proceedings of the IEEE*, vol. 75, pp. 56–73, January 1987.
- [53] B. Hajek and G. Sasaki, “Link scheduling in polynomail time,” *IEEE Transactions on Information Theory*, vol. 34, pp. 910–917, September 1988.
- [54] E. Arıkan, “Some complexity results about packet radio networks,” *IEEE Transactions on Information Theory*, vol. 30, pp. 190–198, July 1984.
- [55] I. Cidon and M. Sidi, “Distributed assignment problems for multihop radio networks,” *IEEE Transactions on Communications*, vol. 38, pp. 1353–1361, October 1989.
- [56] S. Ramanathan, “Scheduling algorithms for multihop radio networks.” Ph.D. dissertation, University of Delaware, Newark, DE, 1992.
- [57] I. Chlamtac and A. Farago, “Making transmission schedules immune to topology changes in multihop packet radio networks,” *IEEE/ACM Transactions on Networking*, vol. 2, pp. 23–29, February 1994.
- [58] R. Ramanathan, “A unified framework and algorithm for (T/F/C)DMA channel assignment in wireless networks,” in *IEEE INFOCOM*, Kobe, Japan, April 1997, pp. 900–907.
- [59] J. Ju and V. Li, “An optimal topology-transparent scheduling method in multihop packet radio networks,” *IEEE/ACM Transactions on Networking*, vol. 6, pp. 298–306, June 1998.
- [60] Z. Tang and J. J. Garcia-Luna-Aceves, “A protocol for topology-dependent transmission scheduling in wireless networks,” in *IEEE WCNC*, New Orleans, LA, September 1999, pp. 1333–1337.

- [61] P. Gupta and P. R. Kumar, “The capacity of wireless networks,” *IEEE Transactions on Information Theory*, vol. 46, pp. 388–404, March 2000.
- [62] R. Rozovsky and P. R. Kumar, “SEEDEx: A MAC protocol for ad hoc networks,” in *ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Long Beach, CA, October 2001, pp. 67–75.
- [63] N. Bambos, “Toward power-sensitive network architectures in wireless communications: Concepts, issues, and design aspects,” *IEEE Personal Communications Magazine*, vol. 5, pp. 50–59, June 1998.
- [64] S. Lu, V. Bharghavan, and R. Srikant, “Fair queuing in wireless packet networks,” in *ACM SIGCOMM*, Cannes, France, September 1997, pp. 63–74.
- [65] G. Bianchi, A. Campbell, and R. Liao, “On utility-fair adaptive services in wireless packet networks,” in *IWQoS*, Napa, CA, May 1998, pp. 256–267.
- [66] D. Eckhardt and P. Steenkiste, “Effort-limited fair (ELF) scheduling for wireless networks,” in *IEEE INFOCOM*, Tel Aviv, Israel, March 2000, pp. 1097–1106.
- [67] N. Vaidya, P. Bahl, and S. Gupta, “Distributed fair scheduling in a wireless LAN,” in *ACM MOBICOM*, Boston, MA, August 2000, pp. 167–178.
- [68] T. Nandagopal, S. Lu, and V. Bharghavan, “A unified architecture for the design and evaluation of wireless fair queueing algorithms,” in *ACM MOBICOM*, Seattle, WA, August 1999, pp. 132–142.
- [69] Y. Lu and R. W. Broderon, “Integrating power control, error correction coding, and scheduling for a CDMA downlink system,” *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 17, pp. 978–989, May 1999.
- [70] T. Nandagopal and X. Gao, “Fair scheduling in wireless data networks,” in *Handbook of Wireless Networks and Mobile Computing*, ed. Ivan Stojmenovic, New York: Wiley, 2001, pp. 171–194.
- [71] D. Karger, C. Stein, and J. Wein, *Handbook of Algorithms and Theory of Computation*. Boca Raton, FL: CRC Press, 1999.
- [72] J. Turek, W. Ludwig, J. L. Wolf, L. Fleischer, P. Tiwari, J. Glasgow, U. Schweigelshohn, and P. S. Yu, “Scheduling malleable and nonmalleable parallel tasks,” in *ACM Symposium on Parallel Algorithms and Architectures*, Cape May, NJ, June 1994, pp. 200–209.
- [73] R. Lepère, D. Trystram, and G. Woeginger, “Approximation algorithms for scheduling malleable tasks under precedence constraints,” in *European Symposium on Algorithms*, Aarhus, Denmark, August 2001, pp. 146–157.
- [74] E. Modiano, “Scheduling packet transmissions in a multi-hop packet switched network based on message length,” in *International Conference on Computer Communications and Networks (ICCCN)*, Las Vegas, NV, September 1997, pp. 350–357.

- [75] E. Modiano, "Scheduling algorithms for message transmission over a satellite broadcast system," in *MILCOM*, Monterey, CA, November 1997, pp. 628–634.
- [76] S. Acharya and S. Muthukrishnan, "Scheduling on-demand broadcasts for heterogeneous workloads: New metrics and algorithms," in *ACM MOBICOM*, Dallas, TX, October 1998, pp. 43–54.
- [77] N. Joshi, S. Kadaba, S. Patel, and G. Sundaram, "Downlink scheduling in CDMA data networks," in *ACM MOBICOM*, Boston, MA, August 2000, pp. 179–190.
- [78] X. Liu, E. K. P. Chong, and N. B. Shroff, "Opportunistic transmission scheduling with resource-sharing constraints in wireless networks," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 19, pp. 2053–2064, October 2001.
- [79] J. W. Lee, R. R. Mazumdar, and N. B. Shroff, "Downlink power allocation for multi-class CDMA wireless networks," to appear in *IEEE INFOCOM*, June 2002.
- [80] D. Tse, "Transmitter Directed, Multiple Receiver System Using Path Diversity to Equitable Maximize Throughput," patent filed, May 1999.
- [81] P. Viswanath, D. Tse, and R. Laroia, "Opportunistic beamforming using dumb antennas," *IEEE Transactions on Information Theory*, to be published.

VITA

Thyagarajan Nandagopal received his B.E. degree in electronics and communication engineering from Anna University, Chennai, India, in 1997. He received his M.S. and Ph.D. degrees in electrical engineering at the University of Illinois at Urbana-Champaign in 2000 and 2002, respectively. He also earned an M.S. degree in applied mathematics at the University of Illinois at Urbana-Champaign in 2002. During his studies, he has worked at AT&T Research Labs and at the Information Technologies and Telecommunication Center at the University of Kansas.

Thyagarajan has been a member of the Phi Kappa Phi honor society since 2000 and has been a member of the IEEE for eight years. He has served as a reviewer for many journals and conferences, and was the recipient of the University Gold Medal for ranking first in undergraduate studies at Anna University.

Thyagarajan's research interests are in the area of wireless networking protocols, scheduling, and network service differentiation for ensuring quality of service. He has six journal publications, eight refereed conference publications, a book chapter, and a pending US patent to his credit. His published work has appeared in prestigious journals (*ACM/Baltzer Wireless Networks Journal*, *Journal of High Speed Networks*, and *IEEE Personal Communications Magazine*) and in reputed conferences (ACM Mobicom, IWQoS, IEEE Infocom, IEEE Globecom, IEEE MoMuC, and WCNC).