Optimal Packet Scheduling in Emerging Wireless Networks

by

Hussein Al-Zubaidy

A Dissertation submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Doctor of Philosophy

Ottawa-Carleton Institute for
Electrical and Computer Engineering

Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, Canada
May 2010

The undersigned recommend to

the Faculty of Graduate Studies and Research

acceptance of the Dissertation

Optimal Packet Scheduling in Emerging Wireless Networks

Submitted by **Hussein Al-Zubaidy**

in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

I. Lambadaris, Supervisor

I. Viniotis

E. Kranakis

A. Miri

C. C. Huang

C. H. Lung

H. M. Schwartz, Department Chair

Carleton University

2010

Abstract

In this work, we follow three different approaches to study the optimal scheduling problem in emerging wireless networks. The results gained from these three approaches complement each other and provide a better understanding for the scheduling function in such networks. In the first part, we model the wireless network by a discrete queuing system that comprises of a set of symmetrical queues (users), served by a set of identical servers (radio channel resources). The queue-server (channel) connectivity is modeled by a sequence of independent and identically distributed (i.i.d.) Bernoulli random variables. At any time slot, a queue may be allocated one or more of the servers that are connected to it during that time slot. We identify a class of policies, namely, the Most Balancing (MB) policies, to be optimal. The optimal policy is the one that minimizes (in a stochastic ordering sense) a set of cost functions of the queue sizes in the system. We provide a theoretical proof of the optimality of MB policies using the *dynamic coupling* method. We also provide simulation results to compare the performance of other work conserving policies to the optimal one under different setups and under some generalizations and relaxations of the mathematical assumptions. The MB policies are shown to dominate all other policies for every setup used.

In the second part of this work, we present an analytic model and a methodology to find the optimal scheduling policy in 3G (third generation) High Speed Downlink Packet Access systems. In this case, the optimal policy is the one that maximizes cell

throughput while maintaining a desired level of fairness between users in that cell. In the proposed model, the arrivals to each queue are assumed to be a sequence of independent Bernoulli random variables; and users' wireless channels are modeled by independent Finite State Markov Channel (FSMC) processes. However, the arrivals, as well as the channel connectivities, for different queues (users) do not have to be identical. We use *stochastic dynamic programming* to model the operations of the downlink scheduler as a Markov Decision Process (MDP) problem. This model is solved numerically for the optimal policy, using value iteration. The optimal policy is shown to have a switch-over structure with respect to the queue lengths. A heuristic approach is developed based on the information gained from studying the optimal policy structure. Simulation is then used to study the performance of the resulted heuristic scheduling policy and to compare it to the optimal one.

The third part of this work focuses on studying the effect of the channel variability on quality of service (QoS) in emerging wireless networks. We develop a server sharing model for 3G wireless systems to evaluate the effect of channel quality on the level of service differentiation that can be achieved. The model is solved to find the server sharing scheme that can achieve a given service differentiation requirement. Although this methodology will result in long-run (static) average server allocation, it has much less computational complexity compared to dynamic allocation methodologies.

To my parents
Amira and Mohammed Al-Zubaidy

Acknowledgments

I would like to express my deep appreciation to Professor Ioannis Lambadaris, my thesis supervisor, for his guidance and support throughout this research. From him I learned to strive for excellence in research, to tackle complex fundamental research problems and to present my work in a concise and clear manner. I also wish to thank Professors I. Viniotis and J. Talim for their time and expertise that they offered throughout parts of this work. Their valuable comments and feedback during our heated discussions helped shape this work to what it is today.

I also would like to thank my family for their continuous support, love and patience. I felt the grace of every prayer they did on my behalf and the embrace of their sincere wishes for my success.

Last but not least, I am thankful to all colleagues and friends who made my stay at Carleton University a memorable and valuable experience.

Table of Contents

List of Tables

List of Figures

Terms and Abbreviations

3G/4G	Third/fourth generation wireless systems.
3GPP	Third generation partnership project.
ARQ	Automatic repeat request.
AMC	Adaptive modulation and coding.
BTS	Base transceiver station.
CC	Chase combining, a form of Hybrid ARQ.
CDMA	Code division multiple access.
CI	Confidence interval.
DP	Dynamic programming.
EVDO	Evolution data only.
FSMC	Finite state Markov channel.
GGSN	Gateway GPRS support node.
GPRS	General packet radio service.
GSM	Global system for mobile communications.

H-ARQ	Hybrid ARQ.
HS-DSCH	High speed downlink shared channel.
HSDPA	High speed downlink packet access.
IP	Internet protocol.
IR	Incremental redundancy.
LB	Least balancing.
LCQ	Longest connected queue.
LCSF	Least connected server first.
LP	Linear programming.
LTE	Long term evolution.
MB	Most balancing.
MC-CDMA	Multi-channel CDMA.
MDP	Markov decision process.
MCS	Modulation and coding scheme.
MCSF	Most connected server first.
PDU	Protocol data unit.
PF	Proportional fair.
QoS	Quality of service.
RFC	Request for comments.

RLC	Radio link control.
RNC	Radio network controller.
RR	Round robin.
SDU	Service data unit.
SGSN	Serving GPRS support node.
SNR	Signal to noise ratio.
TDM	Time division multiplexing.
TFRC	Transport format and resource combination.
TTI	Transmission time interval.
UE	User equipments.
WCDMA	Wide-band CDMA.

Notation

$\mathbb{1}_{\{\Lambda\}}$	The indicator function for condition Λ.
A	Action space.
$\mathbf{a}(\mathbf{s})$	Action taken when the system is in state $\mathbf{s}(t)$.
B	The size of a finite transmission buffer.
B_r	The size of a finite retransmission buffer.
c	The chunk size, number of codes per chunk.
EQ	The average of the total number of packets in the system.
$\mathcal{F}$	The class of cost functions.
$f(\cdot)$	A real function that belongs to $\mathcal{F}$.
$G_{i,j}(t)$	The connectivity of queue i to server j during time slot t.
$g_{i,j}(t)$	A sample value of the random variable $G_{i,j}(t)$.
$H(t)$	State history vector up to time slot t.
$\mathbf{I}_l$	l-dimensional column vector with all entries equal to 1.
I	The set of queues in a system.
$[i]$	Index indicating the i^{th} ordered component in an ordered vector.

$[i]^{\theta}$	The i^{th} ordered component in a vector ordered according to θ.
K	The number of servers in a system.
L	The number of queues in a system.
$\mathcal{M}_i^t$	The set of servers connected to queue i during time slot t.
$M_i(t)$	The cardinality of $\mathcal{M}_i^t$.
$\mathcal{M}^{(n,i)}$	A subset of I (the set of queues) that contains the element $\{i\} \subseteq I$ plus n other elements of I.
m_i	Average share of the server capacity that is assigned to queue i.
$\mathcal{N}$	Channel's state space.
N	The number of channel states in a FSMC model.
$P(E)$	The probability of an event E.
p	The probability that a two-state channel is connected.
P_{sc}	The probability of packet service completion for a scheduled packet.
$P_{ss'}(\mathbf{a})$	State transition probability when starting in state $\mathbf{s}(t)$, taking action $\mathbf{a}(\mathbf{s})$ and end up in state $\mathbf{s}'(t)$.
$P_{\gamma_i,\gamma_i'}$	Channel state transition probability for user i.
ΔP_{γ}	The difference $P(\gamma_1 = 1) - P(\gamma_2 = 1)$ in a two-user system.
ΔP_z	The difference $P(z_1 = u) - P(z_2 = u)$ in a two-user system.
$Q_j(t)$	The index of the queue to be served by server j during time slot t.

$\mathbb{Q}_k$	The set of queues connected to server k during time slot t.
$\mathcal{R}$	The set of real numbers.
$R(\mathbf{s}, \mathbf{a})$	Immediate reward when at state $\mathbf{s}(t)$ and taken the action $\mathbf{a}(\mathbf{s})$.
$s_{[k]^\theta}$	The k^{th} allocated server under server ordering θ during time slot t
$\mathbf{s}(t)$	System state vector at time slot t.
$\mathcal{S}$	System state space.
T	The set of decision time epochs.
Δt	A time interval.
$U(a, b)$	A random variable uniformly distributed over $[a, b]$
u	Arrival batch size.
$V_{i,j}(t)$	A binary variable indicating that server j is both connected and assigned to queue i during time slot t.
$V^\pi(\mathbf{s})$	MDP value function at state $\mathbf{s}(t)$ under the policy π.
$V^*(\mathbf{s})$	The maximal value function.
$W(t)$	The total number of packets in the system during time slot t.
$\mathbf{W}_\pi$	The process of total number of packets in the system under a policy π.
$X_i(t)$	The size of queue i at time slot t.
$x_i(t)$	A sample value of the random variable $X_i(t)$.
$\hat{x}_i(t)$	The size of queue i at time slot t, just before adding the arrivals.

$x_{[l]}$	The l^{th} component in an ordered vector $\mathbf{x}$.
$\mathbf{X}^{MB}$	The process of queue length vector under an MB policy.
$\mathbf{X}^{\pi}$	The process of queue length vector under a policy π.
$\hat{\mathbf{x}}(n,k)$	Queue length vector after allocating the k^{th} server at $t=n$.
$Y_i(t)$	The number of packets withdrawn from queue i during time slot t.
$\mathcal{Y}(\mathbf{x},\mathbf{g})$	The set of feasible withdrawal vectors when in state $(\mathbf{x},\mathbf{g})$.
$\mathcal{Z}$	The set of integer numbers.
$\mathcal{Z}_+$	The set of non-negative integer numbers.
$Z_i(t)$	The number of arrivals to the i^{th} queue during time slot t.
α_i	Arrival rate to queue i.
$\gamma_i(t)$	Channel state for user i during time slot t.
Θ	The set of all possible permutations of the set $\{1,\ldots,K\}$.
$\theta(t)\in\Theta$	An ordering (or permutation) for server allocation during time slot t.
$\kappa_n(\pi)$	Imbalance index at time slot n under policy π.
λ	Discount factor for discounted reward optimality criterion.
$\mu_j(t)$	A Bernoulli RV; service completion for server j during t.
ν_i	Service differentiation rate of user i relative to user 1.
ξ	The probability of successful transmission.
π	A packet scheduling policy.

Π	A set of policies.
Π^{MB}	The set of MB policies.
Π^{LB}	The set of LB policies.
π^{θ}	A policy that is implemented using sequential server allocation following the order $\theta \in \Theta$.
Π_n	The set of policies that have MB property upto time slot n.
Π_n^h	The set of policies that have MB property upto $n-1$ and deviate from MB during time slot n by at most $h > 0$ server allocation.
Π^{WC}	The set of work-conserving policies.
$\vec{\pi}_i$	The FSMC state's distribution for user i's channel.
ρ	Offered load.
σ	Fairness factor.
τ	A time slot $\tau \in \{1, 2, \ldots\}$.
ω	A sample path in the system S under policy π.
$\tilde{\omega}$	A sample path in the system $\tilde{S}$ under policy $\tilde{\pi}$.
$(\Omega, \mathscr{F}, \mathbf{P})$	Sample space.
$\stackrel{\mathscr{D}}{=}$	Equal in distribution.
$\leq_{st}$	Stochastic ordering, i.e., less than or equal to in distribution.
$\prec_p$	The preferred order.

Chapter 1

Introduction

The modern life style in most of the developed countries (and many of the developing ones) is highly dependent on the evolution of technology. Computer networks and telecommunication technologies in particular played a major role in reshaping our communities and the way we work, socialize and communicate with each other. Furthermore, the fast-moving life style we adopted contributed to the demand of reliable, high-speed and portable access networks. One can easily notice the future trend of all wireless access networks connected by a wired backbone.

The emerging wireless networks become increasingly important nowadays as access networks. The ever increasing demand for application versatility and seamless access to the Internet resulted in the evolution of modern wireless networks towards high-speed IP-based packet networks. This trend best manifests itself in emerging third and fourth generation (3G/4G) wireless networks. The third Generation Partnership Project (3GPP) was formed in 1998 to address the need for high-speed IP-based mobile networks. 3GPP is a coalition of a number of telecommunications standards bodies and market representation partners whose goal is to produce globally applicable technical specifications and technical reports for a third generation mobile system (and beyond) based on evolved GSM core networks and the radio access technologies that they support [1].

The higher data rates were achieved by these systems through the deployment of several ground breaking technologies such as fast scheduling, data rate adaptation and hybrid ARQ. Fast scheduling plays a major role in achieving the targeted rates in emerging wireless systems. It also serves as the main tool in quality and service control in these systems. Scheduling in such systems involves the allocation of different types of resources (such as channels, time slots, frequency carriers and antennas) at the same time to maximize spectrum utilization and users' experiences. For example, HSDPA (a 3G wireless network) uses Code and Time Division Multiplexing (CDM/TDM) on the downlink and has 15 codes to be allocated in each two milliseconds Transmission Time Interval (TTI). Scheduling in this system involves not only time slots (i.e., *TTI*) *allocation*, but also *CDMA code allocation*.

1.1 Scheduling in Wireless Networks

In general, scheduling is the process of distributing the available resources between a variety of possible tasks. In wireless networks, a scheduling policy is a rule that decides which user is scheduled for transmission in the next transmission time interval. The decision should optimize some performance metrics, such as throughput and packet delay, subject to some constraints such as users' transmission needs, fairness requirements, channel state, etc.

Scheduling is similar to random access in that both are schemes that allow multiple users to share the same transmission channel. However, they differ in that scheduling has a centralized control to coordinate the allocation of channel resources among all users, an approach that is not possible in random access networks. Although a distributed approach, such as random access, requires only local information (in contrast to a centralized approach), the performance in this case (e.g., throughput) deteriorates rapidly especially in crowded networks.

Rate adaptation is used in emerging wireless networks to adapt the transmission rate to the instantaneous channel conditions. Scheduling may take advantage of multiuser diversity (i.e., the fact that different users may have different channel conditions, and hence different data rates, at a given transmission time interval) to maximize network performance. It does so by scheduling the user with the best channel conditions (highest data rate) to transmit at every transmission time interval resulting in the maximum throughput every time. Maximizing throughput is a desirable feature of scheduling policies, however, there are other important issues that must be taken into consideration. Fairness is one of these issues that might contradict throughput maximization. There is a trade-off between fairness and performance when implementing scheduling in wireless networks. There are several metrics that can be used to represent fairness, for example, proportional fairness, max-min fairness, long-term and short-term fairness [3], [4], [6]. Another issue is the packet delay which has significant importance for users running real-time applications.

In wireless networks, scheduling is usually used for downlink multiuser transmission. Each user in this case has its own data buffer. The scheduling controller chooses packets from these buffers to be transmitted at every transmission time interval. If a reliable feedback channel exists then scheduling can also be used on the uplink. We can classify the scheduling techniques in wireless networks as follows:

1- Opportunistic scheduler (Max C/I): The scheduler allocates the resources to the user with the maximum signal to noise ratio (best channel) at every transmission time interval. This scheme maximizes the network performance (spectrum utilization and throughput) on the expense of fairness. Such scheduling mechanisms are called opportunistic because they take advantage of favorable channel conditions in assigning channel resources to users.

2- Max-min fairness scheduler: The scheduler allocates resources to the user with the minimal received QoS. Such scheme has the advantage of delivering extreme

fairness to all users in the same priority class. However, this scheduler does not take advantage of the multiuser diversity and channel variability and thus results in a reduced network performance compared to the previous scheduler.

3- Proportional fair scheduler (PF): The previous two types of schedulers represent the two extremes of the fairness and performance trade-off. PF scheduler provides a middle ground between these two types. The basic concept behind proportional fairness is to weight the scheduling priorities by both channel conditions and transmission history.

4- Mixed strategy scheduler: This type of schedulers is proposed to accommodate different types of users who use services that have different QoS requirements. An example is a scheduling scheme in a system that supports a mixture of real-time and non-real-time users.

More information regarding scheduling in wireless networks and the above scheduler types can be found in [2], [5], [7] and [8].

1.2 Techniques for Optimal Control

This work lies in the broader field of optimal control of queuing systems. In the literature, there are three major techniques that were used extensively to study optimal control of queuing systems; Dynamic Programming (DP), Linear Programming (LP) and Stochastic Dominance. Dynamic Programming is the traditionally favored technique for optimization [9], [10], [11], [12]. In this technique, a cost function is defined as a function of the system state and the sequence of decisions made by the working policy. Then, using Value Iteration or Policy Iteration, one can identify the policy that minimizes the cost function,(or at least some of its properties). When the underlying random process resulted from this control problem is a Markov Decision Process (MDP), then a feasible solution to the formulated DP equation is possible

most of the time. However, when the DP equation becomes somewhat complicated, then a solution may be hard to find. A similar statement is true when the underlying process is a Semi-Markov Decision Process. The DP equation can also be solved numerically. However, when the state space and/or action space grow larger, then the computational complexity will rapidly increase and may become a prohibiting factor. For more details on Dynamic Programming, the reader may consult [13], [14], [15].

Linear Programming is another technique that can be used to solve optimization problems in queuing systems. LP concerns the problem of maximizing (or minimizing) a linear functional over a polyhedron [16]. In this technique, a linear functional, representing the objective or cost, is maximized (or minimized) over a set of constraints (linear system of inequalities). The general problem is to determine $\max\{cx|Ax \leq b\}$, where c, x, b are vectors and A is a matrix; it is assumed that all quantities have compatible sizes. Integer Linear Programming is yet another optimization technique that investigates LP problems where the variables are restricted to integer values. The problem in this case has the form $\max\{cx|Ax \leq b; x \in \mathcal{Z}\}$, where $\mathcal{Z}$ is the set of all integers. LP is used in the literature to describe and solve Markov Decision Processes and to find the optimal dynamic control policies in systems that can be modeled by these processes [17], [18], [19]. In this work, we did not use LP or ILP in our treatments.

Stochastic Dominance is another important technique in stochastic modeling, analysis and optimization [20]. It refers to the ordering of probability distributions, i.e., the case when one probability distribution can be ranked as "superior" to another. This technique provides tools that can be used in ordering random variables and random processes in distribution. The ability to do this has apparent significance in optimal stochastic control. The underlying processes resulted from applying different controls can be ordered stochastically (in distribution) and the optimal control will be the one that "dominates" all the other ones. However, proving such ordering

can be tricky sometimes. A dynamic coupling method [21] is one way to do that. In this method, the underlying random process can be coupled with a realization of another process on the same probability space using results from stochastic dominance [20], [14]; we will present some of these results later. Then we can order the coupled processes almost surely (i.e., with probability 1), by comparing these processes for all sample values at all times. Coupling is used in the literature to prove optimality in many scheduling and routing problems, c.f. [22], [23], [24].

This work is divided into three parts that are complimentary to each other. All three parts tackle the problem of packet scheduling in emerging wireless networks. They cover the theoretical, performance and implementation aspects of the optimal scheduling policy in such networks. In the following, we will briefly introduce, provide motivation and list the contributions of the research done in each one of the three parts. Full details and results will be reported in the subsequent chapters of the thesis.

1.3 Part I: Optimal Control in Emerging Wireless Networks

This part deals with the theoretical treatment of the scheduling problem in the emerging wireless networks we are studying in this research. We propose a queuing model that covers the packet scheduling function in many of these networks. We identify (and provide mathematical characterization for) a class of optimal policies, the *Most Balancing* (MB) policies. Then we provide theoretical proof of their optimality using dynamic coupling method and stochastic dominance. We also provide simulation results to study the performance of these policies and compare it to that of several other work-conserving policies. The findings of this part of the work are presented in

Chapters 2 and 3.

1.3.1 Motivation

The optimal policy characterization and the proof of its optimality that we present in this part play a major role in understanding packet scheduling in multi-server wireless networks with independent random connectivity. Our intuition suggests that the Most Balancing policies described in this work can be shown (in future work) to be optimal for a wide range of scheduling and routing problems in wireless systems. Therefore, this work not only provides a proof for the optimality of MB policies in the system under investigation, but it also opens the door for a wide set of important research problems for future investigation. This work provides the characterization of the class of MB policies and a methodology to prove their optimality theoretically.

The model we present in this part of the work can be applied to the scheduling problems in the 3G/4G wireless networks. The results obtained provide insight and better understanding of the significant role of the packet scheduler in such networks. The work also provide a practically implementable algorithm for optimal packet scheduling in these networks. These results can be used by scheduler designers as guidelines for designing scheduling policies for more complex systems (e.g., differentiated services for QoS scheduling, priority scheduling, etc.).

Yet another motivation for this work is the significance of this model from the theoretical (and academic research) point of view. It provides a contribution to the field of stochastic optimization and optimal control and has applicability in areas other than wireless network modeling, such as resource management, admission control and other problems in the area of operational research.

1.3.2 List of Contributions

The main contributions presented in this part can be summarized by the following:

1. We provide a formulation for the optimal server allocation problem in multi-server, homogeneous wireless systems with independent connectivity.

2. We introduce the class of Most Balancing (MB) policies and prove theoretically (using stochastic dominance and dynamic coupling method) that they are optimal.

3. Using the same methodology, we also prove that MB policies are optimal in a similar system with the ability to retransmit packets when their service is not completed.

4. We suggest an implementation algorithm for an MB policy.

5. We also propose the Least Connected Server First/Longest Connected Queue (LCSF/LCQ) policy as a low-overhead approximation of MB policies and provide an easily programmable algorithm for constructing the LCSF/LCQ policy.

6. We compare the performance of several policies, including LCSF/LCQ via simulations.

1.4 Part II: Optimal Scheduling in HSDPA Networks: Dynamic Programming Approach

In this part, we study the downlink scheduler in the 3G High Speed Downlink Packet Access networks. We provide a model and a methodology (based on Dynamic Programming) to calculate the optimal packet scheduling policy in such networks. The optimal policy we investigate is the one that maximizes overall system throughput

for a given fairness criterion. We study the structure of this policy and we provide a heuristic policy based on the information we collect from the structure and behavior of the optimal policy. Simulation results are also presented in this part. These results provide performance evaluation and comparison of the optimal, heuristic and Round Robin packet scheduling policies. The findings of this part are presented in Chapter 4.

1.4.1 Motivation

The HSDPA system is a typical example of the emerging wireless networks. It is high-speed, packet-based and all-IP network. It utilizes most of the technologies used in 3G and 4G wireless systems, such as hybrid ARQ, adaptive modulation and coding and fast scheduling. The downlink scheduler in this system is of a particular interest to our work, since it plays a major role in achieving the high data rates in an HSDPA system. The model in Part I provided an intuition to the structure of the optimal policy in such systems. However, Part I is concerned with symmetrical (homogeneous) systems and the direct application of the MB policies in non-homogeneous systems may not be practical. Therefore, we develop a new formulation for this problem (using dynamic programming) to study the effect of asymmetry on the optimal policy structure. We also develop a near-optimal, heuristic policy that can be applied to real systems. Because of the resemblance (of technologies used) of the HSDPA system to the most recent 3GPP releases, such as Long Term Evolution (LTE), the presented approach and results can be easily adapted to these systems.

Until the publication of this work, most of the available work in scheduler design in HSDPA system was based on the intuition and creativity of the designers. The designer usually selects an optimization criterion that represents some important performance measure (in his/her opinion), builds an algorithm based on that criterion and then tries to establish confidence in that algorithm using backward analysis or

simulation. Most likely, this will result in a suboptimal algorithm at best, that performs well in some scenarios and poorly in others. The new features used in HSDPA system (and later releases) introduced many new and interrelated tuning parameters which cannot be grasped by a single selected optimality criterion. This motivated the pursuance of a declarative approach by building a model that captures the effects of all these entwined parameters. Another observation is the lack of work on schedulers that dynamically allocate codes as well as time slots to the users in an HSDPA system.

1.4.2 List of Contributions

The contributions of the work presented in this part can be summarized by the following:

1. We provide an analytical approach to model the downlink scheduler in a 3G HSDPA system. This approach can be extended to other 3G/4G wireless systems.

2. Using the theory of Dynamic Programming, we present an optimization framework for the determination of the optimal policy for the HSDPA downlink packet scheduler.

3. A near-optimal, *heuristic* policy is proposed, based on the structural properties of the optimal policy and its dependence on changing system parameters.

4. We conduct a simulation study to quantify the effect of different model parameters on the behavior of the optimal policy. We also study the performance of the proposed heuristic policy and compare it to that of the optimal policy.

1.5 Part III: Analytic Evaluation of Downlink Service Rate in 3G Wireless Networks

The effect of the data rate adaptation to channel quality on system performance (e.g., throughput and fairness) in 3G wireless networks is studied in this part. We provide a stochastic analytical model for such systems. This model is used to evaluate the achievable average data rate in 3G wireless networks. The wireless channel is modeled by a Finite State Markov Channel (FSMC). This model results in a system of equations that are solved analytically. We conduct a numerical evaluation of the resulted solution (e.g., relations between the throughput and different parameters in the system) for different cases and we plot these results. This model helps researchers understand the relation between the various system parameters and the maximum achievable throughput and fairness limitations in systems with rate adaptation. This model does not provide a dynamic scheduling policy as the previous models; instead, it provides a static evaluation of the achievable data rate per user *on average*, regardless of the dynamic policy used. This information is crucial for providing Quality of Service to different users with different channel qualities. The results obtained show that the QoS that can be provided to a user depends on its channel quality. Therefore, this model can be used as a guideline in conjunction with the dynamic packet scheduling schemes to provide a feasible QoS scheduling in 3G wireless systems. The findings are presented in Chapter 5.

1.5.1 Motivation

This part of the research was conducted initially to understand the complexity of scheduling in wireless systems in general and in 3G wireless systems in particular. The methodology presented in this work provides a mathematical model that can

be computed easily (has low computational complexity) to answer some important questions about the achievable service rates in these systems and the relative quality of service that can be achieved for given channel conditions. This part uses mean value analysis and will result in static allocation of resources (e.g., TTIs and codes), in contrast to Parts I and II, which provide dynamic allocations of these resources. Nevertheless, the results obtained in this part can be used in addition to those obtained in the previous parts to provide long-run fairness or service differentiation (i.e., QoS scheduling) in many emerging wireless networks.

1.5.2 List of Contributions

The main contributions of the work presented in this part are summarized by the following:

1. We use stochastic modeling to provide a server sharing model for the downlink scheduler in 3G wireless networks.

2. We derive a relationship between the service rate achieved by each user in the system and the server share allocated to that user.

3. The model is solved to determine the achievable average service rate of a user as a function of the channel state and the server share scheduled to that user.

4. We present some numerical results for different scheduling schemes such as fair scheduler, equal shares scheduler and differentiated services.

1.6 Contributions to the Literature

The contributions to the literature that resulted from this work are listed below:

1. H. Al-Zubaidy et al., Optimal Resource Scheduling in Wireless Multi-service Systems with Random Channel Connectivity, IEEE Globecom09, Honolulu, HI, Dec 2009.

2. H. Al-Zubaidy et al., Optimal Multi-Server Allocation to Parallel Queues With Random Connectivity and Retransmissions, to appear in IEEE ICC'2010, Cape town, South Africa, May 2010.

3. H. Al-Zubaidy et al., Optimal Scheduling in High Speed Downlink Packet Access Networks, to appear in ACM Transactions on Modeling and Computer Simulation (TOMACS) Journal.

4. H. Al-Zubaidy et al., Code Allocation Policy Optimization in HSDPA Networks Using FSMC Channel Model, IEEE Wireless Communications and Networking Conference (WCNC08), Las Vegas, March 2008.

5. H. Al-Zubaidy et al., Dynamic Scheduling in High Speed Downlink Packet Access Networks: Heuristic Approach, International Conference on Military Communications (MILCOM07), Orlando, USA, Oct 2007.

6. H. Al-Zubaidy et al., Determination of Optimal Policy for Code Allocation in High Speed Downlink Packet Access with Multi-State Channel Model, ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2007), Chania, Crete Island, Greece, Oct 2007.

7. H. Al-Zubaidy et al., Optimal Scheduling Policy Determination for High Speed Downlink Packet Access, IEEE International Conference on Communications (ICC) 2007, Glasgow, Scotland, June 2007.

8. H. Al-Zubaidy et al., Downlink Scheduler Optimization in High-Speed Downlink Packet Access Networks, poster session, The 26th Annual IEEE Conference on Computer Communications INFOCOM 2007, Anchorage, Alaska, USA, May 2007.

9. H. Al-Zubaidy et al., Heuristic Approach of Optimal Code Allocation in High Speed Downlink Packet Access Networks, The Sixth International Conference on

Networking ICN07, Martinique, French Caribbean, April 2007.

10. H. Al-Zubaidy et al., Analytic Evaluation of Achievable Downlink Service Rate and Server Sharing in 3G Wireless Networks, The International Conference on Information and Communication Technologies (ICTTA08), Syria, April 2008.

11. H. Al-Zubaidy et al., Service Rate Determination for Group of Users with Random Connectivity Sharing A Single Wireless Link, Seventh IASTED International Conferences on Wireless and Optical Communications (WOC 2007), Montreal, May 2007.

Chapter 2

Optimal Control in Emerging Wireless Networks

In this chapter, we investigate an optimal scheduling problem in a discrete-time system of L parallel queues that are served by K identical servers. This model has been widely used in studies of emerging 3G/4G wireless systems. We introduce the class of Most Balancing (MB) policies and provide their mathematical characterization. We prove that MB policies are optimal among all work-conserving policies; we define optimality as minimization, in *stochastic ordering sense*, of a range of cost functions of the queue lengths, including the process of total number of packets in the system. We use dynamic coupling arguments for our proof. We provide am algorithm to implement an MB policy. During each time slot, the server-queue (channel) connectivities as well as arrivals to each queue, are modeled by independent Bernoulli random variables. The arrivals to individual queues are assumed to be symmetrical and independent across the queues in the system and independent of the connectivities. We conduct a simulation study to compare the performance of several work-conserving policies to that of the optimal one. In the simulations, we study different scenarios with different packet arrival models including batch and correlated Bernoulli arrival processes.

2.1 Introduction and Model Description

Emerging 3G/4G wireless networks can be categorized as high-speed, IP-based packet access networks. They utilize the channel variability, using data rate adaptation, and user diversity to increase their channel capacity. These systems usually use a mixture of Time and Code Division Multiple Access (TDMA/CDMA). Time is divided into equal size slots, each of which can be allocated to one or more users. To optimize the use of the enhanced data rate, these systems allow several users to share the wireless channel simultaneously, using CDMA. This will minimize the wasted capacity resulted from the allocation of the whole channel capacity to one user at a time even when that user is unable to utilize all of that capacity. Another reason for sharing system capacity between several users, at the same time slot, is that some of the user equipments at the receiving side might have design limitations on the amount of data they can receive and process at a given time.

The connectivity of users to the base station in any wireless system is varying with time and can be best modeled as a random process. In the following subsection, we provide a more formal model description and motivation for the problem at hand.

2.1.1 Model Description

In this work, we assume that time is slotted into equal-length deterministic intervals. We model the wireless system under investigation as a set of L parallel, symmetrical queues with infinite capacity (see Figure 2.1); the queues correspond to the different users in the system. The queues share a set of K identical servers, each server representing transmission channels (or any other network resource, e.g., power, CDMA codes, etc.). We make no assumption regarding the number of servers relative to the number of queues, i.e., K can be less than, equal to or greater than L. The packets in this system are assumed to have constant length, and require one time slot to

complete service. A server can serve one packet only at any given time slot. A server can only serve connected, non-empty queues. Therefore, the system can serve up to K packets during each time slot. Those packets may belong to one or several queues.

The connectivity between a user and a channel is random. The state of the channel connecting the i^{th} queue to the j^{th} server during the n^{th} time slot is denoted by $G_{i,j}(n)$ and can be either connected ($G_{i,j}(n) = 1$) or not connected ($G_{i,j}(n) = 0$). Hence, $G_{i,j}(n)$ will determine (in a real system) if a transmission channel j can be used by user i or not.

The number of arrivals to the i^{th} queue during time slot n is denoted by $Z_i(n)$. We require that $Z_i(n)$ have the same Bernoulli distribution for all i and be independent of $G_{i,j}(n)$ for all $i = 1, 2, \ldots, L$, $j = 1, 2, \ldots, K$ and n. This means that $Z_i(n)$ could be any sequence of integer-valued random variables, as long as they have symmetrical distribution across the queues and are independent of the connectivities. We define $X_i(n)$ to represent the number of packets in the i^{th} queue at the beginning of time slot n.

The symmetry of the system will allow us to use coupling arguments to prove the optimality of the MB policies. If this condition is violated, then it will not be possible to use coupling argument any more, since the distribution of the arrival process and the connectivity process will not be permutation-invariant. In this case for example, if we swap the arrival variables of two queues in the system at a given time slot, as we usually do in a coupling argument, the resulted distribution of the permuted system, will be different than the original one. Therefore, the comparison will be invalid and the argument fails.

A scheduler (or server allocation or scheduling policy) decides, at the beginning of each time slot, what servers will be assigned to which queue during that time slot.

The objective of this work is to identify and study the optimal scheduling policy that minimizes, in a stochastic ordering sense, a range of cost functions of the system

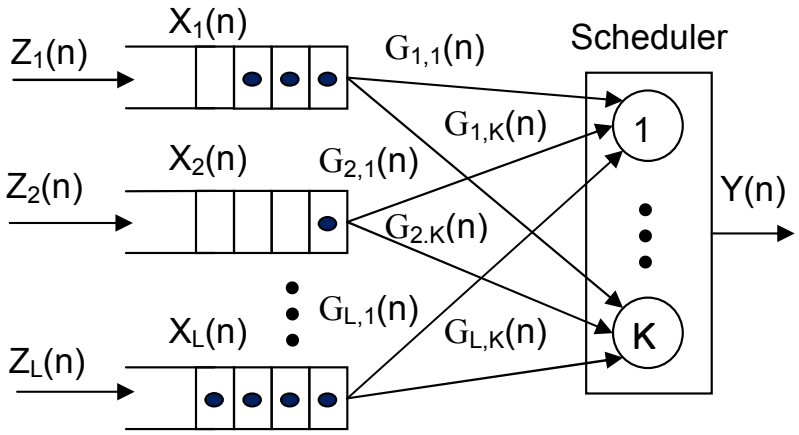


Figure 2.1: Abstraction of downlink scheduler in a 3G wireless network.

queue occupancies, including the total number of queued packets, in the aforementioned system. The choice of the class of cost functions and the minimization process will be discussed later.

2.2 Previous Work and Our Contributions

In the literature, there is substantial research effort focusing on the optimal server allocation in wireless networks. Tassiulas and Ephremides [23] for example, tackled a similar problem where a single server (i.e., $L = 1$) can only be allocated to one user and can only serve one packet at each time slot. They proved, using a coupling argument, that LCQ (Longest Connected Queue) is optimal. In our work, we show that LCQ is not always optimal in a multi-server system since servers can be assigned to one or more queues simultaneously. Bambos and Michailidis [25] worked on a similar model (a continuous time version of [23] with finite buffer capacity) and found that, under stationary ergodic input job flow and modulation processes, both MCW (Maximum Connected Workload) and LCQ dynamic allocation policies maximize the stability region for this system. Furthermore, they proved that C-FES, a policy that allocates the server to the connected queue with the fewest empty spaces,

stochastically minimizes the loss flow and maximizes the throughput [26].

Another relevant result is that reported by Ganti Modiano and Tsitsiklis [24]. They presented a model for a satellite node that has K transmitters. The system was modeled by a set of parallel queues with symmetrical statistics competing for K identical servers. At each time slot, no more than one server is allocated to each scheduled queue. They proved, using stochastic coupling arguments, that LCQ, a policy that allocates the K servers to the K longest connected queues at each time slot, is optimal. This model is similar to the one we consider in this work, except that in our model one or more servers can be allocated to each queue in the system. A further, stronger difference between the two models is that we consider the case where each queue has *independent connectivities* to different servers. We make these assumptions for a more suitable representation of the 3G wireless systems described earlier. These differences make it substantially harder to identify (and even describe) the optimal policy (see Section 2.4). A more recent result that has relevance to our work is the one reported by Kittipiyakul and Javidi in [27]. They proved, using dynamic programming, that a maximum-throughput and load-balancing (MTLB) policy minimizes the expected average cost for a two-queue, multi-server system. In our research work we proved the optimality of the most balancing policies in the more general problem of a multi-queue (more than two queues) and multi-server system with random channel connectivity. A stronger distinction of our work is that we proved the optimality in a stochastic ordering sense which is a stronger notion of optimality compared to the expected average cost criterion that was used in [27]. Lott and Teneketzis [28] tackled a multi-class system of N weighted-cost, parallel queues and M servers. They also used the one server per queue restriction used in [24]. They showed that an index rule is optimal and provided conditions sufficient, but not necessary, to guarantee its optimality.

Koole et al [29] studied a model similar to that of [23] and [26]. They found that

the Best User (BU) policy maximizes the expected discounted number of successful transmissions. Liu et al [30], [31] studied the optimality of opportunistic schedulers (e.g., Proportional Fair (PF) scheduler). They presented the characteristics and optimality conditions for such schedulers. However, Andrews [32] showed that there are six different implementation algorithms of PF scheduler, none of which is stable. For more information on resource allocation and optimization in wireless networks the reader may consult [33], [34], [35], [36], [37], [38], [39] and [40].

In summary, our main contributions in this chapter are the following:

- We introduce the class of Most Balancing (MB) policies for server allocation in the model of Figure 2.1 and prove their optimality for minimizing, in stochastic ordering sense, a set of functions of the queue lengths (e.g., total system occupancy).

- An MB policy attempts to balance all queue sizes at every time slot, so that the total sum of queue size differences will be minimized. Such a policy exists and may be determined through a finite search of all possible server allocations. In our work, we present a method that reduces this search by searching for policies that assign servers to the "longest connected queue" (LCQ allocation) in an ordered manner.

- We provide a heuristic approximation for an MB policy. At any time slot, such policies allocate the "least connected servers first" to their "longest connected queues" (LCSF/LCQ). These policies require reduced complexity, $O(L \times K)$ for their implementation. We further show, using simulation, that their performance (on average) is virtually indistinguishable (falls within the margin of the confidence interval) of that achieved by MB policies.

The rest of the chapter is organized as follows. In section 2.3, we introduce notation and define the server allocation policies. In section 2.4, we introduce and

provide a detailed description of the MB server allocation policies. We also present an implementation algorithm for such policies. In section 2.5, we provide some theoretical background that is necessary for the upcoming proof. In section 2.6, we present the main result, i.e., the optimality of MB policies. In section 2.7, we present the Least Balancing (LB) policies, and show that these policies perform the worst among all work-conserving policies. MB and LB policies provide upper and lower performance bounds. In section 2.8, we introduce a practical, low-overhead approximation for such policies, namely the LCSF/LCQ policy and the MCSF/SCQ policy, with their implementation algorithms. In section 2.9, we present simulation results for varous scheduling policies. We present proofs for some of our results in Appendix A – C.

2.3 Policies for Server Allocation

Recall that L and K denote the number of queues and servers respectively in the model introduced in Figure 2.1. We will use **bold face**, UPPER CASE and lower case letters to represent vector/matrix quantities, random variables and sample values respectively. In order to represent the policy action that corresponds to "idling" a server, we introduce a special, "dummy" queue which is denoted as queue 0. Allocating a server to this queue is equivalent to idling that server. By default, queue 0 is permanently connected to all servers, contains only "dummy" packets. Let $\mathbb{1}_{\{A\}}$ denote the indicator function for condition A. Throughout this paper, we will use the following notation:

- $\mathbf{G}(n)$ is an $(L+1) \times K$ matrix, where $G_{i,j}(n)$ for $i > 0$ is the channel connectivity random variable as defined in Section 2.1.1. We assume that $G_{0,j}(n) = 1$ for all j, n.

- $\mathbf{X}(n) = (X_0(n), X_1(n), X_2(n), \ldots, X_L(n))^T$ is the vector of queue lengths at the

beginning of time slot n, measured in number of packets. We assume $X_0(1) = 0$.

- $\mathbf{Q}(n) = (Q_1(n), \ldots, Q_K(n))^T$ is the server allocation control vector. $Q_j(n) \in \{0, 1, \ldots, L\}$ denotes the index of the queue that is selected (according to some rule) to be served by server j during time slot n. Note that serving the "dummy" queue, i.e., setting $Q_j(n) = 0$ means that server j is idling during time slot n.

- $\mathbf{V}(n)$ is a $(L + 1) \times K$ matrix such that $V_{i,j}(n) = \mathbb{1}_{\{i=Q_j(n)\}} \cdot G_{i,j}(n), i = 0, \ldots, L$ and $j = 1, \ldots, K$. Hence $V_{i,j}(n)$ will be equal to 1 if and only if server j is both connected to queue i and assigned to serve it.

- $\mathbf{Y}(n) = (Y_0(n), Y_1(n), Y_2(n), \ldots, Y_L(n))^T$ is the vector of the number of packets withdrawn from the system during time slot n. For any i, $Y_i(n) \in \{0, 1, \ldots, K\}$ denotes the number of packets withdrawn from queue i (and assigned to servers) during time slot n.

- $\mathbf{Z}(n) = (Z_0(n), Z_1(n), Z_2(n), \ldots, Z_L(n))^T$ is the (column) vector of the number of exogenous arrivals during time slot $n = 1, 2, \ldots$. Arrivals to queue $i \neq 0$ are as defined in Section 2.1.1. The number of arrivals at the dummy queue is defined as $Z_0(n) = Y_0(n)$. This will insure that $X_0(n) = 0, \forall n$.

- The tuple $(\mathbf{X}(n), \mathbf{G}(n))$ denotes the "state" of the system at the beginning of time slot n.

For future reference, we will call $\mathbf{Q}(n)$ the *scheduling (or server allocation) control* and $\mathbf{Y}(n)$ the *withdrawal control*. The matrix $\mathbf{V}(n)$ will be useful in describing feasibility constraints on such controls (see Equations (2.2) and (2.3)).

2.3.1 Feasible Scheduling and Withdrawal Controls

Using the previous notation and given a scheduling control vector $\mathbf{Q}(n)$ we can compute the withdrawal control vector as:

$$Y_i(n) = \sum_{j=1}^{K} \mathbb{1}_{\{i=Q_j(n)\}}, \quad i = 0, 1, 2, \ldots, L. \tag{2.1}$$

We assume that the controller has complete knowledge of the system state information at the beginning of each time slot. Then we say that a given vector $\mathbf{Q}(n) \in \{0, 1, \ldots, L\}^K$ is a *feasible* scheduling control (during time slot n) if: (a) a server is allocated to one connected queue, and, (b) the number of servers allocated to a queue cannot exceed the size of the queue at time n. Mathematically, these conditions are captured by the following (necessary and sufficient) constraints:

$$\mathbf{V}^T(n) \cdot \mathbf{I}_{L+1} = \mathbf{I}_K \tag{2.2}$$

$$\mathbf{V}^*(n) \cdot \mathbf{I}_K \leq \mathbf{X}(n) \tag{2.3}$$

where $\mathbf{I}_l$ is a column vector of size l, with all entries equal to one, and

$$V^*_{i,j}(n) = \begin{cases} 0, & i = 0; \\ V_{i,j}(n), & \text{otherwise.} \end{cases}$$

The K constraints in Equation (2.2) capture condition (a) above; indeed, equality in Equation (2.2) is not possible if a server j is allocated to a non-connected queue, since $V_{i,j}(n) = 0$ for all i in this case. The point-wise inequality in Inequality (2.3) captures condition (b); with the choice of $\mathbf{V}^*(n)$ we guarantee that Inequality (2.3) is satisfied for the dummy queue. Note that allocating more than one server to a queue is feasible.

Similarly, we say that a given vector $\mathbf{Y}(n) \in \{0, 1, \ldots, K\}^{L+1}$ is a *feasible* withdrawal control (during time slot n) if there is a matrix $\mathbf{V}(n)$ that satisfies the feasibility constraints (2.2) and (2.3) such that

$$\mathbf{Y}(n) = \mathbf{V}(n) \cdot \mathbf{I}_K \tag{2.4}$$

From constraints (2.2), it is clear that a server can be allocated to one and only one connected queue; summing the constraints, we can see that the controller can only withdraw a total of up to K packets from the connected nonempty queues in the system. For any feasible $\mathbf{Y}(n)$, from the definition of $V_{i,j}(n)$ and Inequality (2.3) it follows that, at any time slot, the number of packets withdrawn from any queue cannot be larger than the size of the queue or larger than the total number of servers connected to the queue. Therefore, a feasible withdrawal control $\mathbf{Y}(n)$ satisfies the (necessary) conditions

$$0 \leq Y_i(n) \leq \min\left(X_i(n), \sum_{j=1}^{K} G_{i,j}(n)\right), \quad \forall n, i \neq 0, \tag{2.5}$$

$$\sum_{i=0}^{L} Y_i(n) = K, \qquad \forall n. \tag{2.6}$$

It is clear that conditions (2.5) and (2.6) are not sufficient for the feasibility of the withdrawal vector $\mathbf{Y}(n)$. For future reference, we denote the set of all feasible withdrawal controls while in state $(\mathbf{x}, \mathbf{g})$ by $\mathcal{Y}(\mathbf{x}, \mathbf{g})$.

Note from Equation (2.1) that, given a feasible scheduling control $\mathbf{Q}(n)$, the withdrawal control $\mathbf{Y}(n)$ is determined uniquely and is feasible (Equations (2.4) and (2.1) are the same, for a feasible scheduling control). However, for a given system state Equation (2.4) may have more than one solution, i.e., $\mathbf{V}(n)$ (and hence $\mathbf{Q}(n)$), that

satisfy Equations (2.2) and (2.3). The feasibilities of withdrawal control and scheduling control are entwined and by definition imply each other. Nevertheless, deriving $\mathbf{Q}(n)$ from a feasible withdrawal control is not straightforward. One way to do that is to devise an algorithm that searches through all possible scheduling vectors to find one that satisfies Equation (2.4).

For the rest of this chapter, we will refer to $\mathbf{q}(n)$ as an *implementation* of the given feasible control $\mathbf{y}(n)$.

2.3.2 Definition of Policies for Server Allocation

For any feasible control ($\mathbf{Y}(n)$), the system described earlier evolves according to

$$\mathbf{X}(n+1) = \mathbf{X}(n) - \mathbf{Y}(n) + \mathbf{Z}(n), \qquad n = 1, 2, \ldots \tag{2.7}$$

We assume that arrivals during time slot n can only be added after removing served packets. Therefore, packets that arrive during time slot n have no effect on the controller decision at that time slot and may only be withdrawn during $t = n+1$ or later.

A *packet scheduling policy* π (or policy π for simplicity) is a rule that determines feasible withdrawal vectors $\mathbf{Y}(n)$ for all n, as a function of the past history and current state of the system $\mathbf{H}(n)$. The state history is given by the sequence of random variables

$$\begin{aligned} \mathbf{H}(1) &= (\mathbf{X}(1)), \quad \text{and} \\ \mathbf{H}(n) &= (\mathbf{X}(1), \mathbf{G}(1), \mathbf{Z}(1), \ldots, \mathbf{G}(n-1), \mathbf{Z}(n-1), \mathbf{G}(n)), \\ &\qquad n = 2, 3, \ldots \end{aligned} \tag{2.8}$$

Let $\mathcal{H}_n$ be the set of all histories up to time slot n. Then a policy π can be formally

defined as the sequence of measurable functions

$$\begin{aligned} & u_n : \mathcal{H}_n \longmapsto \mathcal{Z}_+^{L+1}, \\ \text{s.t.} \quad & u_n(\mathbf{H}(n)) \in \mathcal{Y}(\mathbf{X}(n), \mathbf{G}(n)), \quad n = 1, 2, \ldots \end{aligned} \tag{2.9}$$

where $\mathcal{Z}_+$ is the set of non-negative integers and $\mathcal{Z}_+^{L+1} = \mathcal{Z}_+ \times \cdots \times \mathcal{Z}_+$, where the Cartesian product is taken $L+1$ times.

At each time slot, the following sequence of events happens: first, the connectivities $\mathbf{G}(n)$ and the queue lengths $\mathbf{X}(n)$ are observed. Second, the packet withdrawals $\mathbf{Y}(n)$ are determined according to the policy in effect. Finally, the new arrivals $\mathbf{Z}(n)$ are added to determine the next queue length vector $\mathbf{X}(n+1)$.

We denote by Π the set of all policies described by Equation (2.9). We will show in Section 2.6.4 that a special subset, namely the set of *Most Balancing* (MB) policies we introduce next are optimal: they minimize (in the stochastic ordering sense) a range of cost functions including the process of total number of packets in the system.

2.4 The Class of MB Policies

In this section, we provide a formal description and mathematical characterization of the class of MB policies.

Intuitively, the MB policies "attempt to balance the lengths of all queues in the system *as much as possible*, at every time slot n"; they do so by choosing a control $(\mathbf{y}(n) \in \mathcal{Y}(\mathbf{x}, \mathbf{g}))$ that minimizes the total difference between the queues in the system. This will hopefully result in the "smallest possible differences in the lengths of the longest queues in the system" (i.e., will achieve a most balancing effect).

For a more formal definition of MB policies, we first define the following:

Given a state $(\mathbf{x}(n), \mathbf{g}(n))$ and a policy π that chooses the feasible control $\mathbf{y}(n)$

at time slot n, define $\hat{x}_i(n) = x_i(n) - y_i(n)$ as the size of queue $i, i = 1, \ldots, L$, after applying the control $y_i(n)$ and just before adding the arrivals during time slot n. For notational simplicity we also define $\hat{x}_0(n) = 0$. Furthermore, we define the *"imbalance index"*, $\kappa_n(\pi)$, as the total sum of differences of the $L+1$-dimensional vector $\hat{x}(n)$ under the policy π at time slot n (where π takes the control $\mathbf{y}(n) \in \mathcal{Y}(\mathbf{x}, \mathbf{g})$ at time slot n), i.e.,

$$\kappa_n(\pi) = \sum_{i=1}^{L+1} \sum_{j=i+1}^{L+1} (\hat{x}_{[i]}(n) - \hat{x}_{[j]}(n)) \tag{2.10}$$

where $[k]$ denotes the index of the k^{th} longest queue after applying the control $\mathbf{y}(n)$ and before adding the arrivals at time slot n. By convention, queue '0' (the "dummy queue") will always have order $L+1$ (i.e., the queue with the minimum length). It follows from Equation (2.10) that the minimum possible imbalance index is $L \cdot \hat{x}_{[L]}$ (i.e., all L queues have the same length which is equal to the shortest queue length) which is indicative of a fully balanced system. Let Π^{MB} denote the set of all MB policies, then we define the elements of this set as follows:

Definition: A *Most Balancing* (MB) policy is a policy $\pi \in \Pi^{MB}$ that, at $n = 1, 2, \ldots$, chooses feasible withdrawal vectors $\mathbf{y}(n) \in \mathcal{Y}(\mathbf{x}, \mathbf{g})$ such that the imbalance index is minimized at every n, i.e.,

$$\Pi^{MB} = \left\{ \pi : \underset{\mathbf{y}(n) \in \mathcal{Y}(\mathbf{x}, \mathbf{g})}{\operatorname{argmin}} \kappa_n(\pi), \quad \forall n \right\} \tag{2.11}$$

The set Π^{MB} in Equation (2.11) is well-defined and non-empty, since the minimization is over a finite set.

The set of MB policies may have more than one element. This could happen, for example, when at a given time slot n, a server k is connected to two or more queues of equal size, which happen to be the longest queues connected to this server. Then,

serving either one of them will satisfy Equation (2.11) even if these allocations result in different $\mathbf{Y}(n)$ (i.e., different policies). The resulting queue length vectors ($\hat{x}(n)$) under any of these policies will be permutations of each other.

Remark: Note that the LCQ policy in [23] is a most balancing (MB) policy for $K = 1$ (i.e., the one server system presented in [23]). Extension of LCQ to $K > 1$ (i.e., allocating all the servers to the longest queue) may not result in a MB policy. □

2.4.1 Possible Implementation of MB Policies

A determination of an MB policy given $\mathbf{X}(t)$ and $\mathbf{G}(t)$ can be done using a direct search over all possible server allocations. This can be a challenging computational task. In what follows we present a more efficient approach for the construction of an MB policy.

We consider a given permutation (ordering) of the K servers in the system. For this permutation we define a "sequential LCQ server allocation," a process of allocating the servers to queues in K steps as follows: Starting from the first server, we assign it to its longest connected queue and we update (i.e., reduce by one) the queue size. We continue with the second server following the same principle until we exhaust all servers in K steps. There are $K!$ server orderings that we have to consider. We will show that at least one "sequential LCQ server allocation" corresponding to an ordering among the $K!$ server permutations will result in an MB policy. We introduce the following notation:

We define the set $\mathcal{M}_i^t$ as the set of servers connected to queue i during time slot t. Let $M_i(t) \triangleq |\mathcal{M}_i^t|$ be the number of servers that are connected to queue i during time slot t, that is

$$M_i(t) = \sum_{j=1}^{K} G_{i,j}(t) \tag{2.12}$$

Let $\mathbb{Q}_k \triangleq \{i : k \in \mathcal{M}_i^t\}$ denote the set of queues that are connected to server k

during time slot t; we omit the dependence on t to simplify notation. Let Θ denote the set of all possible permutations of the set $\{1, \ldots, K\}$. We define a server ordering at time n as a permutation $\theta(n) \in \Theta$. There are $|\Theta| = K!$ possible server orderings. We use the subscript $[j]^\theta$ to denote the j^{th} server to be allocated under the ordering rule $\theta(n)$. Algorithm 1 presents the pseudo-code for the approach we described previously.

Algorithm 1 (MB Policy Implementation)**.**

1. $for \quad t = 1, 2, \ldots \quad do \quad \Big\{$
2. *Input:* $\mathbf{X}(t), \mathbf{G}(t)$. *Calculate:* $\mathbb{Q}_{[k]}$, $k = 1, \ldots, K$.
3. *Let:* $\kappa_t^{min} = L \cdot \max\limits_{l} X_l$ *; maximum possible* κ_t
4. $forall \quad \theta \in \Theta \quad do \quad \Big\{$ *; loop* $|\Theta| = K!$ *times*
5. $\mathbf{X}' \longleftarrow \mathbf{X}(t)$, $\mathbf{Y}' \longleftarrow \mathbf{0}$, $\mathbf{Q}' \longleftarrow \mathbf{0}$
6. $for \quad j = 1 \quad to \quad K \Big\{$ *; allocate servers sequentially*
7. $Q'_{[j]^\theta} = \min \left(k : k \in \left\{ \underset{l: l \in \mathbb{Q}_{[j]^\theta}}{\operatorname{argmax}} (X'_l | X'_l > 0) \right\} \right)$
8. *Let:* $i = Q'_{[j]^\theta}$
9. $Y'_i = Y'_i + 1$
10. $X'_i = X'_i(t) - 1 \quad \Big\}$
11. *Compute:* κ_t^θ *from Equation(2.10)*
12. $if \quad (\kappa_t^\theta < \kappa_t^{min}) \quad \Big\{$
13. $\kappa_t^{min} = \kappa_t^\theta$
14. $\mathbf{y}(t) \longleftarrow \mathbf{Y}'$, $\mathbf{q}(t) \longleftarrow \mathbf{Q}'$, $\theta(t) \longleftarrow \theta \; \Big\}$
15. $\Big\} \quad \Big\} \quad$ *; End of Algorithm 1.*

2.5 Theoretical Background

We will have to present some theoretical background before tackling the proof of the optimality of MB policies.

2.5.1 Stochastic Dominance

Stochastic ordering of random variables and random processes is explained by Stoyan [20]. Readers may also consult Ross [41].

Let $f : \mathcal{R}^n \to \mathcal{R}$ be a measurable function such that $f(x) \leq f(y)$ for every real valued sequences $x, y \in \mathcal{R}^n, n \in \mathcal{Z}_+$ such that $x_i(t) \leq y_i(t)\ \forall\, i$ and $t \in \mathcal{Z}_+$. A discrete-time stochastic process, $\mathbf{X} = \{X(t)\}_{t=1}^{\infty}$ is stochastically smaller than the process $\mathbf{Y} = \{Y(t)\}_{t=1}^{\infty}$ (written $\mathbf{X} \leq_{st} \mathbf{Y}$), if $P[f(\mathbf{X}) > a] \leq P[f(\mathbf{Y}) > a]$ for all $a \in \mathcal{R}$. According to Stoyan [20], $\mathbf{X} \leq_{st} \mathbf{Y}$ if for all t, $X(t) \leq_{st} Y(t)$.

The following proposition, known as *coupling*, provides an important tool that is used frequently to prove stochastic order relationships. We will use it in our proof. For more information about coupling methods, the reader may refer to [21].

Proposition 1: [14] *If F and G are two distributions such that $F(z) \geq G(z)$, then there exist random variables X' and Y' that are distributed according to F and G respectively, such that*

$$P[X' \leq Y'] = 1.$$

The proof is given in Appendix B.1.

Corollary 1. *If $\mathbf{X} \leq_{st} \mathbf{Y}$, then there exists a stochastic process $\mathbf{X}' = \{X'(t)\}_{t=1}^{\infty}$ with the same probability distribution as $\mathbf{X}$ such that, $X'(t) \leq Y(t)$ for every $t \in \mathcal{Z}_+$.*

Proof. The proof of the corollary is straightforward from proposition 1 and the relation $X'(t) \leq_{st} Y(t),\ \forall t$ implies $\mathbf{X}' \leq_{st} \mathbf{Y}$. □

Theorem 1. *Let* $\mathbf{X}$ *and* $\mathbf{Y}$ *be two discrete-time random processes on a common probability space. Then* $\mathbf{X} \leq_{st} \mathbf{Y}$ *is equivalent to* $P[f(X(t_1), \ldots, X(t_n)) > z] \leq P[f(Y(t_1), \ldots Y(t_n)) > z]$ *for all* $\{t_1, \ldots, t_n\}$, n, z, *and for any measurable, monotonically increasing functions* $f : \mathcal{R}^n \to \mathcal{R}_+$, *i.e.,* $x_j \leq y_j$ *for all* $1 \leq j \leq n$ *implies* $f(x_1, \ldots, x_n) \leq f(y_1, \ldots, y_n)$.

The proof is given in Appendix B.2.

2.5.2 The Coupling Method

If we want to perform some kind of comparison of probability measures on a measurable space, then sometimes it is possible to construct random elements, with these measures as their distributions, on a common probability space, such that the comparison can be carried out in terms of these random elements rather than the probability measures. The term *coupling* is usually used to refer to any such construction. A formal definition of coupling of two probability measures on the measurable space $(E, \mathcal{E})$ (the state space, e.g., $E = \mathcal{R}, \mathcal{R}^d, \mathcal{Z}_+, etc.$) is given below [21].

A coupling of the probability measures P *and* P' *on a measurable space* $(E, \mathcal{E})$ *is a probability measure* $\hat{P}$ *on* $(E^2, \mathcal{E}^2)$ *such that*

$$P = \hat{P}u^{-1} \text{ and } P' = \hat{P}(u')^{-1}, \tag{2.13}$$

where $u(x, x') = x, u'(x, x') = x'$ *for* $(x, x') \in E^2$. *Therefore,* P *and* P' *are marginals of* $\hat{P}$.

The above definition is not easy to utilize in practical sense. A more usable definition will follow. First, we define a random element in $(E, \mathcal{E})$ to be a quadruple $(\Omega, \mathscr{F}, \mathbf{P}, X)$, where $(\Omega, \mathscr{F}, \mathbf{P})$ is the sample space and X is the class of measurable mappings from Ω to E (X is an E-valued random variable, s.t. $X^{-1}(B) \in \mathscr{F}$ for all

$B \in \mathcal{E}$).

Definition: *A coupling of the two random elements* $(\Omega, \mathscr{F}, \mathbf{P}, X)$ *and* $(\Omega', \mathscr{F}', \mathbf{P}', X')$ *in* $(E, \mathcal{E})$ *is a random element* $(\hat{\Omega}, \hat{\mathscr{F}}, \hat{\mathbf{P}}, (\hat{X}, \hat{X}'))$ *in* $(E^2, \mathcal{E}^2)$ *such that*

$$X \stackrel{\mathscr{D}}{=} \hat{X} \quad \text{and} \quad X' \stackrel{\mathscr{D}}{=} \hat{X}', \tag{2.14}$$

where $\stackrel{\mathscr{D}}{=}$ *means 'equal in distribution'.*

This means that $\hat{\mathbf{P}}(\hat{X}, \hat{X}')^{-1}$ is a coupling of $\mathbf{P}X^{-1}$ and $\mathbf{P}'X'^{-1}$ in the same sense as in (2.13) above.

The following corollary is a direct result of Corollary 1 above. We will use it to show that the above definition can be utilized even when X and X' are sequences of random variables (or discrete random processes) and the state space $(E, \mathcal{E})$ is a vector space.

Corollary 2. *If* $\mathbf{X}$ *and* $\mathbf{Y}$ *are two discrete time random processes such that* $\mathbf{X} \stackrel{\mathscr{D}}{=} \mathbf{Y}$*, then there exist a stochastic process* $\mathbf{X}' = \{X'(t)\}_{t=1}^{\infty}$ *with the same probability distribution as* $\mathbf{X}$ *such that,* $X'(t) = Y(t)$ *almost surely for every* $t \in \mathcal{Z}_+$*.*

Corollary 2 is a strict version of Corollary 1 (since equal in distribution is a special case of $\leq_{st}$ which is basically an ordering in distribution).

Remark: The above definition makes no assumption about the distribution of the random variables in $\mathbf{X}$; for example, $\mathbf{X}$ may be a sequence of non-i.i.d. random variables. □

2.6 Optimality of MB Policies

In this section, we present the main result in this chapter, that is, the optimality of MB policies among all feasible policies. We will establish the optimality of MB

policies for a wide range of performance criteria including the minimization of the total number of packets in the system. We introduce the following definition first.

2.6.1 Definition of Preferred Order

First we define the relation $\preceq$ on $\mathcal{Z}_+^{(L+1)}$; we write $\tilde{\mathbf{x}} \preceq \mathbf{x}$ if:

1- $\tilde{x}_i \leq x_i$ for all i (i.e., point wise comparison),

2- $\tilde{\mathbf{x}}$ is obtained from $\mathbf{x}$ by permuting two of its components; the two vectors differ only in two components $i > 0$ and $j > 0$, such that $\tilde{x}_i = x_j$ and $\tilde{x}_j = x_i$, or

3- $\tilde{\mathbf{x}}$ is obtained from $\mathbf{x}$ by performing a *"balancing interchange"*. The two vectors differ in two components $i > 0$ and $j > 0$ only, where $x_j \leq \tilde{x}_j \leq x_i$ and $x_j \leq \tilde{x}_i \leq x_i$, under the constraints that: $\tilde{x}_i = x_i - 1$ and $\tilde{x}_j = x_j + 1$.

To prove the optimality of MB policies, we will need a methodology that enables comparison of the queue lengths under different policies. Towards this end, we define a "preferred order" as follows:

Definition: (Preferred Order). The transitive closure of the relation $\preceq$ defines a partial order (which we call *preferred order* and use the symbol $\prec_p$ to represent) on the set $\mathcal{Z}_+^{(L+1)}$. □

The transitive closure [42] of $\preceq$ on the set $\mathcal{Z}_+^{(L+1)}$ is the smallest transitive relation on $\mathcal{Z}_+^{(L+1)}$ that contains the relation $\preceq$. From the engineering point of view, $\tilde{\mathbf{x}} \prec_p \mathbf{x}$ if $\tilde{\mathbf{x}}$ is obtained from $\mathbf{x}$ by performing a sequence of *reductions*, *permutations of two components* and/or *balancing interchanges*.

For example, if $\tilde{\mathbf{x}} = (3, 4, 5)$ and $\mathbf{x} = (4, 5, 3)$ then $\tilde{\mathbf{x}} \prec_p \mathbf{x}$ since $\tilde{\mathbf{x}}$ can be obtained from $\mathbf{x}$ by performing the following two consecutive two-component permutations: first swap the second and third components of $\mathbf{x}$, yielding $\mathbf{x}^1 = (4, 3, 5)$ then swap the first and second components of $\mathbf{x}^1$, yielding $\mathbf{x}^2 = (3, 4, 5) = \tilde{\mathbf{x}}$.

Suppose that $\tilde{\mathbf{x}}, \mathbf{x}$ represent queue size vectors. Case (3) in this case describes moving a packet from one real, large queue i to another real, smaller one j (note that the queue with index $j = 0$ is excluded since it represents the dummy queue in our formulation). We say that $\tilde{\mathbf{x}}$ is *more balanced* than $\mathbf{x}$ when (3) is satisfied. For example, if $L = 2$ and $\mathbf{x} = (0, 5, 2)$ then a balancing interchange (where $i = 1$ and $j = 2$) will result in $\tilde{\mathbf{x}} = (0, 4, 3)$. In summary, the queue size vector $\tilde{\mathbf{x}}$ is preferred over $\mathbf{x}$ ($\tilde{\mathbf{x}} \prec_p \mathbf{x}$) if $\tilde{\mathbf{x}}$ can be obtained from $\mathbf{x}$ by performing a sequence of packet removals, permutations or balancing interchanges.

2.6.2 The Class $\mathcal{F}$ of Cost Functions

Let $\tilde{\mathbf{x}}, \mathbf{x} \in \mathcal{Z}_+^{(L+1)}$ be two vectors representing queue lengths. Then we denote by $\mathcal{F}$ the class of real-valued functions on $\mathcal{Z}_+^{(L+1)}$ that are monotone, non-decreasing with respect to the partial order $\prec_p$; that is, $f \in \mathcal{F}$ if and only if

$$\tilde{\mathbf{x}} \prec_p \mathbf{x} \Rightarrow f(\tilde{\mathbf{x}}) \leq f(\mathbf{x}) \tag{2.15}$$

From (2.15) and the definition of preferred order, it can be easily seen that the function $f(\mathbf{x}) = x_1 + x_2 + \ldots + x_L$ belongs to $\mathcal{F}$. This function corresponds to the total number of queued packets in the system[1].

2.6.3 Definition of The Subsets $\Pi_n^h, 0 \leq h \leq K$

Recall that in Section 2.3 we defined a dummy queue that we refer to as queue 0. Allocating a server to queue 0 is equivalent to idling that server. We have $x_0(1) = 0$ by assumption. Furthermore, we let $z_0(n) = y_0(n), \forall n$. We define $\hat{x}_0(n) = x_0(n) - y_0(n)$ such that $\hat{x}_0(n) \in \mathcal{Z}$, i.e., we allow $\hat{x}_0(n)$ to be negative. This definition will insure

[1]Another example is the function $f'(\mathbf{x}) = \max\{x_1, \ldots, x_L\}$ which also belongs to the class $\mathcal{F}$.

that $x_0(n) = 0, \forall n$ and therefore will not affect the calculation of the cost function $f(\cdot) \in \mathcal{F}$.

For any fixed $n \geq 1$, let Π_n denote the set of policies that have the MB property at time slots $t = 1, 2, \ldots, n$. We can easily see that these sets form a monotone sequence, with $\Pi_n \subseteq \Pi_{n-1}$. Then the set Π^{MB} in Equation (2.11) can be defined as $\Pi^{MB} = \bigcap_{n=1}^{\infty} \Pi_n$.

Two-queue packet interchanges

Let $f \in \{0, 1, \ldots, L\},\ t \in \{0, 1, \ldots, L\}$ represent the indices of two queues that we refer to as the from and to queues. Define the $(L+1) \times 1$-dimensional vector $I(f,t)$, whose j-th element is given by[2]:

$$I_j(f,t) \quad = \quad \begin{cases} 0, & t = f; \\ +1, & j = f,\ f \neq t; \\ -1, & j = t,\ t \neq f; \\ 0, & \text{otherwise.} \end{cases} \tag{2.16}$$

Let $\mathbb{1}_{\{A\}}$ denote the usual indicator function. We can restate (2.16) as follows

$$I_j(f,t) = \mathbb{1}_{\{j=f\}} - \mathbb{1}_{\{j=t\}}, \quad \forall j = 0, 1, \ldots, L \tag{2.17}$$

Fix an initial state $\mathbf{x}(n)$ at time slot n; consider a policy π with a (feasible) withdrawal vector $\mathbf{y}(n)$. Let

$$\mathbf{y}^*(n) = \mathbf{y}(n) + I(f,t), \quad f \neq t, \tag{2.18}$$

[2]Intuitively, $I(f,t)$ represents an operation of removing a packet 'from' queue f and adding it 'to' queue t.

be another withdrawal vector. The two vectors $\mathbf{y}(n), \mathbf{y}^*(n)$ differ only in the two components t, f; under the withdrawal vector $\mathbf{y}^*(n)$, an additional packet is removed from queue f, while one packet less is removed from queue t. Note that either t or f can be the dummy queue.

$$y_f^*(n) = y_f(n) + 1 \tag{2.19}$$

$$y_t^*(n) = y_t(n) - 1 \tag{2.20}$$

$$y_i^*(n) = y_i(n), \quad \forall i \neq f, t. \tag{2.21}$$

In the sequel, we will call $I(f,t)$ an *interchange* between queues f and t. We will call $I(f,t)$ a *feasible interchange* if it results in a feasible withdrawal vector $\mathbf{y}^*(n)$. From Equations (2.7) and (2.19) – (2.21), it is clear that the $I(f,t)$ interchange will result in a new vector $\hat{\mathbf{x}}^*(n)$ such that:

$$\hat{x}_f^*(n) = \hat{x}_f(n) - 1, \quad f \in \{0, 1, \ldots L\} \tag{2.22}$$

$$\hat{x}_t^*(n) = \hat{x}_t(n) + 1, \quad t \in \{0, 1, \ldots L\} \tag{2.23}$$

$$\hat{x}_i^*(n) = \hat{x}_i(n), \quad \forall i \neq f, t; \quad i \in \{0, 1, \ldots L\} \tag{2.24}$$

or, in vector notation,

$$\hat{\mathbf{x}}^*(n) = \hat{\mathbf{x}}(n) - I(f,t), \quad f \neq t. \tag{2.25}$$

Implementation of two-queue packet interchanges

The interchange $I(f,t)$ in Equation (2.18) can be implemented via a series of $m \geq 1$ server reallocations. For example, suppose that a server k is connected to both queues t and f (i.e., $g_{f,k}(n) \cdot g_{t,k}(n) = 1$), and that the server is allocated to queue t, under $\mathbf{y}(n)$ (i.e., $\mathbb{1}_{\{q_k(n)=t\}} = 1$ and $\hat{x}_f(n) \geq 1$). Then reallocating server k to queue f will result in the interchange $I(f,t)$ in Equation (2.18); this is the case $m = 1$.

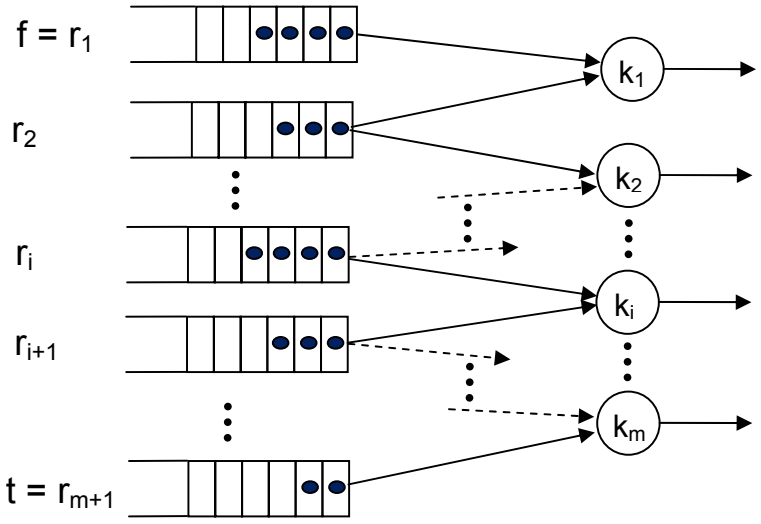


Figure 2.2: The sequence of m server interchanges that implement the interchange $I(f,t)$.

Note that the condition $g_{f,k}(n) \cdot g_{t,k}(n) \cdot \mathbb{1}_{\{q_k(n)=t\}} = 1$ and $\hat{x}_f(n) \geq 1$ is sufficient but not necessary for $I(f,t)$ to be feasible.

In general, we define the series of m server reallocations required to implement $I(f,t)$ through the sequence of indices of the queues involved in that interchange. Let $\mathbf{r} \in \mathcal{Z}^{m+1}$ be such a sequence, where $r_1 = f$ and $r_{m+1} = t$. Let $k_i : k_i \in \{1, 2, \ldots, K\}$ be the server reallocated to queue r_i from queue r_{i+1}.

For the interchange operation of Equation (2.18), the following are sufficient feasibility constraints:

$$\sum_{i=1}^{m} g_{r_i,k_i}(n) \cdot g_{r_{i+1},k_i}(n) \cdot \mathbb{1}_{\{q_{k_i}(n)=r_{i+1}\}} = m, \tag{2.26}$$

$$\hat{x}_f(n) \geq 1, \quad \text{if } f \in \{1, 2, \ldots, L\}, \tag{2.27}$$

$$\hat{x}_i(n) \geq 0, \quad \text{if } \forall i \neq f, i \in \{1, 2, \ldots, L\} \tag{2.28}$$

$$\hat{x}_0(n) \leq 0, \tag{2.29}$$

for some integer $m \geq 1$ and $\mathbf{r} \in \mathcal{Z}^{m+1}$.

Constraint (2.26) is sufficient to ensure that connectivity conditions allow for the series of m server reallocations. Server k reallocation to queue j is feasible only if queue j is non-empty (i.e., $\hat{x}_j(n) \geq 1$) and is connected to server k (i.e., $g_{j,k}(n) = 1$). Furthermore, the feasibility of $\mathbf{y}^*(n)$ implies that constraint (2.26) must hold for at least one $m \geq 1$ and one $\mathbf{r} \in \mathcal{Z}^{m+1}$. Therefore, if $C(\mathbf{r}, m)$ is one such constraint, then $\bigcup_{\mathbf{r},m} C(\mathbf{r}, m), 1 \leq m \leq K, \mathbf{r} \in \mathcal{Z}^{m+1}$ is a "necessary" feasibility constraint. Note that $1 \leq m \leq K, \mathbf{r} \in \mathcal{Z}^{m+1}$ are exhaustive in the sense that they cover all possible server allocations.

Constraint (2.27) is necessary since a packet will be removed from queue f to be added to queue t, therefore, queue f must contain at least one packet for the interchange to be feasible. A special case is the dummy queue (queue 0) that has queue length equal to 0 at the beginning of every time slot and therefore, $\hat{x}_0(n)$ has a length that does not exceed 0. The remaining queues may be empty. The sequence of intermediate interchanges starts by removing a packet from queue $f = r_1$ and adding a packet to queue r_2. Therefore, constraints (2.26) – (2.29) insure that queue r_2 will contain at least one packet for the second intermediate server reallocation to be feasible even when $\hat{x}_{r_2}(n) = 0$. Same is true for any queue $r_i, i \in \{2, 3, \ldots m\}$. Therefore, these constraints are also sufficient for feasibility of (2.18).

"Balancing" two-queue packet interchanges

In light of the above definition, we can restate the definition of balancing interchange as follows:

Definition: A feasible interchange $I(f, t)$ is called "*balancing*" if

$$\hat{x}_f(n) \geq \hat{x}_t(n) + 1 \tag{2.30}$$

$I(f,t)$ is called "*unbalancing*" if

$$\hat{x}_f(n) \leq \hat{x}_t(n) \tag{2.31}$$

According to Lemma 11 of Appendix A, a balancing interchange will not increase the imbalance index. In particular, the new policy π^* resulting from applying one balancing interchange to policy π's withdrawal vector $\mathbf{y}(n)$ (in accordance with Equation 2.18) during time slot n will have a cost that is given by:

$$\kappa_n(\pi^*) = \kappa_n(\pi) - 2(s-l) \cdot \mathbb{1}_{\{\hat{x}_{[l]}(n) \geq \hat{x}_{[s]}(n)+2\}} \tag{2.32}$$

where l (respectively s) is the order of queue f (respectively t) in $\hat{x}(n)$ when ordered in descending manner[3]. By the definition of the balancing interchange, we have $s > l$.

In words, Equation (2.32) states that an interchange $I(f,t)$, when balancing, results in a cost reduction of $2(s-l)$ when $\hat{x}_f(n) = \hat{x}_{[l]}(n) \geq \hat{x}_{[s]}(n) + 2 = \hat{x}_t(n) + 2$ and keeps it unchanged otherwise, i.e., when $\hat{x}_f(n) = \hat{x}_t(n) + 1$. The latter makes intuitive sense, since the balancing interchange in this case will result in permuting the lengths of queues f and t which does not change the total sum of differences (and hence the imbalance index) in the resulted queue length vector. Check the second part of the proof of Lemma 11 for details.

Lemma 1. *The feasible interchange $I(f,0), f > 0$ is a balancing interchange.*

Proof. By definition, $x_0(n) = 0$. Since $y_0(n) \geq 0$, therefore $\hat{x}_0(n) = x_0(n) - y_0(n) \leq 0$. According to the feasibility constraint (2.27), the interchange $I(f,0)$ is feasible only when $\hat{x}_f(n) \geq 1$. Therefore, $\hat{x}_f(n) \geq \hat{x}_0(n) + 1$, and it follows that $I(f,0)$ is a balancing interchange. □

[3]Intuitively, we use s and l to refer to a "shorter" and a "longer" queues respectively.

The difference vector D

The following defines a difference vector of policy π from MB policy during a given time slot n. This vector and the results we present in this section are crucial for the proof of the optimality of MB policies.

Definition: Consider a given state $(\mathbf{x}(n), \mathbf{g}(n))$ and a policy $\pi \in \Pi_{n-1}$ that chooses the feasible withdrawal vector $\mathbf{y}(n)$ during time slot n. Let $\mathbf{y}^{MB}(n)$ be a withdrawal vector chosen by an MB policy during the same time slot n. We define the $(L+1) \times 1$-dimensional vector $\mathbf{D} \in \mathcal{Z}^{L+1}$ as follows:

$$\mathbf{D} = \mathbf{y}^{MB}(n) - \mathbf{y}(n) \tag{2.33}$$

where, for simplicity, we omit the dependence of $\mathbf{D}$ on the policy π.

Lemma 2. *For a given policy $\pi \in \Pi_{n-1}$ and a time slot n,*

$$\sum_{i=0}^{L} D_i \cdot \mathbb{1}_{\{D_i>0\}} = -\sum_{j=0}^{L} D_j \cdot \mathbb{1}_{\{D_j<0\}} \tag{2.34}$$

i.e., the sum of all positive elements of $\mathbf{D}$ *equals the sum of all negative elements of* $\mathbf{D}$.

Proof. For any withdrawal vector $\mathbf{y}$, we have

$$\sum_{i=0}^{L} y_i(n) = K,$$

where K is the number of servers. From equation (2.33), we have then:

$$\begin{aligned} \sum_{i=0}^{L} D_i &= \sum_{i=0}^{L} y_i^{MB}(n) - \sum_{i=0}^{L} y_i(n) \\ &= K - K = 0, \end{aligned}$$

and Equation (2.34) follows. □

Lemma 3. *Consider a given state* $(\mathbf{x}(n), \mathbf{g}(n))$ *during time slot* n. *Let* $f, t \in \{0, 1, \ldots, L\}$ *be any two queues such that* $I(t, f)$ *is feasible. A policy* $\pi \in \Pi$ *that results in* $\hat{x}_f(n) \leq \hat{x}_t(n) - 2$ *is not an MB policy.*

Proof. The interchange $I(t, f)$ is a balancing interchange by definition. Since $\hat{x}_f(n) \leq \hat{x}_t(n) - 2$, then the balancing interchange $I(t, f)$ reduces the imbalance index by a factor of two according to Equation (2.32). Therefore, π does not achieve the minimum imbalance index during time slot n, and hence, is not an MB policy. □

Lemma 4. *Consider a given state* $(\mathbf{x}(n), \mathbf{g}(n))$ *and a feasible withdrawal vector* $\mathbf{y}(n)$. *Any feasible interchange* $I(f, t)$ *where* $D_f \geq +1$, $D_t \leq -1$ *is a balancing interchange.*

Proof. We observe that $f \neq t$ must be true. Otherwise, we arrive at a contradiction, i.e., $+1 \leq D_f \leq -1$. This leaves three cases to consider:

Case 1: $f = 0$.

This case is not possible by contradiction. By assumption, $D_0 \geq +1$, which means that $y_0^{MB}(n) \geq y_0(n) + 1$. This case states that an MB policy idled at least one more server than π. Therefore, $\hat{x}_0^{MB}(n) \leq -1$. This makes queue 0 the shortest queue. Allocating the idled server to queue t, i.e., the interchange $I(t, 0)$, is both feasible (since $\mathbf{y}(n)$ is feasible by assumption) and balancing (by Lemma 1). The interchange $I(t, 0)$ will result in a withdrawal vector $\mathbf{y}'(n) = \mathbf{y}^{MB} + I(t, 0)$.

Let s be the order of queue $f = 0$ when ordering the vector $\hat{\mathbf{x}}^{MB}(n)$ in a descending manner. Therefore, $s = L + 1$. Furthermore, in order for $I(t, 0)$ to be feasible queue t must not be empty (according to feasibility constraint (2.27)) which implies that $\hat{x}_t^{MB}(n) \geq 1$ and the order of queue t is $l < s$. Therefore, $\hat{x}_f^{MB}(n) \leq \hat{x}_t^{MB}(n) - 2$ and the interchange $I(t, 0)$ will reduce the imbalance index by $2(s - l)$ according to Equation (2.32). This implies that the new policy has less imbalance index than an

MB policy. This contradicts the fact that any MB policy minimizes the imbalance index.

Case 2: $t = 0$.

When $t = 0$ then the interchange $I(f, t)$ is the process of allocating an idled server to queue $f > 0$. This, according to Lemma 1, is a balancing interchange.

Case 3: $t, f > 0$.

We will show that this case will also result in a balancing interchange. Let $\mathbf{y}(n)$ be the original withdrawal vector. Let $\mathbf{y}^*(n)$ be the withdrawal vector resulted from the interchange $I(f, t)$, i.e.,

$$\mathbf{y}^*(n) = \mathbf{y}(n) + I(f, t)$$

Using the assumption $D_t \leq -1$ and Equation (2.20), we arrive at the following:

$$\begin{aligned} y_t^{MB}(n) - y_t(n) &\leq -1 \\ y_t^{MB}(n) &\leq y_t(n) - 1 = y_t^*(n) \\ y_t^{MB}(n) &\leq y_t^*(n) = y_t(n) - 1 \end{aligned} \tag{2.35}$$

and,

$$\begin{aligned} x_t(n) - y_t^{MB}(n) &\geq x_t(n) - y_t^*(n) = x_t(n) - (y_t(n) - 1) \\ \hat{x}_t^{MB}(n) &\geq \hat{x}_t^*(n) = \hat{x}_t(n) + 1, \quad t > 0 \end{aligned} \tag{2.36}$$

Similarly, using the assumption $D_f \geq +1$ and Equation (2.19), we have

$$\begin{aligned} y_f^{MB}(n) - y_f(n) &\geq +1 \\ y_f^{MB}(n) &\geq y_f(n) + 1 = y_f^*(n) \\ y_f^{MB}(n) &\geq y_f^*(n) = y_f(n) + 1 \end{aligned} \tag{2.37}$$

and,

$$\begin{aligned} x_f(n) - y_f^{MB}(n) \leq x_f(n) - y_f^*(n) = x_f(n) - (y_f(n) + 1) \\ \hat{x}_f^{MB}(n) \leq \hat{x}_f^*(n) = \hat{x}_f(n) - 1, \quad f > 0 \end{aligned} \tag{2.38}$$

To show that $I(f,t)$ in this case is a balancing interchange, we have to show that $\hat{x}_f(n) \geq \hat{x}_t(n) + 1$. Suppose to the contrary that $\hat{x}_f(n) \leq \hat{x}_t(n)$; then, from Equations (2.36) and (2.38), we have

$$\begin{aligned} \hat{x}_f(n) &\leq \hat{x}_t(n) \\ \hat{x}_f^*(n) + 1 &\leq \hat{x}_t^*(n) - 1 \\ \hat{x}_f^*(n) &\leq \hat{x}_t^*(n) - 2 \end{aligned} \tag{2.39}$$

From (2.36) and (2.38), we have

$$\begin{aligned} \hat{x}_f^{MB}(n) &\leq \hat{x}_f^*(n) \leq \hat{x}_t^*(n) - 2 \leq \hat{x}_t^{MB}(n) - 2 \\ \hat{x}_f^{MB}(n) &\leq \hat{x}_t^{MB}(n) - 2 \end{aligned} \tag{2.40}$$

The differences $D_f \geq +1$ and $D_t \leq -1$ by assumption, i.e., there is at least one more (respectively one less) server allocated to queue f (respectively queue t) under the MB policy. Therefore, $y^*(n) = y^{MB}(n) + I(t,f)$ is feasible. Therefore, Inequality

(2.40) is a contradiction according to Lemma 3. Therefore, $\hat{x}_f(n) \geq \hat{x}_t(n) + 1$ and by definition the interchange $I(f,t)$ is a balancing interchange $\square$

Lemma 5. *Consider a given state* $(\mathbf{x}(n), \mathbf{g}(n))$ *and a policy* π*. If* $\mathbf{D} = 0$*, then* π *has the MB property. Conversely, if* π *has the MB property, the vector* $\mathbf{D}$ *has components that are* $0, +1,$ *or* -1 *only.*

Proof. Assume that $\mathbf{D} = 0$; then, using Equation (2.33), we have:

$$\begin{aligned} \mathbf{y}(n) &= \mathbf{y}^{MB}(n) \\ \mathbf{x}(n) - \mathbf{y}(n) &= \mathbf{x}(n) - \mathbf{y}^{MB}(n) \\ \hat{\mathbf{x}}(n) &= \hat{\mathbf{x}}^{MB}(n) \end{aligned} \tag{2.41}$$

From Equations (2.41) and (2.10), we have that $\kappa_n(\pi) = \kappa_n(\pi^{MB})$ and thus π has the MB property during time slot n.

To prove the converse part of the lemma, assume that π has the MB property. Therefore, $\kappa_n(\pi) = \kappa_n(\pi^{MB})$. From Lemma 11 this is only possible if either: (i) $\hat{\mathbf{x}}(n) = \hat{\mathbf{x}}^{MB}(n)$, or (ii) $\hat{\mathbf{x}}(n)$ is obtained by performing a balancing interchange between the pair of the l^{th} and the s^{th} longest queues ($l < s$) in $\hat{\mathbf{x}}^{MB}(n)$ such that $\hat{x}_{[l]}(n) = \hat{x}_{[s]}(n) + 1$, is satisfied; note that there may be multiple such queue pairs. The balancing interchange in case (ii) will affect the length of two queues only (call them i and j) such that $\hat{x}_i(n) = \hat{x}_i^{MB}(n) - 1$ and $\hat{x}_j(n) = \hat{x}_j^{MB}(n) + 1$, where $i = [l]$ and $j = [s]$ (for each given pair). Therefore,

$$\begin{aligned} y_i(n) &= x_i(n) - \hat{x}_i(n) = x_i(n) - (\hat{x}_i^{MB}(n) - 1) \\ &= y_i^{MB}(n) + 1, \end{aligned} \tag{2.42}$$

and,

$$\begin{aligned} y_j(n) &= x_j(n) - \hat{x}_j(n) = x_j(n) - (\hat{x}_j^{MB}(n) + 1) \\ &= y_j^{MB}(n) - 1, \end{aligned} \tag{2.43}$$

while withdrawals from the remaining queues will be the same, i.e.,

$$y_b(n) = y_b^{MB}(n), \forall b \neq i, j. \tag{2.44}$$

From Equations (2.42) through (2.44), we conclude that the vector $\mathbf{D}$ has components that are $0, +1$, or -1 only. □

Lemma 6. *For any policy $\pi \in \Pi_{n-1}$, at most $0 \leq \sum_i |D_i|/2 \leq K$ balancing interchanges are required to make the resulting policy $\pi^* \in \Pi_n$.*

Proof. The policy $\pi \in \Pi_{n-1} \subseteq \Pi$ and therefore $\mathbf{y}(n)$ is a feasible withdrawal vector. A necessary feasibility condition is the one given by Equation (2.6), i.e.,

$$\sum_{i=0}^{L} y_i(n) = K \tag{2.45}$$

Therefore, the total difference between the two vectors is bounded by:

$$0 \leq \sum_{i=0}^{L} |y_i^{MB}(n) - y_i(n)| \leq 2K, \tag{2.46}$$

or equivalently,

$$0 \leq \sum_{i=0}^{L} |D_i| \leq 2K \tag{2.47}$$

When the sum equals 0, then π has the MB property during time slot n according to Lemma 5. When the sum equals $2K$, then one can conclude using Lemma 2 that the

sum of positive terms is equal to the sum of negative terms in the summation above. To put it differently, there are at most K (+1)'s and K (-1)'s in the difference vector $\mathbf{D}$.

According to Lemma 4, $I(f,t)$ where $D_f \geq +1$ and $D_t \leq -1$ is a balancing interchange. Since we have $\sum_i |D_i|/2$ such pairs of queues, then applying the balancing interchange described by Lemma 4 for $\sum_i |D_i|/2$ times will result in a new difference vector $\mathbf{D}^* = 0$, i.e., $\mathbf{y}^*(n) = \mathbf{y}^{MB}(n)$. To show that an interchange $I(f,t)$ that is feasible does exist, consider the following:

Consider a given state $(\mathbf{x}(n), \mathbf{g}(n))$ and two policies $\pi, \pi' \in \Pi$ that choose the feasible withdrawal vectors $\mathbf{y}(n), \mathbf{y}'(n)$ during time slot n. Let $\mathbf{q}(n), \mathbf{q}'(n)$ denote two implementations of $\mathbf{y}(n), \mathbf{y}'(n)$ respectively. We can write

$$\mathbf{y}'(n) = \mathbf{y}(n) + \sum_{k=1}^{K} I(q'_k(n), q_k(n)) \tag{2.48}$$

where, by definition, server k is connected to both queues $q_k(n)$ and $q'_k(n)$. Therefore, each interchange $I(q'_k(n), q_k(n))$ is feasible. Note that $q_k(n) = q'_k(n)$ is possible, for some k, in which case $I(q'_k(n), q_k(n)) = 0$. By construction, all the interchanges in the right hand side of Equation (2.48) are feasible. If π' is an MB policy then for any arbitrary feasible policy π, there exist a sequence of feasible interchanges that satisfy Equation (2.48).

Therefore, we conclude that for any arbitrary feasible policy $\pi \in \Pi_{n-1}$ and a corresponding withdrawal vector $\mathbf{y}(n)$, at most $\sum_i |D_i|/2$ feasible balancing interchanges are required to make $\mathbf{y}^*(n) = \mathbf{y}^{MB}(n)$ and hence the resulting policy $\pi^* \in \Pi_n$. □

Given the state $(\mathbf{x}(n), \mathbf{g}(n))$ and a policy π that resulted in a withdrawal vector $\mathbf{y}(n)$. We define $h \in \mathcal{Z}_+$ as the distance of π from an MB policy (in terms of balancing

interchanges) during time slot n. Using Lemma 6, h can be calculated as follows:

$$h = \sum_{i=0}^{L} |D_i|/2, \tag{2.49}$$

According to Lemma 6, we have $0 \leq h \leq K$. If $h = 0$ then π has the MB property during time slot n according to Lemma 5.

Now we are ready to define the sets Π_n^h.

Definition: Define the set $\Pi_n^h, 0 \leq h \leq K$ as the set that contains all feasible policies $\pi \in \Pi_{n-1}$, such that π has a distance of at most $h = \sum_{i=0}^{L} |D_i|/2$ balancing interchanges from an MB policy.

The above defines subsets of the set Π_{n-1}. The set Π_{n-1} is divided into $K+1$ intervals $\Pi_n^h, 0 \leq h \leq K$ where $\Pi_n^0 = \Pi_n$. Note that all policies that belong to any of these subsets also belong to Π_{n-1}.

2.6.4 Main Result

In this section we present and prove the optimality of MB policies with respect to cost functions $f \in \mathcal{F}$. In the following, $\mathbf{X}^{MB}$ and $\mathbf{X}^{\pi}$ represent the queue sizes under a MB and an arbitrary policy π. For two real-valued random variables A and B, $A \leq_{st} B$ defines the usual stochastic ordering [20].

Theorem 2. *Consider a system of L queues served by K identical servers, as shown in Figure 2.1 with the assumptions of Section 2.1.1. Then a Most Balancing (MB) policy dominates any arbitrary policy when applied to this system, i.e.,*

$$f(\mathbf{X}^{MB}(t)) \leq_{st} f(\mathbf{X}^{\pi}(t)), \quad \forall\ t = 1, 2, \ldots \tag{2.50}$$

for all $\pi \in \Pi$ and all cost functions $f \in \mathcal{F}$.

Proof. From (2.15) and the definition of stochastic dominance, it is sufficient to show that $\mathbf{X}^{MB}(t) \prec_p \mathbf{X}^{\pi}(t)$ for all t and all sample paths in a suitable sample space. The sample space is the standard one used in stochastic coupling methods [21]; see Appendix C for more details.

Denote by $\mathbf{y}^{\pi}(n)$ (resp. $\mathbf{y}^{MB}(n)$) the (feasible) withdrawal vector under policy π (resp. under an MB policy) during time slot n.

To prove the optimality of an MB policy, π^{MB}, we start with an arbitrary policy π and apply a series of modifications that result in a sequence of policies $(\pi_1, \pi_2, \ldots)$. The modified policies have the following properties: (a) π_1 dominates the given policy π, (b) $\pi_i \in \Pi_i$, i.e., policy π_i has the MB property at time slots $t = 1, 2, \ldots, i$, and, (c) π_j dominates π_i for $j > i$ (and thus π_j has the MB property for a longer period of time than π_i). The following lemma is needed to complete the proof of Theorem 2.

Lemma 7. *For any policy $\pi \in \Pi_\tau^h$ and $h > 0$, a policy $\tilde{\pi} \in \Pi_\tau^{h-1}$ can be constructed such that $\tilde{\pi}$ dominates π.*

The proof of Lemma 7 is given in Appendix C.

We now proceed with the proof of Theorem 2. Let π be any arbitrary policy; then $\pi \in \Pi_0 = \Pi_1^K$. We construct a sequence of policies starting from π by applying Lemma 7 repeatedly. Each of these policies dominates the previous one. According to Lemma 7, we obtain policies that belong to $\Pi_1^K, \Pi_1^{K-1}, \ldots, \Pi_1^0 = \Pi_1$. We call the last such policy π_1, and by construction $\pi_1 \in \Pi_2^K$. By recursively continuing in this fashion we construct $\pi_n \in \Pi_n$ for $n = 1, 2, \ldots$. From the construction of π_n, we can see that it satisfies properties (a), (b) and (c) above.

For any value of n, this sequence of policies defines a limiting policy π^* that agrees with π_n until time n. Thus π^* acts similar to π^{MB} at all times and dominates all the previous policies, including π. □

Remark: The optimality of MB policies is intuitively apparent; any such policy

will tend to reduce the probability that any server idles. This is because the MB policies distribute the servers among the longest connected queues in the system and try to keep packets spread in a "uniform" manner among all the queues. The MB policies also outperform a *Longest Connected Queue* (LCQ) policy which assigns all K servers to the longest connected queue at each time slot. □

2.7 The Least Balancing Policies

The *Least Balancing* (LB) policies are the server allocation policies that at every time slot ($n = 1, 2, \ldots$), choose a packet withdrawal vector $\mathbf{y}(n) \in \mathcal{Y}(\mathbf{x}, \mathbf{g})$ that "maximizes the differences" between queue lengths in the system (i.e., maximizes $\kappa_n(\pi)$ in Equation (2.10)). In other words, if Π^{LB} is the set of all LB policies and Π^{WC} is the set of all work conserving policies then

$$\Pi^{LB} = \left\{\pi : \underset{\mathbf{y}(n)\in\mathcal{Y}(\mathbf{x},\mathbf{g})}{\operatorname{argmax}} \kappa_n(\pi), \pi \in \Pi^{WC}, \quad \forall n\right\} \tag{2.51}$$

The maximization of the imbalance index can be achieved by serving packets from the shortest nonempty queues in the system. Such action maximizes the number of empty queues in the system, thus maximizing the chance that servers are forced to idle in future time slots because they are connected to empty queues only. This intuitively suggests that LB policies will be outperformed by any work conserving policy. Furthermore, a non-work conserving policy can by constructed such that it will perform worse than LB policies, e.g., a policy that idles all servers. The next theorem states this fact formally. Its proof is analogous to that of Theorem 2.

Theorem 3. *Consider a system of L queues served by K identical servers, under the assumptions described in Section 2.1.1. A Least Balancing (LB) policy is dominated*

by any arbitrary work-conserving policy when applied to this system, i.e.,

$$f(\mathbf{X}^{\pi}(t)) \leq_{st} f(\mathbf{X}^{LB}(t)), \quad \forall\, t = 1, 2, \ldots \tag{2.52}$$

for all $\pi \in \Pi^{WC}$ *and all cost functions* $f \in \mathcal{F}$.

An LB policy has no practical significance, since it maximizes the cost functions presented earlier. Intuitively, it should also minimize the system stability region and hence the system throughput. However, it is interesting to study the worst possible policy behavior and to measure its performance. The LB and MB policies provide lower and upper limits to the performance of any work-conserving policy. Furthermore, the performance of any policy can be measured by the deviation of its behavior from that of the MB and LB policies.

2.8 MB and LB Approximate Implementation Algorithms

In this section, we present two policies that approximate the behavior of the MB and LB policies respectively. We present a feasible implementation algorithm for each of the two policies. These algorithms can be used as approximate implementations for the MB and LB policies in practical and simulated systems.

2.8.1 Approximate Implementation of MB Policies

We introduce the *Least Connected Server First/Longest Connected Queue* (LCSF/LCQ) policy, a low-overhead approximation of an MB policy, with $O(L \times K)$ computational complexity. We show that it results in a feasible withdrawal vector. The policy is stationary and depends only on the current state $(\mathbf{X}(n), \mathbf{G}(n))$ during

time slot n.

The LCSF/LCQ implementation during a given time slot is described as follows: The least connected server is identified and is allocated to its longest connected queue. The queue length is updated (i.e., decremented). We proceed accordingly to the next least connected server until all servers are assigned. In algorithmic terms, the LCSF/LCQ policy can be described/implemented as follows:

Recall that $\mathbb{Q}_j$ denotes the set of all queues that are connected to server j at time slot t. Let $\mathbb{Q}_{[i]}$ be the i^{th} element in the sequence $(\mathbb{Q}_1, \ldots, \mathbb{Q}_K)$, when ordered in ascending manner according to their size (set cardinality), i.e., $|\mathbb{Q}_{[l]}| \geq |\mathbb{Q}_{[m]}|$ if $l > m$. Ties are broken arbitrarily. Then under the LCSF/LCQ policy, the K servers are allocated according to the following algorithm:

Algorithm 2 (LCSF/LCQ Implementation)**.**

1. $for \quad t = 1, 2, \ldots \quad do \quad \Big\{$
2. $\quad$ *Input:* $\mathbf{X}(t), \mathbf{G}(t)$. *Calculate* $\quad \mathbb{Q}_{[l]}$, $l = 1, \ldots, K$.
3. $\quad \mathbf{X}' \longleftarrow \mathbf{X}(t),\ \mathbf{Y} \longleftarrow \mathbf{0},\ \mathbf{Q} \longleftarrow \mathbf{0}$
4. $\quad for\ j\ =\ 1 \quad to \quad K \quad \Big\}$ *; allocate servers sequentially*
5. $\quad\quad Q_{[j]} = \min\left(l : l \in \left\{ \underset{k:k\in\mathbb{Q}_{[j]}}{\operatorname{argmax}}(X'_k | X'_k > 0) \right\} \right)$
6. $\quad\quad for\ i\ =\ 1 \quad to \quad L \quad \Big\{$
7. $\quad\quad\quad Y_i = Y_i + \mathbb{1}_{\{i=Q_{[j]}\}}$
8. $\quad\quad\quad X'_i = X_i(t) - Y_i \quad \Big\} \quad \Big\}$
9. $\quad$ *Output:* $\mathbf{y}(t) \longleftarrow \mathbf{Y}, \mathbf{q}(t) \longleftarrow \mathbf{Q} \quad$ *; report outputs*
10. $\Big\} \quad$ *; End of Algorithm 2.*

Note that in line 5 of Algorithm 2, if the set $\mathbb{Q}_{[j]}$ is empty, then the argmax returns

the empty set. In this case, the j^{th} order server will not be allocated (i.e., will be idle during time slot t). Algorithm 2 produces two outputs, when it is run at $t = n$: $\mathbf{y}(n)$ and $\mathbf{q}(n)$ as shown in line 9 of the algorithm. In accordance to the definition of a policy in Equation (2.9), the LCSF/LCQ policy can be formally defined as the sequence of time-independent mappings $u(\mathbf{x}(n), \mathbf{g}(n))$ that produce the withdrawal vector $\mathbf{y}(n)$ described in line 9 above. The following lemma asserts that the mapping defines feasible controls.

Lemma 8. *The policy obtained from applying Algorithm 2 results in a feasible withdrawal vector at every time slot n and any state $(\mathbf{x}(n), \mathbf{g}(n))$.*

Proof. Let $\mathbf{y}(n)$ and $\mathbf{q}(n)$ denote the outputs of Algorithm 2; the inputs are $(\mathbf{x}(n), \mathbf{g}(n))$. Let $\mathbf{V}(n)$ be the matrix with elements:

$$V_{ij}(n) = \mathbb{1}_{\{i=q_j(n)\}} \cdot g_{i,j}(n). \tag{2.53}$$

We must show that the output $\mathbf{y}(n)$ can be written as

$$\mathbf{y}(n) = \mathbf{V}(n) \cdot \mathbf{I}_K \tag{2.54}$$

and that $\mathbf{V}(n)$ satisfies the feasibility constraints (2.2) and (2.3).

From Algorithm 2, line 5, it can be seen that for every server $[j]$, only the set of queues that are connected to server $[j]$ are considered as candidates for allocating this server. Therefore, $V_{i[j]}(n) = 1$ is true only when $g_{i,[j]}(n) = 1$ and $\mathbb{1}_{\left\{i=q_{[j]}(n)\right\}} = 1$ are true, establishing Equation (2.53). From Equations (2.54) and (2.3) we can easily see that

$$\mathbf{y}(n) \leq \mathbf{x}(n) \tag{2.55}$$

is a sufficient condition for Inequality (2.3) to hold. Note that queue i will be selected in Algorithm 2, line 5 (to be served by server $[j]$) only if its current size X_i' is strictly

positive. This will ensure that the number of servers allocated to any queue is no larger than the number required to empty that queue. Therefore, $y_i(n) \leq x_i(n), i = 1, \ldots, L$, proving Inequality (2.55).

Constraints (2.2) are satisfied. To prove that, fix a server $[j]$; the initialization in step 3 assigns this server to the dummy queue. Observe that even though the inner for-loop in Algorithm 2 is executed $L + 1$ times, the indicator function $\mathbb{1}_{\{i=Q_{[j]}\}}$ in line 7 is nonzero for only one value of $i \in \{0, 1, \ldots, L\}$; each server is allocated to one queue only, either the dummy queue or the queue with the minimum index out of the outcome of the argmax function in line 5 of Algorithm 2. Therefore the statement

$$\sum_{i=0}^{L} \mathbb{1}_{\{i=q_{[j]}(t)\}} = 1$$

is true for all j, proving equality (2.2). □

Although allocating the available servers to their longest connected queues in the order specified by Algorithm 2 may not be "most balancing" in some occasions, the LCSF/LCQ is expected to perform very close to any MB policy.

Lemma 9. *LCSF/LCQ is not an MB policy.*

Proof. To prove lemma 9 we present the following counter example. Consider a system with $L = 4$ and $K = 7$. At time slot n the system has the following configuration:

The queue state at time slot n is $\mathbf{x}(n) = (5, 5, 5, 4)$. Servers 1 to 6 are connected to queues 1, 2 and 3 and server 7 is connected to queues 1 and 4 only.

Under this configuration, we can show that the LCSF/LCQ algorithm will result in $\hat{\mathbf{x}}(n) = (0, 2, 3, 3, 4)$ (where the first element represents the dummy queue that by assumption holds no real packets) and $\kappa_n(LCSF/LCQ) = 18$. A policy π can be constructed that selects the feasible server allocation $\mathbf{q} = (1, 2, 3, 1, 2, 3, 4)$ which yields the state $\hat{\mathbf{x}}(n) = (0, 3, 3, 3, 3)$ and $\kappa_n(\pi) = 12 < \kappa_n(LCSF/LCQ)$. Therefore, the LCSF/LCQ does not belong to the class of MB policies. □

The LCSF/LCQ policy is of particular interest for the following reasons: (a) It follows a particular server allocation ordering (LCSF) to their longest connected queues (LCQ) and thus it is closely related to Algorithm 1, (b) the selected server ordering (LCSF) and allocation (LCQ) intuitively attempt to maximize the opportunity to target and reduce the longest connected queue in the system thus minimizing the imbalance among queues, and (c) as we will see in Section 2.9, the LCSF/LCQ performance is statistically indistinguishable from that of an MB policy (implying that the counterexamples similar to the one in Lemma 9 proof has low probability of occurrence under LCSF/LCQ system operation).

2.8.2 Approximate Implementation of LB Policies

In this section, we present the MCSF/SCQ policy as a low complexity, easy-to-implement approximation of LB policies. We also provide an implementation algorithm for MCSF/SCQ using the sequential server allocation principle that we used in the previous algorithms.

The *Most Connected Server First/Shortest Connected Queue* (MCSF/SCQ) policy is the server allocation policy that allocates (in a sequential manner) each one of the K servers to its shortest connected queue (not counting the packets already scheduled for service) starting with the most connected server first.

The MCSF/SCQ implementation algorithm is analogous to Algorithm 2 except for lines 4 and 5 which are described next:

Algorithm 3 (MCSF/SCQ Implementation)**.**

1. $for \quad t = 1, 2, \ldots \quad do \quad \Big\{$

$\vdots$

4. $\quad for\ j = K \quad to \quad 1 \quad \Big\}$ *; Servers in descending order*

5. $\quad\quad Q_{[j]} = \min\left(l : l \in \left\{ \underset{k:k\in\mathbb{Q}_{[j]}}{\operatorname{argmin}}(X'_k | X'_k > 0) \right\}\right)$

$\vdots$

10. *; End of Algorithm 3.*

Comments analogous to those valid for Algorithm 2 are also valid for Algorithm 3.

2.9 Performance Evaluation and Simulation Results

We used simulation to study the performance of the system under these two policies and to compare against the system performance under several other policies. The metric we used in this study is $EQ \triangleq E(\sum_{i=1}^{L} X_i)$, the average of the total number of packets in the system.

We focused on two groups of simulations. In the first, we evaluate the system performance with respect to number of queues (L) and servers (K) as well as channel connectivity (Figures 2.3 to 2.11). Random arrivals to queues are assumed to be i.i.d. Bernoulli. In the second group of simulations (Figures 2.12 to 2.14) we consider batch arrivals with random (uniformly distributed) burst size.

The policies used in this simulation are: LCSF/LCQ, as an approximation of an MB policy; MCSF/SCQ, as an approximation of an LB policy. An MB policy was

implemented following Algorithm 1 and its performance was indistinguishable from that of the LCSF/LCQ. Therefore, in the simulation graphs the MB and LCSF/LCQ are represented by the same curves. Other policies that were simulated include the randomized, Most Connected Server First/Longest Connected Queue (MCSF/LCQ), and Least Connected Server First/Shortest Connected Queue (LCSF/SCQ) policies. The randomized policy is the one that at each time slot allocates each server, randomly and with equal probability, to one of its connected queues. The MCSF/LCQ policy differs from the LCSF/LCQ policies in the order that it allocates the servers. It uses the exact reverse order, starting the allocation with the most connected server and ending it with the least connected one. However, it resembles MB policies in that it allocates each server to its longest connected queue. The LCSF/SCQ policy allocates each server, starting from the one with the least number of connected queues, to its shortest connected queue. The difference from an MB policy is obviously the allocation to the shortest connected queue. This policy will result in greatly unbalanced queue lengths and hence a performance that is closer to the LB policies.

Figure 2.3 shows the average total queue occupancy versus arrival rate under the five different policies. The system in this simulation is a symmetrical system with 16 parallel queues ($L = 16$), 16 identical servers ($K = 16$) and i.i.d. Bernoulli queue-to-server (channel) connectivity with parameter $p = P[G_{i,j}(t) = 1] = 0.2$.

The curves in Figure 2.3 follow a shape that is initially almost flat and ends with a rapid increase. This abrupt increase happens at a point where the system crosses a "stability threshold". In this case, the queue lengths in the system will grow fast and the system becomes unstable. The graph shows that LCSF/LCQ, the MB policy approximation outperforms[4] all other policies. It minimizes EQ and hence the queuing delay. We also noticed that it maximizes the system stability region and

[4]99% confidence intervals are very narrow and would affect the readability of the graphs and therefore are not included.

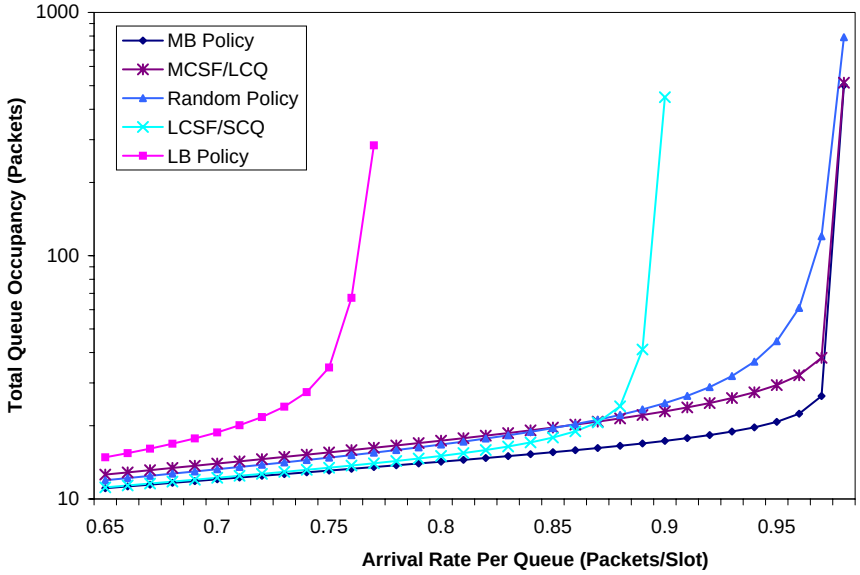


Figure 2.3: Average total queue occupancy, EQ, versus load under different policies, $L = 16, K = 16$ and $p = 0.2$.

hence the system throughput as well. As expected, the performance of the other three policies lies within the performance of the MB and LB policies.

The MCSF/LCQ and LCSF/SCQ policies are variations of the MB and LB policies respectively. The performance of MCSF/LCQ policy is close to that of the MB policy. The difference in performance is due to the order of server allocation. On the other hand, the LCSF/SCQ policy shows a large performance improvement on that of the LB policy. This improvement is a result of the reordering of allocations of servers.

Figure 2.3 also shows that the randomized policy performs reasonably well. Moreover, its performance improves as the number of servers in the system decreases, as the next set of experiments shows.

2.9.1 The Effect of The Number of Servers

In this section, we study the effect of the number of servers on policy performance. Figures 2.4 ($K = 8$) and 2.5 ($K = 4$) show EQ versus arrival rate per queue under the five policies, in a symmetrical system with $L = 16$ and $p = 0.2$. Comparing these

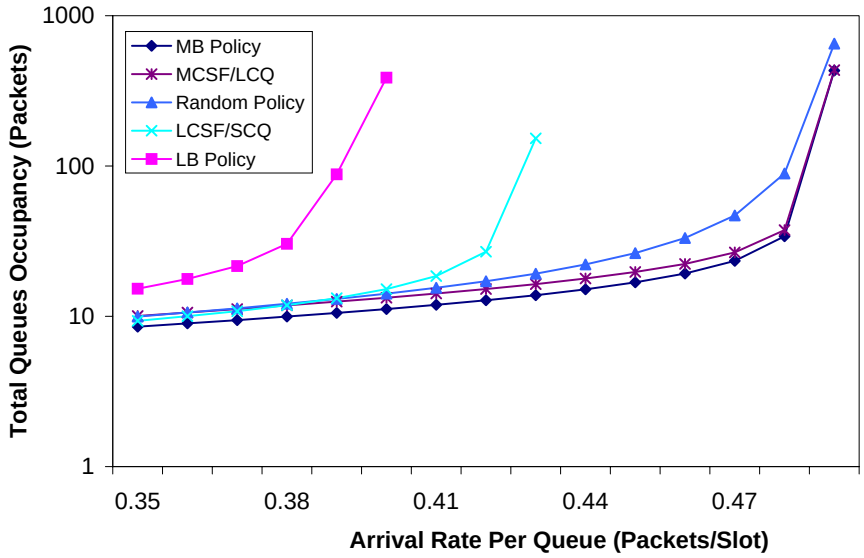


Figure 2.4: Average total queue occupancy, EQ, versus load, $L = 16, K = 8$ and $p = 0.2$.

two graphs to the one in Figure 2.3, we notice the following:

First, the performance advantage of the LCSF/LCQ (an hence of an MB policy) over the other policies increases as the number of servers in the system increases. The presence of more servers implies that the server allocation action space is larger. Selecting the optimal (i.e., MB) allocation, over any arbitrary policy, out of a large number of options will produce better performance as compared to the case when the number of server allocation options is reduced.

Second, the stability region of the system becomes narrower when less servers are used. This is true because fewer resources (servers) are available to be allocated by the working policy in this case.

Finally, we notice that the MCSF/LCQ performs very close to the LCSF/LCQ policy in the case of $K = 4$. Apparently, when K is small, the order of server allocation does not have a big impact on the policy performance.

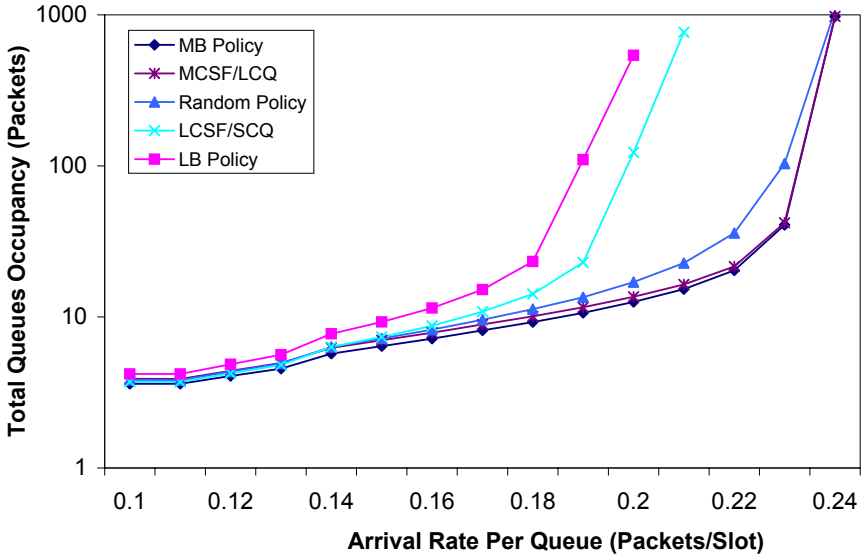


Figure 2.5: Average total queue occupancy, EQ, versus load, $L = 16, K = 4$ and $p = 0.2$.

2.9.2 The Effect of Channel Connectivity

In this section we investigate the effect of channel connectivity on the performance of the five policies. Figures 2.6 and 2.11 show this effect under two different setups with respect to L and K. We can make the following observations:

First, we notice that for larger channel connection probabilities ($p \geq 0.9$), the effect of the policy behavior on the system performance becomes less significant. Therefore, the performance difference among the various policies is getting smaller. The LCSF/LCQ policy still has a small advantage over the rest of the policies, even though statistically indistinguishable. MCSF/SCQ continues to have the worst performance. As p increases, the probability that a server will end up connected to a group of empty queues will be very small regardless of the policy in effect. In fact, when the servers have full connectivity to all queues (i.e., $p = 1.0$) we expect that any work conserving policy will minimize the total number of packets in a symmetrical homogeneous system of queues. Therefore, any (work-conserving) policy will be optimal in a system with full connectivity.

Second, from all graphs we observe that there is a maximum input load that results in a stable system operation (maximum stable throughput). An upper bound (for stable system operation) for the arrival rate α for each queue can be shown to be

$$\alpha < \frac{K}{L}\big(1 - (1-p)^L\big) \tag{2.56}$$

In other words, the average number of packets entering the system (αL) must be less than the rate they are being served. When $p = 1.0$, the stability condition in Inequality (2.56) will be reduced to $\alpha L < K$, which makes intuitive sense.

Finally, we noticed that the MCSF/LCQ policy performs very close to the LCSF/LCQ policy. However, its performance deteriorates in systems with higher number of servers and lower channel connectivity probabilities. It is intuitive that with more servers available, the effect of the order of server allocations on performance will increase. Since MCSF/LCQ differs from LCSF/LCQ only by the order of server allocation, therefore, more servers implies larger performance difference. Also, the lower the connectivity probability, the higher the probability that a server will end up with no connectivity to any nonempty queue, and hence be forced to idle.

2.9.3 Batch Arrivals With Random Batch Size

We studied the performance of the presented policies in the case of batch arrivals with uniformly distributed batch size, in the range $\{1, \ldots, u\}$. Figures 2.12–2.14 show EQ versus load for three cases with $u = 2, 5, 10$, and hence average batch sizes 1.5, 3, and 5.5. The MB policy clearly dominates all the other policies. However, the performance of all the policies (including the MB policy) deteriorates when the arrivals become burstier, i.e., the batch size increases. The performance of the other policies, including the LB policy approaches that of the LCSF/LCQ policy as the average batch size increases.

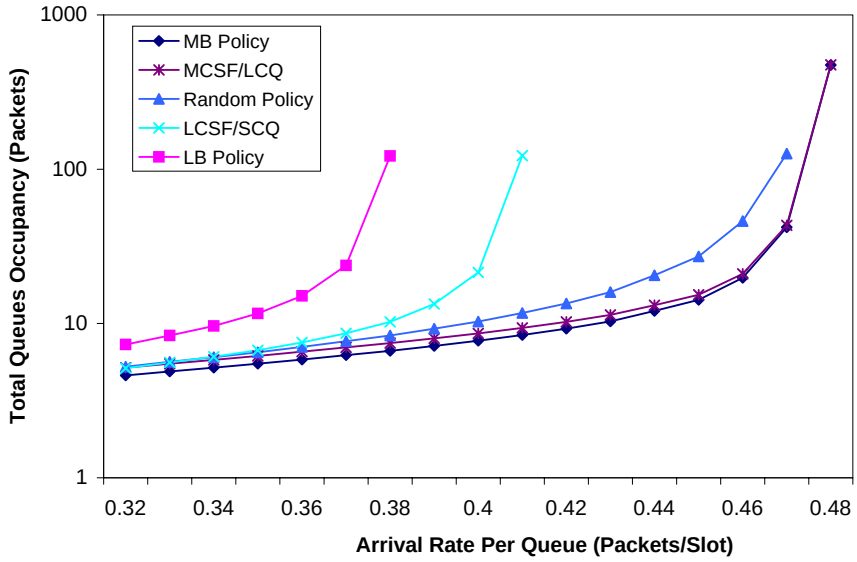


Figure 2.6: Average total queue occupancy, EQ, versus load under different policies, $L = 8$, $K = 4$ and $p = 0.3$.

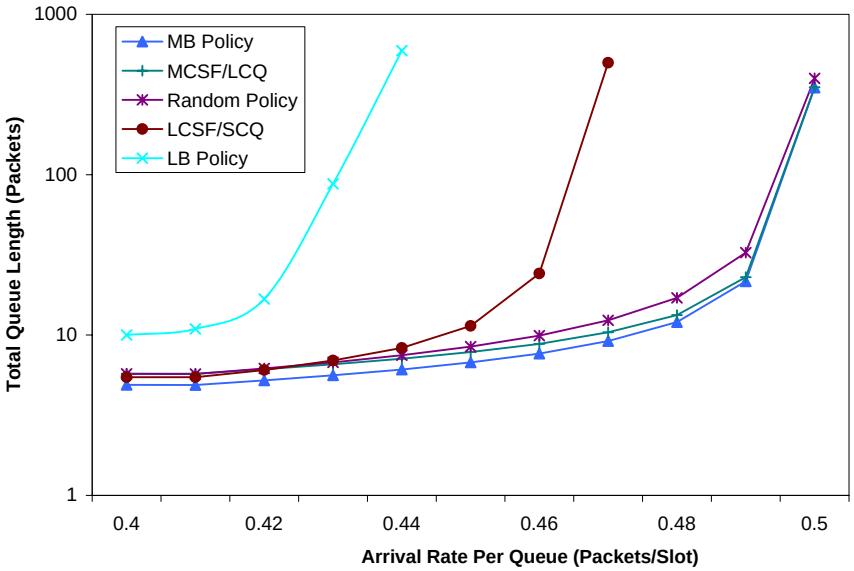


Figure 2.7: Average total queue occupancy, EQ, versus load under different policies, $L = 8$, $K = 4$ and $p = 0.5$.

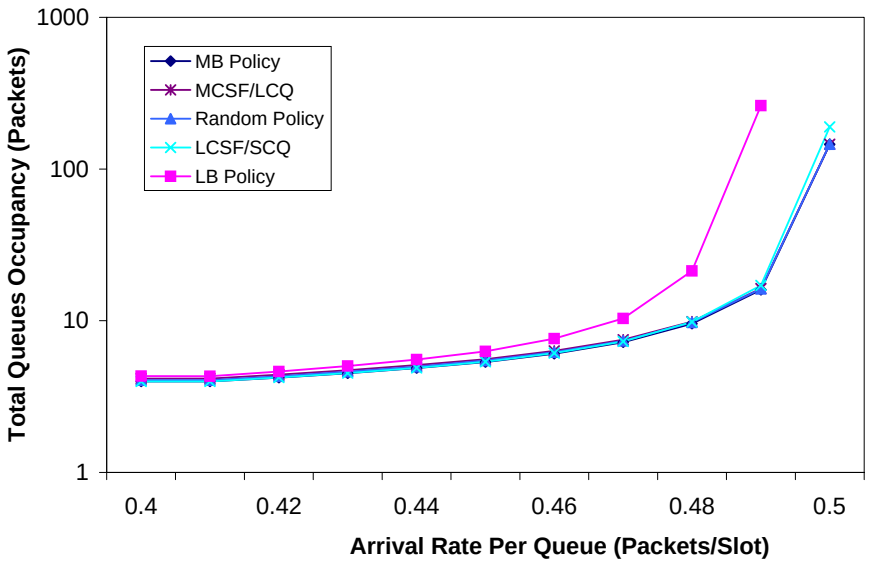


Figure 2.8: Average total queue occupancy, EQ, versus load under different policies, $L = 8$, $K = 4$ and $p = 0.9$.

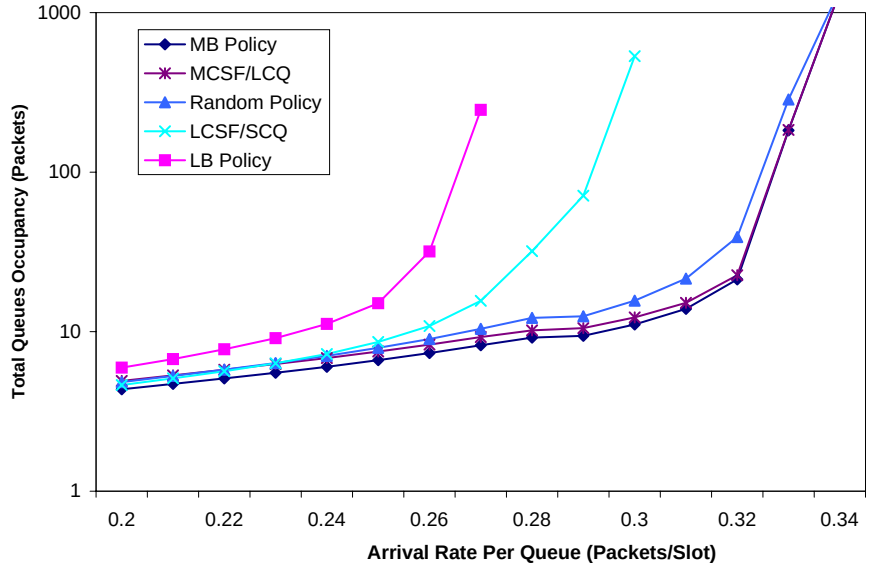


Figure 2.9: Average total queue occupancy, EQ, versus load under different policies, $L = 12$, $K = 4$ and $p = 0.3$.

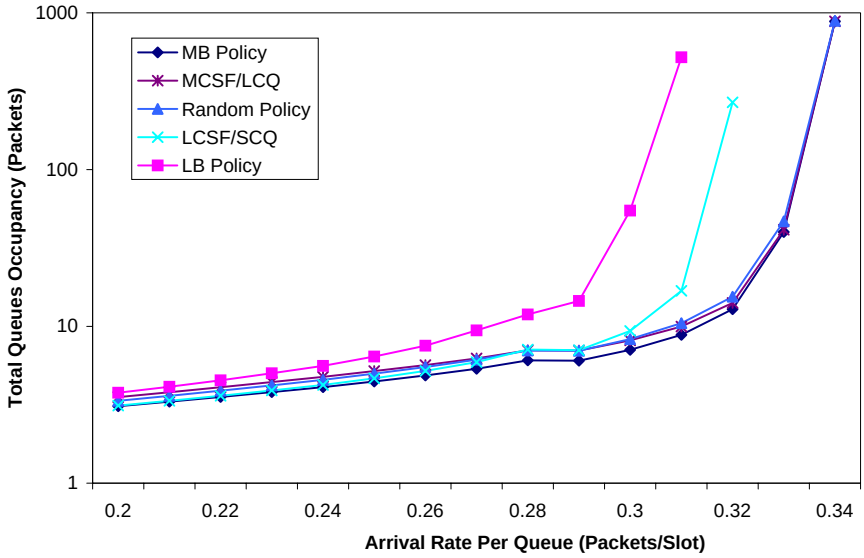


Figure 2.10: Average total queue occupancy, EQ, versus load under different policies, $L = 12$, $K = 4$ and $p = 0.5$.

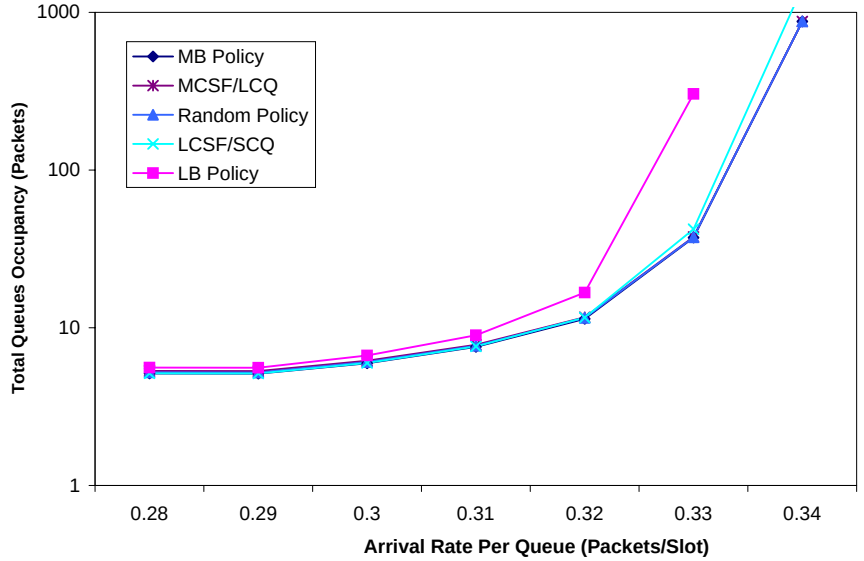


Figure 2.11: Average total queue occupancy, EQ, versus load under different policies, $L = 12$, $K = 4$ and $p = 0.9$.

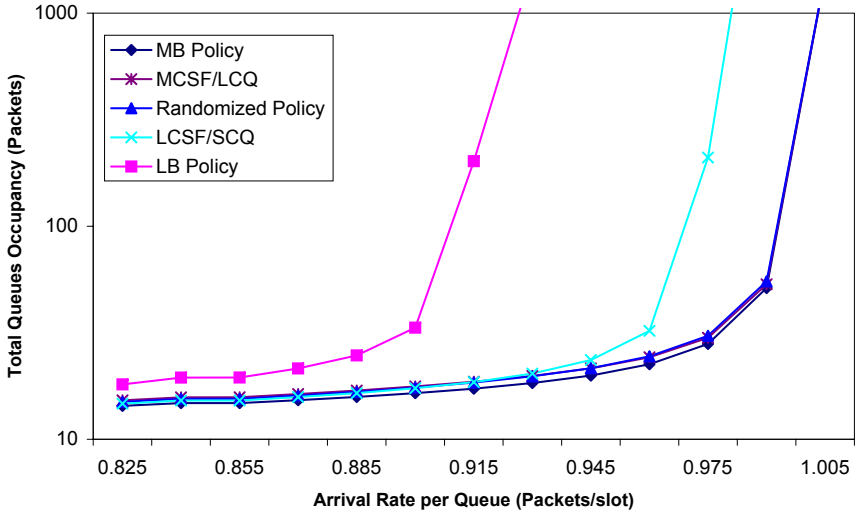


Figure 2.12: Average total queue occupancy, EQ, versus load, batch arrivals, $L = 16$, $K = 16$ and $p = 0.3$; batch size =U(2).

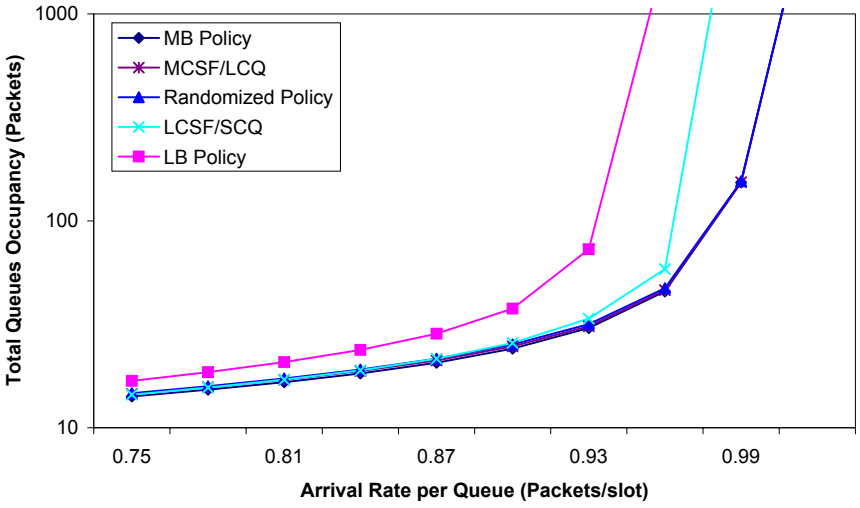


Figure 2.13: Average total queue occupancy, EQ, versus load, batch arrivals, $L = 16$, $K = 16$ and $p = 0.6$; batch size =U(5).

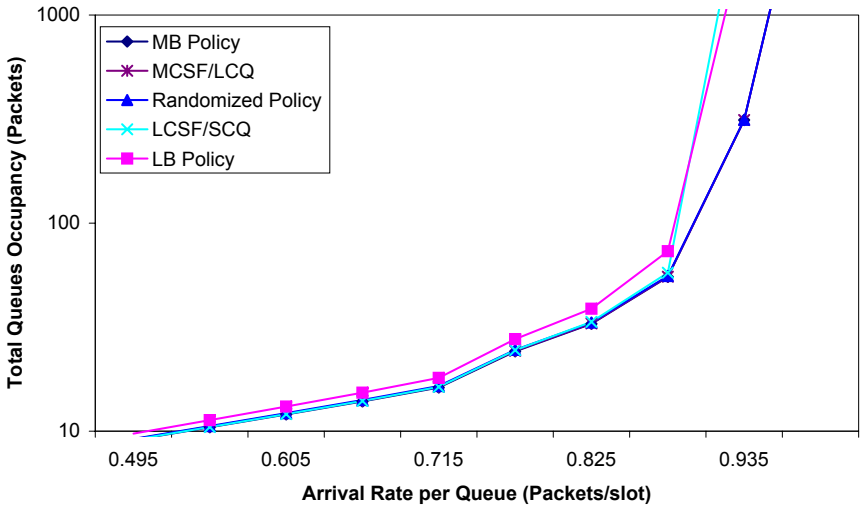


Figure 2.14: Average total queue occupancy, EQ, versus load, batch arrivals, $L = 16$, $K = 16$ and $p = 0.8$; batch size =U(10).

2.9.4 Correlated Arrivals

The graphs in Figures 2.15–2.17 show the effect of correlation in the arrivals to each queue (the arrivals to different queues still being i.i.d.) on the different policies performance. The correlation was generated in the simulation using the following technique:

We define the random sequence $U_n, n = 0, 1, \ldots$ where the initial state U_0 is a uniformly distributed random variable in $[0, 1]$, i.e., $U_0 \sim U[0, 1]$. Let $V_n \sim U[-c_i, c_i], \forall n = 0, 1, \ldots$ be another uniformly distributed random sequence, where $0 < c_i \leq 0.5$ is the correlation interval. Then U_{n+1} is given by

$$U_{n+1} =< U_n + V_n >$$

where $< \mathcal{A} >$ returns $|\mathcal{A} \mod 1|$. The amount of correlation in this case depends on the value of c_i. The larger this value is the less the correlation between the elements of the resulted sequence. If $c_i = 0.5$ then there is no correlation between the sequence

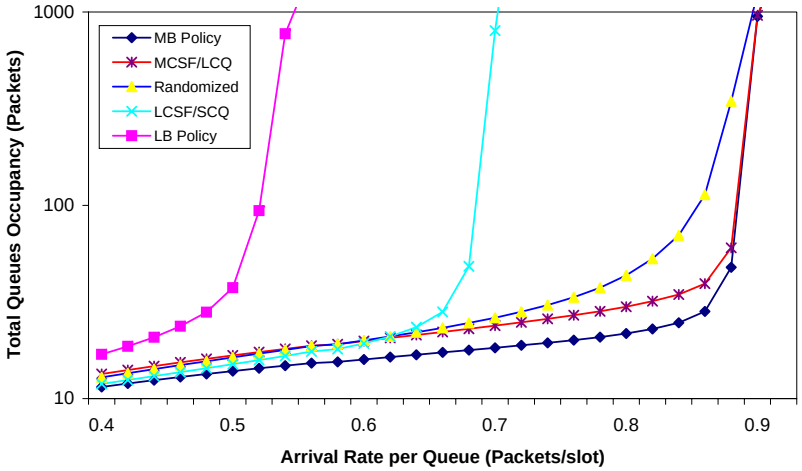


Figure 2.15: Average total queue occupancy, EQ, versus load, correlated arrivals, $L = 16$, $K = 16$ and $p = 0.2$; correlation interval =0.2.

elements. The random variables $V_n, \forall n = 0, 1, \ldots$ are computed as follows:

$$Vn = (2 \cdot v_n - 1) \cdot c_i \tag{2.57}$$

where $v_n \sim U[0, 1]$ is a uniformly distributed random variable that is generated for every time n. Now let α_i be the arrival rate to queue i, and let the arrival process to queue i be the following

$$Z_i(n) = \mathbb{1}_{\{U_n \geq \alpha_i\}}, \quad \forall n = 1, 2, \ldots \tag{2.58}$$

In this case the arrivals to queue i will be correlated in time. The amount of correlation between arrivals in consecutive time slots depends on the parameter c_i.

The simulation results show that the higher correlation has adverse effect on the performance of all policies. As Figures 2.15–2.17 show, for higher correlation ($c_i = 0.05$) the system stability region is reduced and the total number of packets in the system is increased compared to the other cases (i.e., when $c_i = 0.1$ and $c_i = 0.2$). In all cases the simulation results show that an MB policy (namely the LCSF/LCQ) still outperforms the other policies under study. They also show that an LB policy (MCSF/SCQ) performs the worst among the policies we studied.

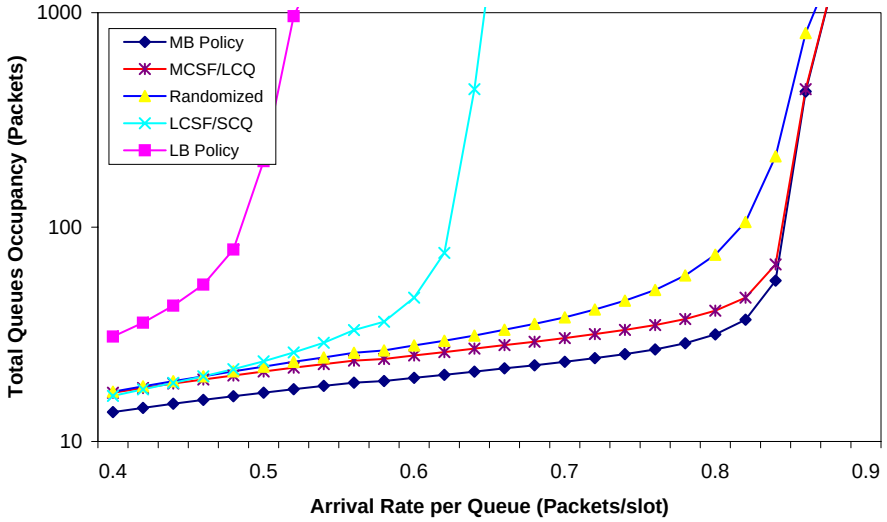


Figure 2.16: Average total queue occupancy, EQ, versus load, correlated arrivals, $L = 16$, $K = 16$ and $p = 0.2$; correlation interval =0.1.

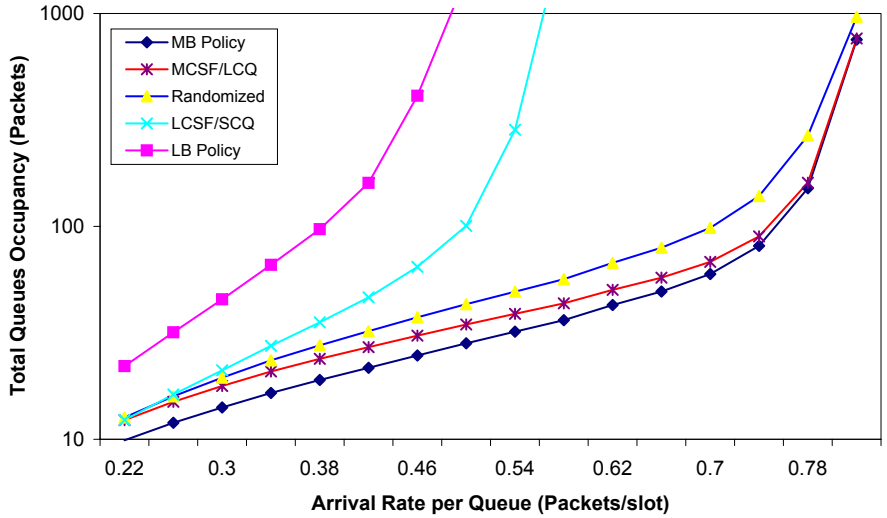


Figure 2.17: Average total queue occupancy, EQ, versus load, correlated arrivals, $L = 16$, $K = 16$ and $p = 0.2$; correlation interval =0.05.

Chapter 3

Optimal Multi-Server Allocation to Parallel Queues With Random Connectivity and Retransmissions

In this chapter, we investigate an optimal scheduling problem for a discrete-time system of two parallel queues with infinite capacity, sharing two symmetrical servers. This model can be used to study a variety of scheduling problems in wireless networks. At any time slot, a queue can be served by one or two connected servers; the queue-server connectivity is assumed to be random and modeled by a two-state Markov chain. The arrivals to each queue are assumed to be independent and identically distributed. A scheduled packet completes service successfully with a given probability. Otherwise, it has to be retransmitted in a later time slot. The optimal scheduling policy is defined as the server allocation policy that minimizes, in a stochastic ordering sense, the total number of packets in the system. We prove, using a dynamic coupling method, that a "Most Balancing" policy, a policy that attempts to balance the lengths of the two queues at every time slot, is optimal. We also compare the performance of the optimal policy to that of a few other policies via simulations.

3.1 Introduction

In this chapter, we investigate an optimal queuing control problem similar to the one in Chapter 2 with the addition of preemptive service completion and the ability of retransmission. In such models, time is slotted in constant length slots. The packets in this system are assumed to have constant length, equal to one time slot. In order to capture noise, interference, etc. effects, models with *retransmissions* assume that a scheduled packet completes its service with a given probability. The packet has to be retransmitted if it does not complete service.

The objective of this work is to identify the optimal server allocation policy that minimizes, in a stochastic sense, the total queue occupancy (and hence queuing delay) in a queuing system with two servers, random queue-server connectivity and retransmissions. Our main contribution is to prove, using stochastic coupling argument, that a Most Balancing policy, a policy that attempts to balance the lengths of the two queues (defined precisely in Section 2.4), is optimal.

3.2 Model Description

We consider the system of two queues served by two servers, as depicted in Figure 3.1. We define the following random sequences to be used in this chapter:

- $\mathbf{X_i} = \{X_i(t)\}_{t=1}^{\infty}$, where $X_i(t), i = 1, 2$, is the length of queue i at time t.
- $\mathbf{Z_i} = \{Z_i(t)\}_{t=1}^{\infty}$, where $Z_i(t), i = 1, 2$, is the number of exogenous arrivals to queue i at time t.
- $\mu_\mathbf{j} = \{\mu_j(t)\}_{t=1}^{\infty}, j \in \{1, 2\}$, where $\mu_j(t) = 1$ (resp. 0) denotes that a packet served by server j completes (resp. does not complete) service at time t.

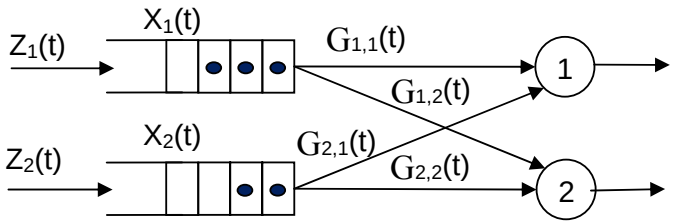


Figure 3.1: The queuing model under consideration.

- $\mathbf{G_{i,j}} = \{G_{i,j}(t)\}_{t=1}^{\infty}, i, j \in \{1, 2\}$, where $G_{i,j}(t) = 1$ (resp. 0) denotes that queue i is connected (resp. not connected) to server j at time t.

- $\mathbf{V_{i,j}} = \{V_{i,j}(t)\}_{t=1}^{\infty}, i, j \in \{1, 2\}$, where $V_{i,j}(t) = 1$ (resp. 0) if server j is serving (resp. not serving) a packet from queue i during time slot t. $\mathbf{V_{i,j}}$ represents the control actions taken by the scheduling policy (to be defined precisely in the next section).

- $\mathbf{W} = \{W(t)\}_{t=1}^{\infty}$, where $W(t) = \sum_{i=1}^{2} X_i(t)$ is the total number of packets in the system at time t.

Upon successful completion of service, the packet is removed from (the head of) its queue; otherwise, the packet remains at the head of its queue and will be retransmitted at a later time slot, when a connected server is allocated to its queue. Note that the system can serve up to two packets (that belong to one or both queues) at each time slot.

We make the following statistical assumptions. The arrival sequences $(\mathbf{Z_i}, i = 1, 2)$ are assumed to be i.i.d. Bernoulli, with parameter $P[Z_i(t) = 1] = \alpha$, independent of i. For fixed i, j, the connectivity sequence $\mathbf{G_{i,j}}$ is assumed to be a 2-state Markov chain, whose transition probabilities do not depend on i, j. The packet service completion sequences $\mu_\mathbf{j}$ are assumed to be i.i.d. Bernoulli, with $P[\mu_j(t) = 1] = P_{sc}$, independent of j. It is further assumed that the connectivity, arrival and service completion sequences are independent of each other.

Let $[e]^+$ equal e if $e \geq 0$ and 0 otherwise. Assuming unlimited buffer capacity, the length of queue i evolves according to the following equation for all $t \in T = \{1, 2, \ldots\}$

$$X_i(t) = \left[X_i(t-1) - \sum_{j=1}^{2} V_{i,j}(t) \cdot G_{i,j}(t) \cdot \mu_j(t) \right]^+ + Z_i(t) \tag{3.1}$$

By convention and during any time slot t, events in this system happen in the following order: the scheduler observes the channel state $G_{i,j}(t)$. It sets the values of $V_{i,j}(t)$ depending on the information it has about the history of the system state. If a scheduled packet service (by server j) is completed (i.e., if $\mu_j(t) = 1$) then the packet is removed from the system, otherwise it remains in its respected queue. Arrivals are added after packet removals. With this interpretation, Equation (3.1) captures the operations of a wireless packet transmission network.

3.3 Optimal Server Allocation Policies

In the system depicted in Figure 3.1, a packet scheduling policy π is a rule that at each time slot t determines how servers are allocated to queues. More precisely, a policy π determines the value of the four variables $V_{i,j}(t), i = 1, 2,\ j = 1, 2$. We assume that at the beginning of each time slot, the policy has complete knowledge of the history of the "system state" (i.e., the queue connectivity and lengths) up to time t.

Let $\mathbb{A} = \{(0,0), (0,1), (1,0), (1,1), (0,2), (2,0)\}$. Denote by $\pi(t) = (a, b)$ the fact that at time slot t the policy π allocates $a = \sum_{j=1}^{2} V_{1,j}(t)$ servers to queue 1 and $b = \sum_{j=1}^{2} V_{2,j}(t)$ servers to queue 2. We say that the policy π is feasible if it "does not allocate more than the system capacity", i.e., if

$$\sum_{i=1}^{2} \sum_{j=1}^{2} V_{i,j}(t) \leq 2, \qquad \forall t \in T \tag{3.2}$$

We call the set Π, of all policies that satisfy Inequality (3.2), the set of all feasible policies, over which we will determine the optimal one.

3.3.1 Definition of Most Balancing Policies

A "Most Balancing" (MB) policy, π^{MB}, is any (feasible) work-conserving packet scheduling policy that at every time slot t selects a control that minimizes the difference between the two queue lengths. Let $\hat{X}_i(t) = X_i(t) - \sum_{j=1}^{2} V_{i,j}(t)$ denote the length of queue i after subtracting the number of servers allocated to queue i under the policy π. Let Π^{WC} be the set of all work conserving packet scheduling policies, then π^{MB} can be defined via[1]:

$$\pi^{MB} = \underset{\pi \in \Pi^{WC}}{\operatorname{argmin}} \left| \hat{X}_1(t) - \hat{X}_2(t) \right| \tag{3.3}$$

We can easily check that π^* is a MB policy if it allocates the available servers to the two longest *connected* queues at time slot t as follows:

$$\pi^*(t) = \begin{cases} (1,1) & \text{if } X_1(t-1) = X_2(t-1) > 0, \\ (2,0) & \text{if } X_1(t-1) > X_2(t-1) > 0, \\ (0,2) & \text{if } X_2(t-1) > X_1(t-1) > 0. \end{cases} \tag{3.4}$$

When a server is connected to one nonempty queue only then the server is allocated to that queue. When any server is disconnected then idling it is the only feasible action.

Let $\mathbf{W_{MB}}, \mathbf{W}_\pi$ denote the sequences of total number of packets in the system under policies π^{MB} and π respectively. The optimal policy that we investigate in this section is the policy that minimizes the total number of packets in the system in a

[1] $\hat{X}_i(t)$ depends on the policy π through the control $V_{i,j}(t)$.

stochastic ordering sense (the reader may refer to [20] for the definition of stochastic ordering). Minimizing this sequence will also minimize the average queuing delay according to Little's law and since stochastic ordering implies the ordering of the expected values. The following theorem states the optimality of the MB policy among all policies that belong to the class Π. The full proof of Theorem 4 is provided in Section 3.4.

Theorem 4. *An MB policy minimizes, in the stochastic ordering sense, the sequence of total number of packets in the system, i.e.,*

$$\mathbf{W_{MB}} \leq_{st} \mathbf{W}_{\pi}, \qquad \forall \pi \in \Pi. \tag{3.5}$$

3.3.2 Implementation Algorithm for MB Policies

The definition of an MB policy in Equation (3.3) is not constructive; moreover, note that it does not guarantee uniqueness of MB policies. For the system presented in Section 3.2, the Least Connected Server First/Longest Connected Queue (LCSF/LCQ) algorithm allocates the *least* connected server to its LCQ, updates the queue lengths and allocates the other connected server to its LCQ (refer to Section 2.8.1). We can easily verify that Equation (3.4), and hence Equation (3.3), is satisfied by the LCSF/LCQ algorithm. Therefore, LCSF/LCQ is an exact implementation of an MB policy, with low computational complexity compared to brute force search through the complete set of feasible policies Π.

3.3.3 Least Balancing Policies

We define a "Least Balancing" (LB) policy, π^{LB}, as the work-conserving packet scheduling policy that at every time slot selects a control that maximizes the difference between the two queues, i.e.,

$$\pi^{LB} = \underset{\pi \in \Pi^{WC}}{\operatorname{argmax}} \left| \hat{X}_1(t) - \hat{X}_2(t) \right| \tag{3.6}$$

Intuitively, this policy increases the probability that the scheduler will end up with empty connected queue(s) and hence be forced to idle. Let $\mathbf{W_{LB}}$ denote the sequence of total number of packets in the system under π^{LB}. This policy performs the worst among all work-conserving policies for the scheduler model presented in the previous section. The following theorem states this result formally:

Theorem 5. *Among all work conserving policies, the LB policy maximizes, in the stochastic ordering sense, the sequence of total number of packets in the system, i.e.,*

$$\mathbf{W}_\pi \leq_{st} \mathbf{W_{LB}}, \qquad \forall \pi \in \Pi^{WC}. \tag{3.7}$$

The proof of Theorem 5 is analogous to that of Theorem 4.

We present the Most Connected Server First/Shortest Connected Queue (MCSF/SCQ) algorithm as an implementation of an LB policy. At every time slot, this algorithm allocates the *most* connected server to its shortest connected non-empty queue, updates the queue lengths and then allocates the other connected server to its shortest connected non-empty queue (refer to Section 2.8.2). It can be easily shown that this algorithm satisfies Equation (3.6).

3.4 Optimality of MB Policies

The basic idea behind the proof of Theorem 4 is to show that an arbitrary policy π can be improved by "making it act similar to an MB policy" for at least one time slot. More precisely, suppose that there exists a (potentially infinite) sequence of instants $t_1 < t_2 < \cdots < t_i < \cdots$, $t_i \in T, T = 1, 2, \ldots$, such that $\pi(t_i) \neq \pi^{MB}(t_i)$. The idea is to show that we can construct a policy π_1 such that $\pi_1(t) = \pi^{MB}(t)$, for $t \leq t_1$

and such that $\mathbf{W}_{\pi_1} \leq_{st} \mathbf{W}_{\pi}$. Then we can use this technique to create a sequence of policies, $(\pi_n : n = 1, 2, \ldots)$, each of which outperforms π and acts similar to π^{MB} up to time t_n.

We will need the following result to prove Theorem 4. The proof of Lemma 10 is given in Appendix D.

Lemma 10. *Consider an arbitrary policy π, such that $\pi(t) = \pi^{MB}(t)$ for all $t \leq \tau$ for some $\tau \in T$; a policy $\bar{\pi}$ can be constructed such that $\bar{\pi}(t) = \pi(t)$, for $t \leq \tau$, $\bar{\pi}(t) = \pi^{MB}(t)$, for $t = \tau + 1$ and*

$$\mathbf{W}_{\bar{\pi}} \leq_{st} \mathbf{W}_{\pi} \tag{3.8}$$

Proof of Theorem 4. We start by constructing a set of policies, $\{\pi_i\}$, using Lemma 10 as follows:

1- For any policy π, construct a policy π_1 which acts similar to MB at $t = 1$ and such that their corresponding total number of packets $\mathbf{W}_{\pi}$ and $\mathbf{W}_{\pi_1}$ respectively satisfy

$$\mathbf{W}_{\pi_1} \leq_{st} \mathbf{W}_{\pi}$$

2- Construct a policy π_2 that acts similar to π_1 at $t = 1$ and similar to MB at $t = 2$ such that

$$\mathbf{W}_{\pi_2} \leq_{st} \mathbf{W}_{\pi_1}$$

3- For any policy π_i we construct a policy π_{i+1} that acts similar to π_i up until time slot $t = i$ and similar to MB at $t = i + 1$ and such that

$$\mathbf{W}_{\pi_{i+1}} \leq_{st} \mathbf{W}_{\pi_i}$$

4- Repeat the argument in step 3 above m times. The resulting policies π_i, $i = 1, 2, \ldots, m$ agree with MB for the time slots $t = 1, 2, \ldots, i$ and their corresponding total number of packets sequences define a monotonic ordering sequence that satisfy the following

$$\mathbf{W}_{\pi_m} \leq_{st} \mathbf{W}_{\pi_{m-1}} \leq_{st} \ldots \leq_{st} \mathbf{W}_{\pi_1} \leq_{st} \mathbf{W}_{\pi} \tag{3.9}$$

For an increasing value of m, the sequence in (3.9) defines a limiting policy π^* that acts similar to MB at all time slots and dominates any arbitrary policy π in the sense that

$$\mathbf{W}_{\pi^*} \leq_{st} \mathbf{W}_{\pi}$$

Theorem 4 follows from the fact that $\mathbf{W}_{\mathbf{MB}} \leq_{st} \mathbf{W}_{\pi^*}$ since $\pi^* \equiv \pi^{MB}$ by construction. □

3.5 Simulation Results

In this section, we present some simulation results that help understand the behavior of MB policies for the system described in Section 3.2. The metric we used in the simulation is denoted by EQ, the long-run average of the total number of queued packets in the system. The 99% confidence interval was computed for all results and is shown (when relevant) in the graphs below.

Similar to Chapter 2, we consider five policies in this simulation. They are: MB (i.e., LCSF/LCQ), LB (i.e., MCSF/SCQ), randomized, MCSF/LCQ, and LCSF/SCQ policies. The MB/LB policies provide "upper and lower" performance limits, in the sense of Theorems 4 and 5. The definitions of these five policies can be found in Section 2.9.

Figures 3.2 and 3.3 are samples of the simulation results we obtained. They depict

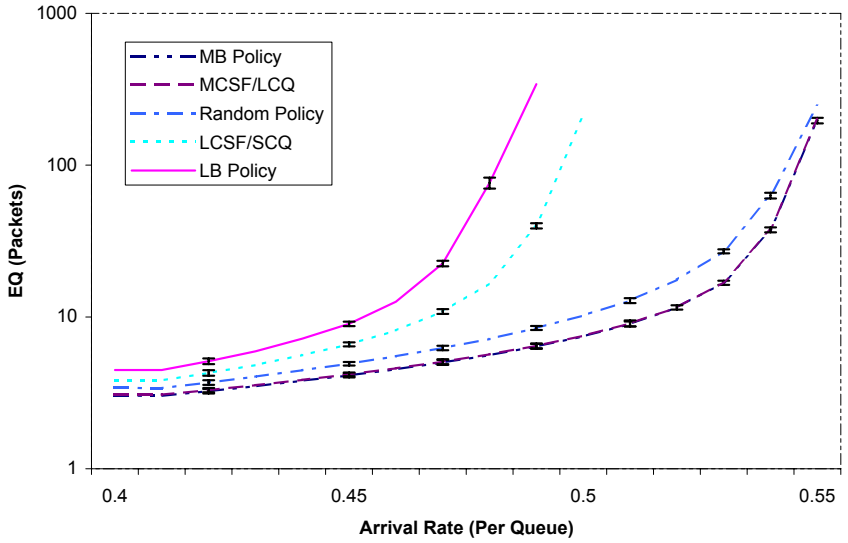


Figure 3.2: EQ versus load under different policies; $p = 0.4$ and $P_{sc} = 0.85$.

EQ, the average total number of packets queued in the system, versus the offered load for different policies with channel connection probability $p = 0.4$ and packet service completion probabilities $P_{sc} = 0.85$ and $P_{sc} = 0.95$ respectively.

The above figures show that the MB policy outperforms the other policies. However, the performance enhancement of the MB compared to MCSF/LCQ is very small and is within the boundaries of the confidence interval. Hence, we can say that the MCSF/LCQ policy performance is comparable to the MB policy. This conclusion makes an intuitive sense since both policies take the same action most of the time.

Figure 3.4 depicts the behavior of the MB policy for different values of P_{sc}. The graph clearly shows the effect of the service completion probability on the policy performance. The higher the value of P_{sc}, the better the MB policy performs.

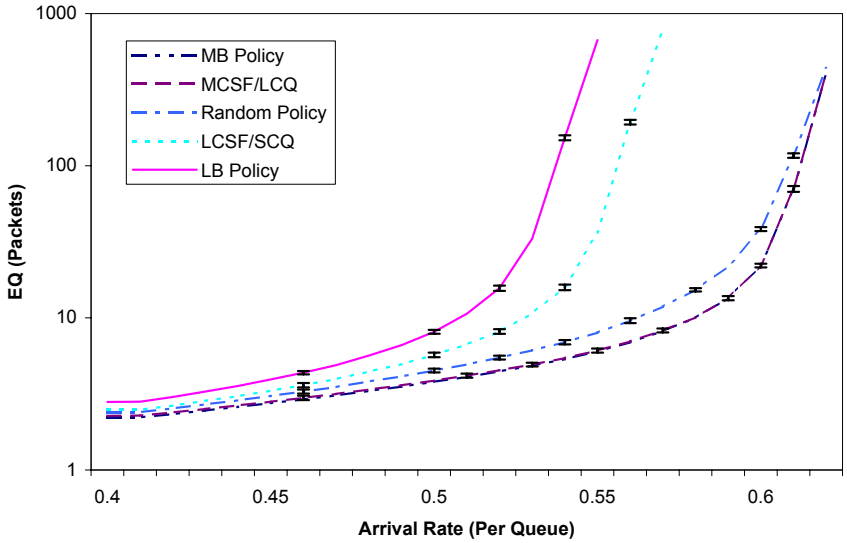


Figure 3.3: EQ versus load under different policies; $p = 0.4$ and $P_{sc} = 0.95$.

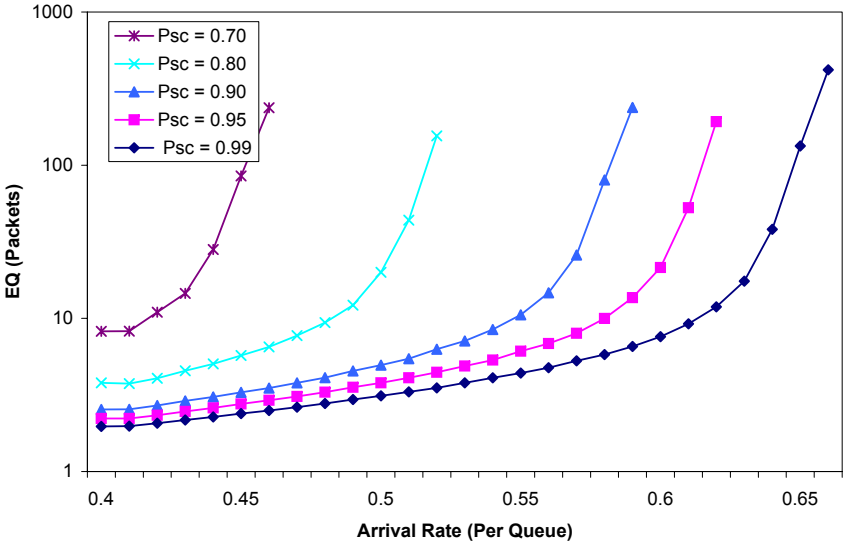


Figure 3.4: EQ versus load under an MB policy for different P_{sc}; $p = 0.4$.

Chapter 4

Optimal Scheduling in HSDPA Networks: Dynamic Programming Approach

In this chapter, we present an analytic model and a methodology to determine the optimal packet scheduling policy in a High Speed Downlink Packet Access (HSDPA) system. The optimal policy is the one that maximizes cell throughput while maintaining a level of fairness between the users in the cell. A discrete stochastic dynamic programming model for the HSDPA downlink scheduler is presented. Value iteration is then used to solve for optimal scheduling policy. In this chapter, we use a FSMC (Finite State Markov Channel) to model the HSDPA downlink channel. A near-optimal heuristic scheduling policy is developed. Simulation is used to study the performance of the resulted heuristic policy and compare it to the computed optimal policy.

4.1 Introduction

The rapid development in wireless technology enabled the implementation of services which are so far available only on IP-based networks. Each one of these services has its own Quality of Service (QoS) requirements, in terms of bandwidth (Web browsing

service for instance), or end-to-end delay for real-time applications such as voice over IP (VoIP). The increasing demand on such services triggered the evolution of third generation wireless networks toward IP-based, packet-switched networks. The new 3G systems (e.g., HSDPA, CDMA2000) were designed to have an IP-based infrastructure that enables the reuse of the available IP resources and technologies and to reduce the operational system cost [43]. Nevertheless, the added packet switching capability introduced new challenges that have to be dealt with.

One of the challenges is to meet the QoS requirements of the offered services. Wireless links in general have different channel characteristics compared with wireline links. They are subject to time- and location-dependent signal attenuation, fading and interference, which will result in bursty errors and time varying channel capacities. Therefore, the direct application of the available wireline QoS methods is impractical. Furthermore, it is extremely difficult to provide hard (absolute) QoS guarantees and only soft QoS (Differentiated Services) can be implemented [44]. Packet scheduling is one of the most important QoS control approaches for wireless communications [45]. The scheduling algorithms in wireless systems should take into consideration the variation in channel characteristics, make use of the user diversity to maximize throughput, and aim at providing all users with a fair share of the network resources.

High-Speed Downlink Packet Access (HSDPA) is a 3G wireless network that provides high cell peak data rate (up to 14.4 Mbps for Revision 5) on the downlink by incorporating Adaptive Modulation and Coding (AMC), Hybrid ARQ and fast scheduling [46], [47]. The development of this system represents an important step in the effort to establish a ubiquitous wireless access paradigm that provides a host of services in a cost-effective, high-speed and reliable manner. This motivated our initial interest in this problem.

Scheduling in HSDPA systems involves not only *Transmission Time Interval (TTI) allocation* but also *code allocation*. On the downlink, HSDPA uses Code and

Time Division Multiplexing (CDM/TDM) and has 15 codes to be allocated per TTI. Most of the available work in scheduler design (e.g. [44], [48] and [49]) is based on the intuition and creativity of the designers. The designer usually selects an optimization criterion that represents some important performance measure (in her/his opinion), builds an algorithm based on that criterion and then tries to establish confidence in it using backward analysis or simulation. Such an approach can be described as a *procedural approach*. This, most likely, will result in a suboptimal algorithm at the best, that performs well in some scenario and poorly in others. This happens especially in systems such as HSDPA, since it uses a complex set of features such as Hybrid Automatic Repeat reQuest (H-ARQ) and Adaptive Modulation and Coding (AMC). These features introduced many new and interrelated tuning parameters which cannot be grasped by a single selected optimality criterion. Another observation is the lack of work on schedulers that dynamically allocate codes as well as time slots to the users in an HSDPA system. Scheduling related processes in this system spans across both layer 1 (physical) and layer 2 (media access control). Hence, a cross-layer design is desirable for such systems [50].

In this chapter, we present a novel approach for scheduling in 3G systems. Using stochastic dynamic programming, we built and analyzed a realistic model of a HSDPA scheduler. This model introduces a simplifying abstraction of the real scheduler which estimates system behavior under different operating conditions and describes the role of various system components. This model can be readily solved numerically to obtain the optimal code allocation policy for a given *objective function*.

This approach can be considered as a *unified approach* since the same model can be used when solving for different objective functions by simply changing the reward associated with the model to reflect a new optimality objective. Different objective functions may result in different optimal policies. For example, if the objective is to maximize the cell throughput, then a greedy Max C/I scheduler can achieve this goal

by favoring the user with the best channel conditions. However, this policy will result in starvation of the users with poor channel quality. On the other hand, Max-min scheduler will divide the resources fairly between all the users in the cell to achieve fairness at the expense of cell throughput. The optimal policy lies somewhere in the middle and depends on what degree of fairness is required. The proposed approach produces an optimal scheduling policy in the sense that it maximizes cell throughput for a given fairness. It provides an elegant analytic foundation for scheduling problems and may be used as a benchmarking tool for other schedulers or to test against heuristics.

The presented approach can also be used to tackle optimal packet scheduling in the most recent releases of 3GPP, e.g., Long Term Evolution (LTE). These systems use the same basic components at the link layer level (e.g., HARQ and AMC) that are used in HSDPA systems. They also use other new technologies in the physical layer (that was not used in HSDPA systems) to increase downlink/uplink data rates up to 150Mbps/50Mbps, namely orthogonal frequency-division multiplexing (OFDM) and multiple-input multiple-output (MIMO) antenna systems [51], [52]. The packet scheduler in these systems is responsible for frequency allocation (frequency carriers that correspond to OFDM) as well as CDM codes during each TTI [53], [54]. The model we present here may be extended for an LTE system. For this we need to modify the feasible actions available to the scheduler to include frequency allocation during each TTI.

Overall, the contributions of the work presented in this chapter can be summarized by the following:

1. We provide an analytical approach to model the downlink scheduler in a 3G HSDPA system. This approach can be extended to other 3G/4G wireless systems.

2. Using the theory of Dynamic Programming we present an optimization framework for the determination of the optimal policy for the HSDPA downlink packet scheduler. The applicability of this framework is demonstrated using a two user/queue case (in this case the pictorial visualization of the optimal policy structure is possible).

3. A near-optimal, *heuristic* policy is proposed, based on the structural properties of the optimal policy and its dependence on changing system parameters.

4. We conduct a simulation study to quantify the effect of different model parameters on the behavior of the optimal policy. We also study the performance of the proposed heuristic policy and compare it to that of the optimal policy.

4.2 Problem Definition

4.2.1 General Description of HSDPA System

Third generation release R'5 [43] [55], also called High-Speed Downlink Packet Access (HSDPA), is an IP-based network that can offer users a high-speed asymmetric radio link with downlink peak bit rate up to 14.4 Mbps. The HSDPA uses a single time-shared channel, called High Speed Downlink Shared CHannel (HS-DSCH), per cell/sector. This channel is divided into 2-ms Transmission Time Intervals (TTI). Each TTI may be used to transfer packets to one or more users at a rate that depends on their User Equipment (UE) capabilities and needs. The UE can use up to 15 codes simultaneously to achieve higher rate. More than one user can share the same slot by dividing the available 15 CDM codes between them. In such case, the scheduler need to choose not only the user/users to be served in the next time slot, but also the number of codes each user will receive.

The HSDPA system uses Adaptive Modulation and Coding (AMC) technique to

adapt the transmission rate to the user's channel conditions. The selected modulation scheme and coding rate are chosen such that a fixed low error rate is achieved (usually 10%). The erroneous packets will be retransmitted during the next scheduled TTI using Hybrid ARQ [56]. In this technique, a retransmitted packet will be soft combined at the receiver with the previous unsuccessful transmitted versions of itself (i.e., combining the signal energy of multiple retransmissions of the same packet) to increase its SNR and its detection probability. HSDPA supports two combining techniques: Chase Combining (CC) and Incremental Redundancy (IR). In CC, the base station (which is called NodeB in the 3GPP technical specification [43]) retransmits the exact same set of coded symbols of the original packet. Then the receiver combines the packet energy with the energy of previously received unsuccessful transmissions of this packet. With IR, the same principle is used except that different redundancy information can be sent (i.e., using different error coding rate) in every re-transmission. This will result in incremental increase in the coding gain and hence, fewer retransmissions will be needed (compared to CC). This is particularly useful when the initial transmission uses high coding rates. However, it increases the complexity requirements for the UE [57].

Our objective is *to investigate a methodology that determines the optimal scheduling regime, controlling the allocation of the time-code resources fairly between all the active users while maximizing the overall cell throughput.* The desired scheduling algorithm should have the following characteristics: channel awareness, fairness and high-speed resource allocation.

4.2.2 HSDPA Downlink Scheduler Abstraction

The HSDPA downlink channel uses a mix of Time Division Multiplexing and Code Division Multiplexing:

- Time is slotted into fixed-length, 2-ms TTIs.
- During each TTI, there are 15 available codes that may be allocated to one or more users.

During one TTI, the channel capacity associated to a single user depends on the number of allocated codes and on the channel condition. This is mainly due to the fact that HSDPA uses AMC to adapt the transmission rate to the current channel conditions. A mobile user with good channel conditions will experience higher data rate than the other users.

The diagram in Figure 4.1 depicts a conceptual realization of the HSDPA downlink scheduler. Different users have separate buffers in the base station (Node-B according

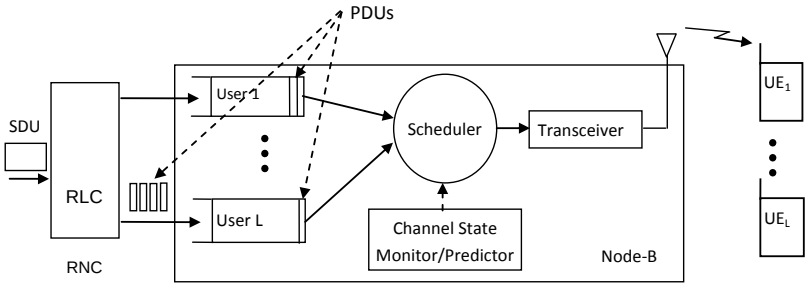


Figure 4.1: HSDPA scheduler model (downlink)

to 3GPP), and they are competing for the system resources. A channel state monitor/predictor is necessary to monitor current channel conditions of each user and predict channel states during the next TTI. This information will then be used to adapt the transmission rate to the expected channel conditions. The arrived Service Data Units (SDU) are assumed to be segmented by the Radio Link Control (RLC) into u_i fixed size Protocol Data Units (PDU) (for user i) before delivering them to Node-B. The PDUs then will be classified and inserted into the proper buffers awaiting transmission to the intended user. RNC is the Radio Network Controller unit which implements the RLC protocol.

4.2.3 HSDPA Downlink Channel Model

The wireless channel for the HSDPA system is modeled as a Finite-State Markov Channel (FSMC) following [58]. The FSMC was proved to be a good model of the wireless medium and has been shown to be in good agreement with realistic cases (c.f. [59], [60]). FSMC modeling is done by partitioning the signal-to-noise ratio (SNR) into finite number of intervals, each representing a state in a Markov Chain. Assuming that the fading is slow enough then the channel states for consecutive time epochs are neighboring states. In this case, the model will be reduced into a discrete-time birth and death process, as shown in Figure 4.2.

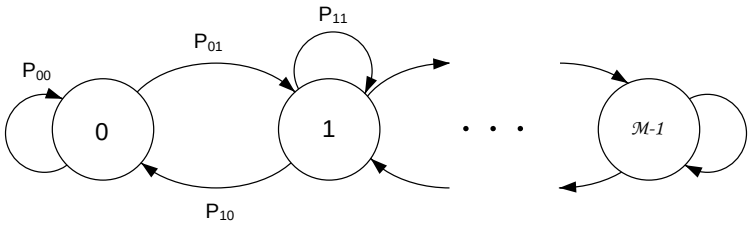


Figure 4.2: FSMC model for HSDPA downlink channel.

The FSMC model is a more accurate representation of the wireless channel with rate adaptation (as it is the case with third generation networks and beyond) than the traditionally used Gilbert-Elliott two-state channel model.

Depending on the expected SNR state, different modulation and error-correcting coding rates can be dynamically selected from a set of Modulation and Coding Schemes (MCS) [61]. The higher the order of the MCS selected the higher the transmission rate. The SNR is mapped directly into MCS and hence into data rates. In light of this, the states in our channel model will equivalently represent data rate levels rather than SNR.

4.3 Optimal Policy Determination

In this section, we propose an approach, based on Markov Decision Process (MDP), to find the optimal code allocation policy for the HSDPA downlink scheduler. We present a general model for this system and suggest a reward function that captures the objective function.

To describe a system as a MDP model, the states, actions, rewards and transition probabilities have to be defined first. In our proposed model, time is slotted in constant intervals of size Δt. Let $T = \{1, 2, \ldots\}$ denote the set of decision epochs of the system (it is assumed that the decision epochs are at the beginning of each time slot, PDUs that arrive during any time slot can only be considered for transmission in the subsequent time slots). At time $t \in T$, we denote by $\boldsymbol{s}(t)$ and $\boldsymbol{a}(\boldsymbol{s})$, the system state and the action taken at that state (to be defined later) respectively. HSDPA downlink scheduler is modeled by the 5-tuple $(T, \mathcal{S}, A, P_{\boldsymbol{ss'}}(\boldsymbol{a}), R(\boldsymbol{s}, \boldsymbol{a}))$, where $\mathcal{S}$ and A are the state and action spaces, $P_{\boldsymbol{ss'}}(\boldsymbol{a}) \triangleq Pr(\boldsymbol{s}(t+1) = \boldsymbol{s'} | \boldsymbol{s}(t) = \boldsymbol{s}, \boldsymbol{a}(\boldsymbol{s}) = \boldsymbol{a})$ is the state transition probability, and $R(\boldsymbol{s}, \boldsymbol{a})$ is the immediate reward function (to be introduced shortly) when at state $\boldsymbol{s}$ and taking action $\boldsymbol{a}$.

4.3.1 Basic Assumptions

There are L active users in the cell. A user $i \in I = \{1, 2, \ldots, L\}$ is allocated a buffer of finite size B. Initially, error-free transmission will be assumed to eliminate the need for retransmission queues and to reduce the model complexity. Later (in Section 4.5), the more general problem with retransmission will be examined. SDUs arrive at the RNC during the current TTI will be segmented by RLC into a fixed number of PDUs (u_i) and delivered to Node-B to be inserted into their respected buffer at the beginning of the next TTI.

For each user $i \in I$ and slot $t \in T$, we define:

- $y_i(t)$ the number of scheduled PDUs,
- $x_i(t) \in \mathcal{X} = \{0, 1, 2, ..., B\}$ the queue size,
- $z_i(t) \in \{0, u_i\}$ the number of arriving PDUs per slot.

The SDUs destined to user i arrive at the RNC during one TTI according to a Bernoulli process with parameter q_i. Arrivals are assumed to be independent of the system state and of each other. The PDU size is chosen to be equal to the minimum Transport Format and Resource Combination (TFRC) for one code (i.e., one code is needed to transmit one PDU when the channel is in state 1). The scheduler can assign the available 15 codes as chunks of c codes at a time to active users in the system. The chunk size c must divide the total number of codes (15); therefore, $c \in \{1, 3, 5, 15\}$. For example, choosing $c = 5$ means that the policy can assign 0, 5, 10, or 15 of the available 15 codes to any user at any given TTI. We introduced the chunk size c in our analysis in order to reduce the action set size and the computational complexity. At the same time, this formulation enables us to measure the system performance sensitivity to the resolution of code allocation in HSDPA systems as it will be shown later.

4.3.2 FSMC State Space

The channel state of user i during time slot t is denoted by $\gamma_i(t)$; and its associated channel state space is the set $\mathcal{N} = \{0, 1, \ldots, N-1\}$, where N is the total number of available channel states. $\mathcal{N}$ constitutes a subset of the available MCS set recommended by 3GPP. The elements of $\mathcal{N}$ were ordered in a way such that $\gamma_i(t) \in \mathcal{N}$ is directly proportional to the number of PDUs that can be transmitted by user i in one TTI. This assumption is in good agreement with the corresponding RFC [43]. Furthermore, we assume that user i channel can handle up to $\gamma_i(t)$ PDUs per code, i.e., a value of $\gamma_i(t) = 2$ means that at time t, user i can transmit two PDUs using one

code and up to 30 PDUs when using all the 15 codes. The state transition probability $P_{\gamma_i\gamma_i'}$ is assumed known since it can be calculated (from SNR measurements) for any mobile environment with Rayleigh fading channel [58].

4.3.3 State Space and Action Set

The system state $\boldsymbol{s}(t) \in \mathcal{S}$ is a vector comprised of multiple state variables representing the queue sizes and the channel states for the L users. In other words,

$$\boldsymbol{s}(t) = (x_1(t), x_2(t), \ldots, x_L(t), \gamma_1(t), \gamma_2(t), \ldots, \gamma_L(t)) \tag{4.1}$$

and the state space of the system, $\mathcal{S} = \{\mathcal{X} \times \mathcal{N}\}^L$, has finite size since the buffers have finite sizes and the channel state spaces are also finite.

The action space A is the set of all possible actions. The action $\boldsymbol{a}(\boldsymbol{s}) \in A$ is taken when the system is in state $\boldsymbol{s}$. The action taken at each slot corresponds to the number of codes allocated to each user. Let $D = \{0, 1, \ldots, 15/c\}$ be the action space for a single user, where c is the code chunk size (the minimum number of codes that can be allocated at any given time). For example, if $c = 5$ then $D = \{0, 1, 2, 3\}$. Let $a_i(\boldsymbol{s}) \in D$ be the number of code chunks allocated to user i when in state $\boldsymbol{s}$. Then the number of codes allocated to user i during time slot t is $a_i(t)c$ and the number of scheduled (for transmission) PDUs from queue i (corresponding to user i) is $y_i(t) = a_i(t)\gamma_i(t)c$. In this case, $\boldsymbol{a}(\boldsymbol{s})$ will be the collection of code allocation to all users, that is

$$\boldsymbol{a}(\boldsymbol{s}) = (a_1(\boldsymbol{s}), a_2(\boldsymbol{s}), \ldots, a_L(\boldsymbol{s})) \tag{4.2}$$

subject to

$$\sum_{i=1}^{L} a_i(\boldsymbol{s}) \cdot c \;\leq\; 15, \tag{4.3}$$

$$a_i(\boldsymbol{s}) \;\leq\; \left\lceil \frac{x_i(t)}{\gamma_i(t)c} \right\rceil, \tag{4.4}$$

$$\gamma_i(t) \;>\; 0 \tag{4.5}$$

The constraint in (4.3) means that the policy can not allocate more than the available 15 codes at each time slot. The constraint in (4.4) makes the policy conserving by allocating no more codes to user i than that required to empty its buffer. The right side of (4.4) represents the number of code chunks required to empty queue i.

4.3.4 Reward Function

In this subsection, we introduce the reward function used to determine the optimal allocation policy. As stated previously, the objective is to maximize the throughput while maintaining fairness between active users. Let the *fairness factor*, denoted by $\sigma > 0$, be a parameter that reflects the significance of fairness in the optimal policy. Define $\bar{x}$ as the average instantaneous size of the L queues in the system at time t, i.e., $\bar{x} = \frac{1}{L}\sum_{i=1}^{L} x_i$, (we suppressed the time index to simplify notation). The reward function $R(\boldsymbol{s}, \boldsymbol{a})$ will have two components corresponding to two objectives (throughput and fairness) and it is given by

$$\begin{aligned} R(\boldsymbol{s}, \boldsymbol{a}) &= \sum_{i=1}^{L} y_i - \sigma \sum_{i=1}^{L} (x_i - \bar{x})\, \mathbb{1}_{\{x_i = B\}} \\ &= \sum_{i=1}^{L} a_i \gamma_i c - \sigma \sum_{i=1}^{L} (B - \bar{x})\, \mathbb{1}_{\{x_i = B\}} \end{aligned} \tag{4.6}$$

where $\mathbb{1}_{\{\cdot\}}$ is the indicator function. The positive term of the reward relates to the cell throughput. If the reward is composed of this part only, then the policy will

always favor the users with good channel conditions. Therefore the users with less favorable channels will starve; their queues will grow larger and start loosing packets. We introduced the second term, which guarantees a level of fairness and reduces dropping probability. Lower σ will result in a policy that favors cell throughput over fairness, while higher σ has the opposite effect. Overall, $R(\boldsymbol{s}, \boldsymbol{a})$ will produce a policy that maximizes cell throughput for a given fairness factor σ.

4.3.5 State Transition Probability

$P_{\boldsymbol{s}\boldsymbol{s}'}(\boldsymbol{a})$ denotes the probability that choosing an action $\boldsymbol{a}$ at time t when in state $\boldsymbol{s}$ will lead to state $\boldsymbol{s}'$ at time $t+1$. Using Equations (4.1) and (4.2), $P_{\boldsymbol{s}\boldsymbol{s}'}(\boldsymbol{a})$ can be stated as follows

$$\begin{aligned} P_{\boldsymbol{s}\boldsymbol{s}'}(\boldsymbol{a}) &\triangleq Pr(\boldsymbol{s}(t+1) = \boldsymbol{s}' | \boldsymbol{s}(t) = \boldsymbol{s}, \boldsymbol{a}(t) = \boldsymbol{a}) \\ &= Pr(x'_1, \ldots, x'_L, \gamma'_1, \ldots, \gamma'_L | x_1, \ldots, x_L, \gamma_1, \ldots, \gamma_L, a_1, \ldots, a_L) \end{aligned} \quad (4.7)$$

The evolution of the queue size (x_i) is given by

$$\begin{aligned} x'_i &= \min\left([x_i - y_i]^+ + z'_i\,, B\right) \\ &= \min\left([x_i - a_i\gamma_i c]^+ + z'_i\,, B\right) \end{aligned} \quad (4.8)$$

where, z'_i is the number of packets arriving at queue i at $t+1$, $[e]^+$ equals e if $e \geq 0$ and 0 otherwise. The channel state γ_i depends only on the previous channel state, that is $Pr(\gamma'_i|\boldsymbol{s}) = Pr(\gamma'_i|\gamma_i) = P_{\gamma_i\gamma'_i}$. Accordingly, we can write Equation (4.7) as follows (refer to Appendix E.1 for detailed derivation of (4.9))

$$P_{\boldsymbol{s}\boldsymbol{s}'}(\boldsymbol{a}) = \prod_{i=1}^{L} \left(P_{x_i x'_i}(\gamma_i, a_i)\, P_{\gamma_i\gamma'_i}\right) \quad (4.9)$$

where $P_{\gamma_i\gamma_i'}$ is the Markov transition probability of the FSMC. Define $W1$ and $W2$ as follows

$$\begin{aligned} W1 &= [x_i - a_i\gamma_i c]^+ + u_i \\ W2 &= [x_i - a_i\gamma_i c]^+ \end{aligned}$$

$P_{x_i x_i'}(\gamma_i, a_i)$ can be analytically derived using Equation (4.8) and the total probability law (see Appendix E.2), and is given by the following expression

$$P_{x_i x_i'}(\gamma_i, a_i) = \begin{cases} 1 & \text{if } x_i' = x_i = B \ \&\ a_i\gamma_i = 0, \\ q_i & \text{if } x_i' = x_i = B \ \&\ 0 < a_i\gamma_i c \leq u_i, \\ q_i & \text{if } x_i' = B \ \&\ x_i < B \ \&\ W1 \geq B, \\ q_i & \text{if } x_i' < B \ \&\ x_i' = W1, \\ 1 - q_i & \text{if } x_i' < B \ \&\ x_i' = W2, \\ 0 & \text{otherwise.} \end{cases} \tag{4.10}$$

The first three cases in Equation (4.10) correspond to the boundary state/condition (i.e., when the queue length reaches the buffer size), while the remaining cases correspond to interior states.

4.3.6 Dynamic Programming Formulation

In this work, we investigate an infinite-horizon MDP. We use the total expected discounted reward optimality criterion with discount factor λ, where $0 < \lambda < 1$, in order to find the policy π among all policies, that maximizes the *value function* $V^\pi(s)$. Let $V^*(\boldsymbol{s})$ be the maximal discounted value function (i.e., $V^*(\boldsymbol{s}) = \sup_\pi V^\pi(\boldsymbol{s})$), attained

when applying the optimal policy π^*. Then the following optimality equation (also known as Bellman equation) is used to characterize the optimal policy [62], [63]:

$$V^*(\boldsymbol{s}) = \max_{\boldsymbol{a}\in A}[R(\boldsymbol{s}, \boldsymbol{a}) + \lambda \sum_{\boldsymbol{s}'\in\mathcal{S}} P_{\boldsymbol{s}\boldsymbol{s}'}(\boldsymbol{a})V^*(\boldsymbol{s}')] \tag{4.11}$$

Value iteration [13] (also known as successive approximation) is used to solve this model numerically. The first step is to define $V_0(\boldsymbol{s})$ to be an arbitrary bounded function. Then $V_n(\boldsymbol{s})$, $n > 0$ is determined by the following recursion

$$V_n(\boldsymbol{s}) = \max_{\boldsymbol{a}\in A}[R(\boldsymbol{s}, \boldsymbol{a}) + \lambda \sum_{\boldsymbol{s}'\in\mathcal{S}} P_{\boldsymbol{s}\boldsymbol{s}'}(\boldsymbol{a})V_{n-1}(\boldsymbol{s}')]$$

V_n converges to V^* as $n \to \infty$ [14]. For a given $\epsilon > 0$, the algorithm can be stopped after n iterations, provided that

$$\|V_{n+1} - V_n\| < \epsilon(1-\lambda)/2\lambda \tag{4.12}$$

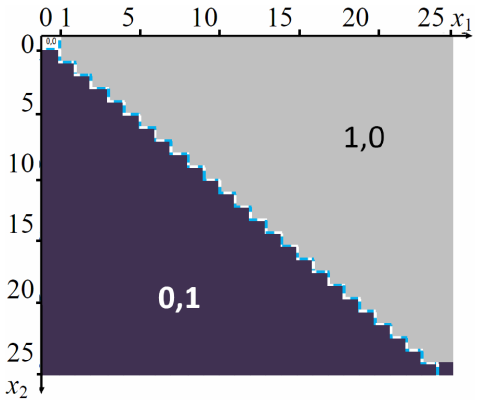
where $\|v\| = \sup_{s\in\mathcal{S}} |v(s)|$. If (4.12) holds, then $\|V_{n+1} - V^*\| < \epsilon/2$, according to [13].

4.4 Two Users with 2-State FSMC

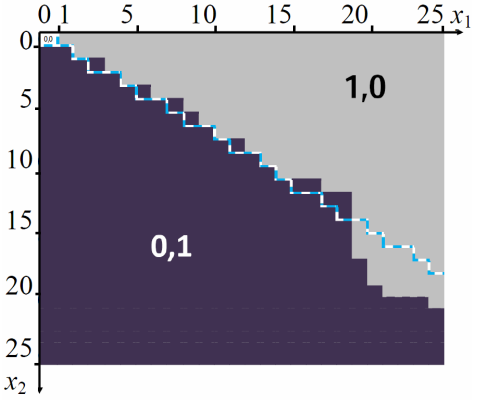
The approach presented earlier was used to model the case when there are two users (i.e., $L = 2$) sharing the same cell. The channel for user i is modeled as a two-state FSMC with transition probability matrix P_i

$$P_i = \begin{bmatrix} 1-\alpha_i & \alpha_i \\ \beta_i & 1-\beta_i \end{bmatrix} \tag{4.13}$$

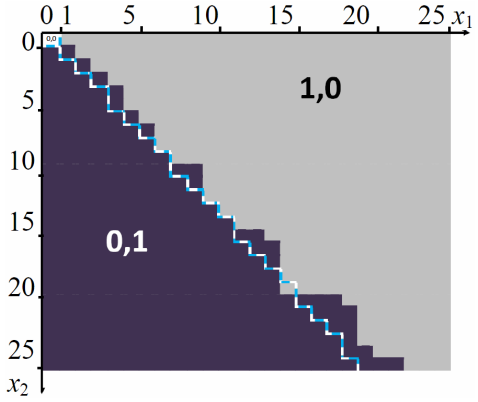
The two-user case will yield a policy that is easy to visualize, evaluate, and to



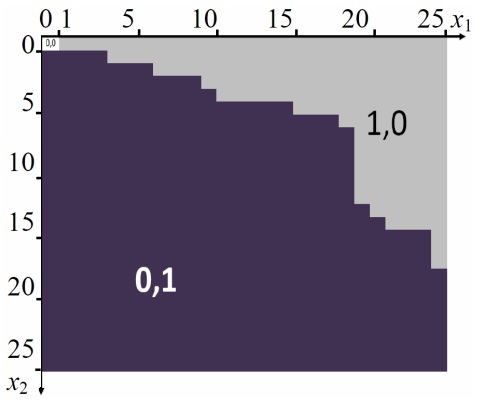
(a) Symmetrical case



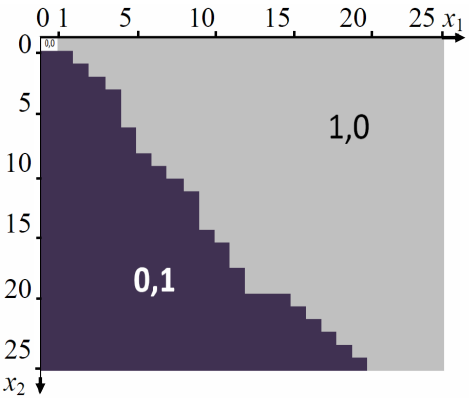
(b) $P(\gamma_1 = 1) = 0.8$ and $P(\gamma_2 = 1) = 0.5$



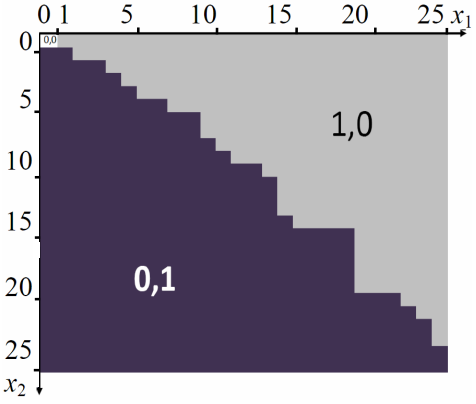
(c) $P(z_1 = 5) = 0.8$ and $P(z_2 = 5) = 0.5$



(d) $P(\gamma_1 = 1) = 0.8$ and $P(\gamma_2 = 1) = 0.3$



(e) $P(z_1 = 5) = 0.8$ and $P(z_2 = 5) = 0.3$



(f) $P(z_1 = 5) = 0.8$, $P(z_2 = 5) = 0.5$, $P(\gamma_1 = 1) = 0.8$ and $P(\gamma_2 = 1) = 0.5$

Figure 4.3: Optimal and heuristic (dotted line) policies for two user case; $c = 15$ (i.e., 0 or 1 chunks of size 15 codes can be assigned to a user), arrival batch size $u = 5$.

deduct conclusions for the optimal policy. It also serves as a verification for the proposed approach, since it is possible to visualize and plot the optimal policy in this case as it will be shown later. The obtained results can then be generalized to more complex cases involving more than 2 queues.

User i is said to be *connected* when $\gamma_i = 1$ with probability $P(\gamma_i = 1) = \alpha_i/(\alpha_i + \beta_i)$, and *not connected* ($\gamma_i = 0$) with probability $P(\gamma_i = 0) = \beta_i/(\alpha_i + \beta_i)$.

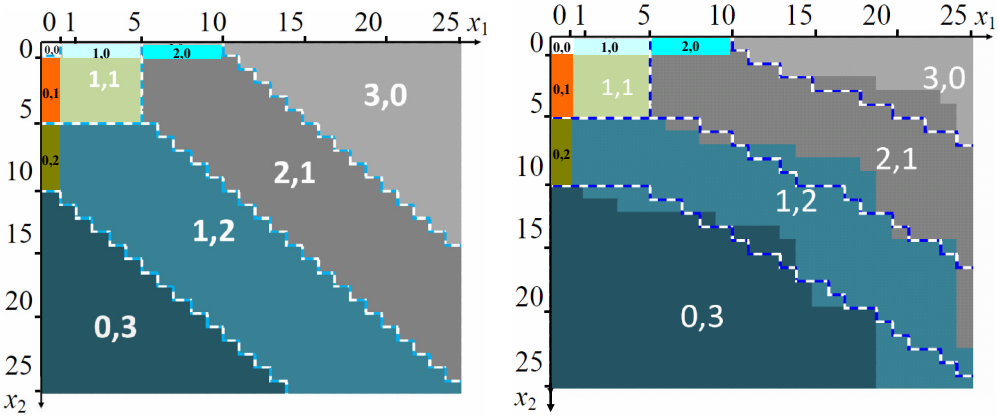
The remaining parameters were chosen as follows: $B = 25$, $\sigma = 0.5$, $\lambda = 0.95$, $\epsilon = 0.1$, and c =3, 5 or 15. The action space depends on the value of c. For example, if $c = 5$ then there are *four* possible actions for each user (i.e., $D = \{0, 1, 2, 3\}$). Since $\boldsymbol{a} = (a_1, a_2)$ corresponds to $a_1 c$ codes assigned to user 1 and $a_2 c$ codes assigned to user 2, we get $A = \{(0,0), (0,1), (0,2), (0,3), (1,0), (1,1), (1,2), (2,0), (2,1), (3,0)\}$. Similarly, when $c = 15$ then there are *two* possible actions per user (i.e., $D = \{0, 1\}$) and when $c = 3$ then there are *six* possible actions per user (i.e., $D = \{0, 1, 2, 3, 4, 5\}$).

4.4.1 Optimal Policy Structure

The model is solved using value iteration (Section 4.3.6) to determine the optimal scheduling policy. The effect of the channel quality and arrival probability on the behavior of the optimal policy was studied. Figures 4.3-4.5 provide the general structure of the optimal policy for c =15, 5, and 3 respectively.

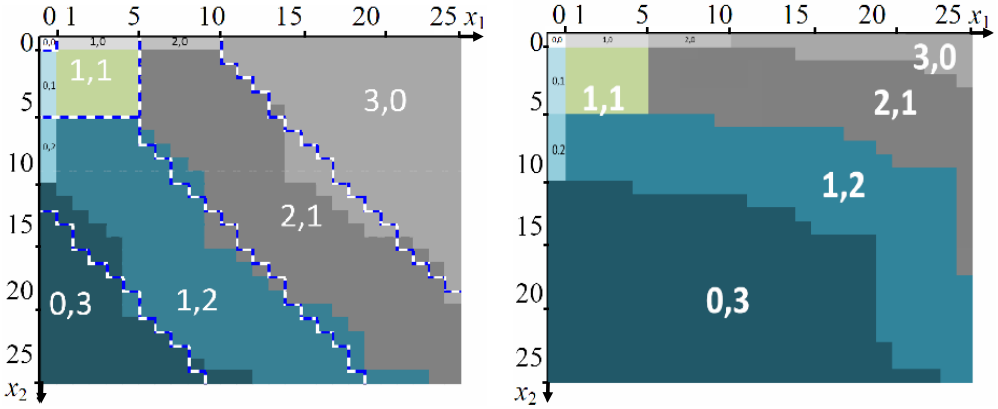
Optimal Policy for Two Symmetrical Users (Homogeneous System)

The optimal policy for two symmetrical users with the same channel characteristics ($\alpha_i = \beta_i = p$) for all $0 \leq p \leq 1$ and with $P(z_i = 5) = 0.5$ for all $i \in \{1, 2\}$ is shown in Figures 4.3(a)-4.5(a). Results for the case when the two users are connected, i.e., $\gamma_i = 1$, is shown here, since the two users are competing for the system resources. The other three cases when one or both of them are not connected, i.e., $\gamma_i = 0$, resulted



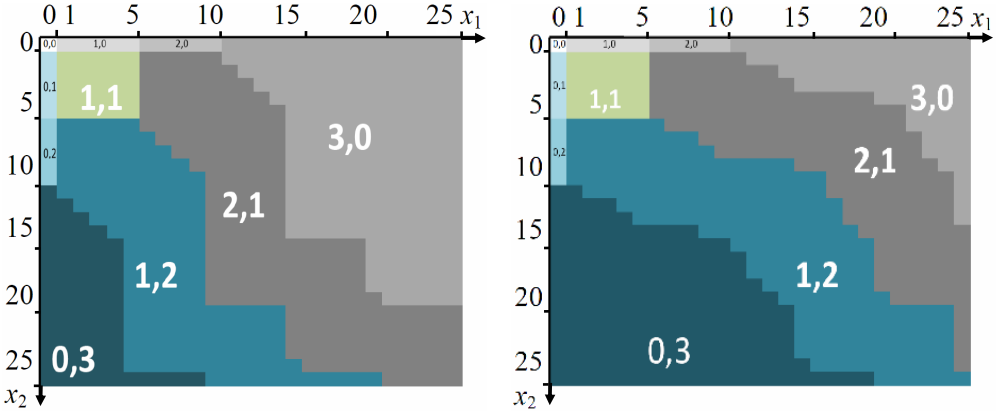
(a) Symmetrical case

(b) $P(\gamma_1 = 1) = 0.8$ and $P(\gamma_2 = 1) = 0.5$



(c) $P(z_1 = 5) = 0.8$ and $P(z_2 = 5) = 0.5$

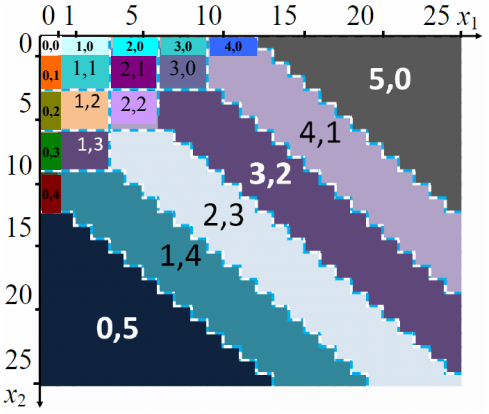
(d) $P(\gamma_1 = 1) = 0.8$ and $P(\gamma_2 = 1) = 0.3$



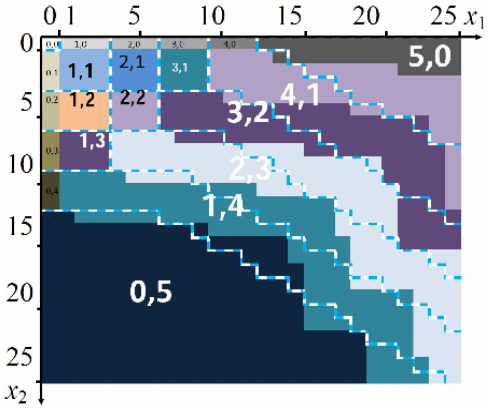
(e) $P(z_1 = 5) = 0.8$ and $P(z_2 = 5) = 0.3$

(f) $P(z_1 = 5) = 0.8$, $P(z_2 = 5) = 0.5$, $P(\gamma_1 = 1) = 0.8$ and $P(\gamma_2 = 1) = 0.5$

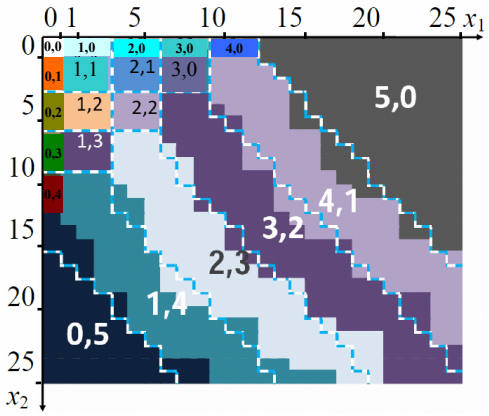
Figure 4.4: Optimal and heuristic (dotted line) policies for two user case; $c = 5$ (i.e., 0,1,2 or 3 chunks of size 5 can be assigned to a user), $u = 5$.



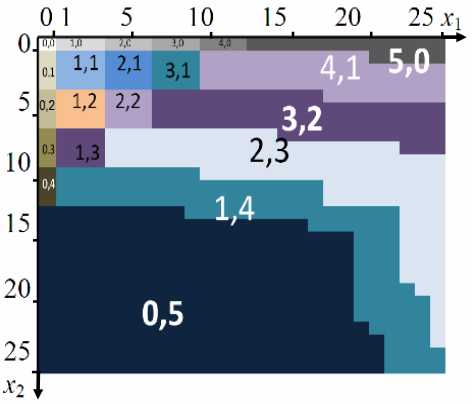
(a) Symmetrical case



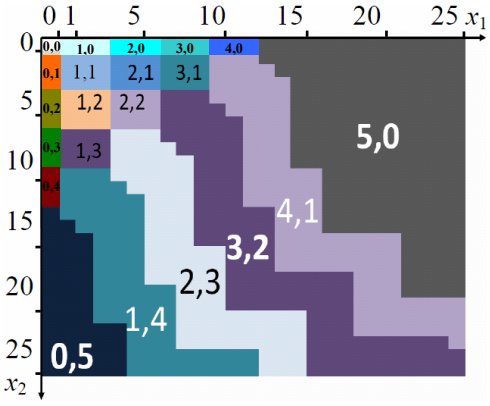
(b) $P(\gamma_1 = 1) = 0.8$ and $P(\gamma_2 = 1) = 0.5$



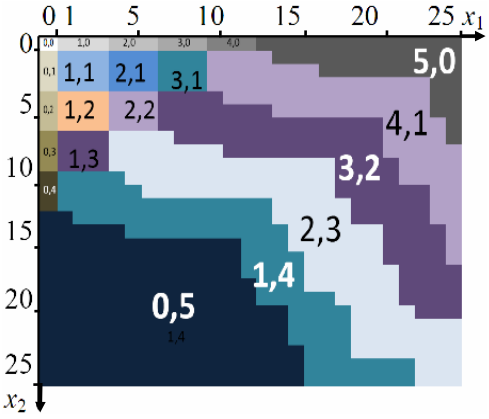
(c) $P(z_1 = 5) = 0.8$ and $P(z_2 = 5) = 0.5$



(d) $P(\gamma_1 = 1) = 0.8$ and $P(\gamma_2 = 1) = 0.3$



(e) $P(z_1 = 5) = 0.8$ and $P(z_2 = 5) = 0.3$



(f) $P(z_1 = 5) = 0.8$, $P(z_2 = 5) = 0.5$, $P(\gamma_1 = 1) = 0.8$ and $P(\gamma_2 = 1) = 0.5$

Figure 4.5: Optimal and heuristic (dotted line) policies for two user case; $c = 3$ (i.e., 0,1,2,3,4 or 5 chunks of size 3 can be assigned to a user), $u = 5$.

in a policy that assigns all the codes (required) to the connected user and nothing to the other.

The symmetrical users case is a special case of the model presented in Chapter 2, since a queue (user) in this model has the same connectivity variables to all servers (codes) at any given TTI (a special case of independent connectivity assumption of the model in Chapter 2). The optimal policy in this case can be described as follows: *divide the codes between the connected users in proportion to their queue length.* This conclusion agrees with the results of Chapter 2, since this action can be shown to be a most balancing one, i.e., the code allocation policy for the homogeneous case is an MB policy. When $c = 15$, the action space will be reduced to $A = \{(0,0),(0,1),(1,0)\}$ and the policy will be equivalent to *serve the longest connected queue (LCQ)*, which makes intuitive sense and matches with the findings in [23] for a case similar to our $c = 15$ case.

The Effect of Channel Quality on Policy Structure

The effect of the channel quality on the optimal policy structure when $\gamma_1 = \gamma_2 = 1$ is shown in Figures 4.3(b)-4.5(b). When $P(\gamma_1 = 1) > P(\gamma_2 = 1)$ the policy favors user 2, *which is less likely to be connected* compared to user 1. The bias in favor of user 2 is depicted in Figures 4.3(b)-4.5(b) by a larger dark area, which corresponds to optimal action (0,1), (0,3) or (0,5) respectively, compared to Figures 4.3(d)-4.5(d). We noticed that this bias increases as the difference between $P(\gamma_1 = 1)$ and $P(\gamma_2 = 1)$ increases. The reason is that using an LCQ in this situation will result in unbalanced system (queue 2 will grow larger and will have more packet drops than queue 1). User 2 will start experiencing unfairness in terms of higher delay and more drops. Hence, more resources have to be assigned to the user with the worst channel to avoid such a result. The resource sharing in this case will depend on the difference $\Delta P_\gamma = P(\gamma_1 = 1) - P(\gamma_2 = 1)$ as well as their relative queue length.

The Effect of Arrival Probability on Policy Structure

The arrival probability has an analogous effect (to that of the channel quality) on the optimal policy structure. The relative increase in one of the users arrival probability will result in more traffic inserted in that user's buffer and it will require more resources to keep the queue lengths stable and achieve fairness between the two users.

Figures 4.3(c)-4.5(c) show the optimal policy when $P(z_1 = 5) = 0.8$ and $P(z_2 = 5) = 0.5$ and both users have the same channel quality. The policy shifts in favor of the user with higher arrival probability (user 1 in this case). By comparing Figures 4.3(c)-4.5(c) to Figures 4.3-4.5(e), we noticed that this shift is proportional to the difference $\Delta P_z = P(z_1 = u) - P(z_2 = u)$.

Figures 4.3(f)-4.5(f) show that when $\Delta P_z = \Delta P_\gamma = 0.3$ the optimal policy favors user 2, i.e., the user with less connectivity. This agrees with intuition since both users share the available server capacity (the number of codes). The exogenous arrivals will always be added to the corresponding buffer (provided the availability of space in that buffer), while departures depend not only on the channel connectivity, but also on the maximum server capacity (total number of departures is bounded by the server capacity during each time slot; 15 PDUs in this case).

4.5 HSDPA System with Retransmission

We now expand our model to include the case of packet retransmission for unsuccessful packet transmissions. The resulting model will have two queues per user, a transmission queue and a retransmission queue. The state space for this system will be $\mathcal{S} = \{\mathcal{X} \times \mathcal{W} \times \mathcal{N}\}^L$, where $\mathcal{W} = \{0, 1, \ldots, B_r\}$ is the state space for the retransmission queue; and $\mathcal{X}$, $\mathcal{N}$ are the same as defined earlier. The computational complexity for finding the optimal policy for such system is substantially greater than the previous model and could become a prohibitive factor for a system with a large

number of users. Instead, we present an alternative model for this system, having less computational complexity. In order to avoid the increased dimensionality due to the retransmission queue, we consider a Bernoulli random process $\{\mu_i(t)\}_{t=1}^{\infty}$ that indicates the status of a packet transmission. This will result in a retransmission model with i.i.d. geometric service time, an assumption well-vetted in the literature, e.g., [23] and [25]. When codes are allocated to user i at time slot t, then the transmission is successful ($\mu_i(t) = 1$) with a probability ξ (corresponding to the probability of successful transmission). When $\mu_i(t) = 0$, then the transmission is unsuccessful and the PDUs have to be retransmitted. This happens with probability $1 - \xi$. The HSDPA scheduler with retransmissions is modeled as shown in Figure 4.6 below.

The evolution of the queue size (x_i) is given by

$$\begin{aligned} x_i(t+1) &= \min\left(\left[x_i(t) - y_i(t)\mu_i(t)\right]^+ + z_i(t+1)\,, B\right) \\ &= \min\left(\left[x_i(t) - \gamma_i(t)a_i(\boldsymbol{s})c\mu_i(t)\right]^+ + z_i(t+1), B\right) \end{aligned} \tag{4.14}$$

where $a_i(\boldsymbol{s})c$ is the number of codes allocated to user i when in state $\boldsymbol{s}$ and $\boldsymbol{a}(\boldsymbol{s}) = (a_1(\boldsymbol{s}), a_2(\boldsymbol{s}), \ldots, a_L(\boldsymbol{s}))$. Equation (4.14) means that PDUs removed from the head of the queue only when a transmission is successful, and remain there otherwise.

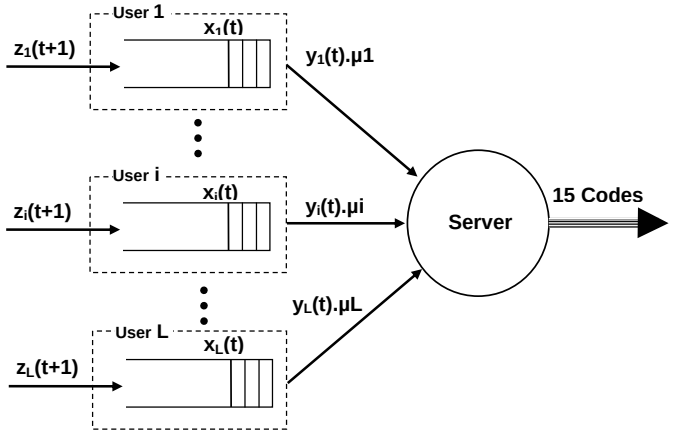


Figure 4.6: model for HSDPA downlink scheduler with retransmission

4.5.1 Reward Function

The reward function $R(\boldsymbol{s}, \boldsymbol{a})$ is similar as before and is given by

$$R(\boldsymbol{s}, \boldsymbol{a}) = \sum_{i=1}^{L}(y_i\mu_i) - \sigma\sum_{i=1}^{L}\Big[(x_i - \bar{x})\,\mathbb{1}_{\{x_i=B\}}\Big] \tag{4.15}$$

where $y_i = a_i c\gamma_i$. Similar as before, the objective is to maximize throughput while providing a fair allocations of resources (depending on the value of σ) to all users in the system.

4.5.2 Transition Probability Function

The MDP state transition probability $P_{\boldsymbol{ss'}}(\boldsymbol{a})$ can be formed as before, namely equations (4.7) and (4.9). In this case however, the queue state transition probability depends on μ_i. Using the total probability law, this probability can be rewritten as

$$\begin{aligned} P_{x_i x_i'}(\gamma_i, a_i) &\triangleq P(x_i'|x_i, \gamma_i, a_i) \\ &= P_{x_i x_i'|\mu_i=1}(\gamma_i, a_i)P(\mu_i = 1) + P_{x_i x_i'|\mu_i=0}(\gamma_i, a_i)P(\mu_i = 0) \\ &= P_{x_i x_i'|\mu_i=1}(\gamma_i, a_i)\cdot\xi + P_{x_i x_i'|\mu_i=0}(\gamma_i, a_i)\cdot(1-\xi) \end{aligned} \tag{4.16}$$

4.5.3 Queue State Transition Probability

The marginal queue state transition probabilities needed to find $P_{x_i x_i'}(\gamma_i, a_i)$ are given below (refer to Appendix E.3 for complete derivation). We remind the reader that B

is the buffer size for each user buffer in the model.

$$P_{x_i x_i'|\mu_i=1}(\gamma_i, a_i) = \begin{cases} 1 & \text{if } x_i' = x_i = B, \gamma_i a_i = 0 \\ q_i & \text{if } x_i' = x_i = B, 0 < y_i \leq u_i \\ q_i & \text{if } x_i' = B, x_i < B, [x_i - y_i]^+ + u_i \geq B \\ q_i & \text{if } x_i' < B, x_i' = [x_i - y_i]^+ + u_i \\ 1 - q_i & \text{if } x_i' < B, x_i' = [x_i - y_i]^+ \\ 0 & \text{Otherwise} \end{cases} \tag{4.17}$$

and,

$$P_{x_i x_i'|\mu_i=0}(\gamma_i, a_i) = \begin{cases} 1 & \text{if } x_i' = x_i = B \\ q_i & \text{if } x_i' = B, x_i < B, x_i + u_i \geq B \\ q_i & \text{if } x_i' < B, x_i' = x_i + u_i \\ 1 - q_i & \text{if } x_i' < B, x_i' = x_i \\ 0 & \text{Otherwise} \end{cases} \tag{4.18}$$

The overall queue transition probability $P_{x_i x_i'}$ can be determined by substituting equations (4.17) and (4.18) in (4.16).

4.6 The Heuristic Policy

In this section, we will present a heuristic approach for code allocation in our HSDPA model, by focusing on the cases where c =15, 5 or 3. We will utilize the information (regarding the structure of the optimal policy) we obtained from the results of Section 4.4.

4.6.1 Optimal Policy Structural Analysis and Weight Function Estimation

Analyzing the results obtained in Section 4.4.1 we observe that the optimal policy exhibits the following structural characteristics:

- When there is only one connected user then the optimal policy allocates all the 15 codes to that user or as many codes as required to empty its queue if its queue length is less than 15 PDUs.

- When both users are connected, then both are competing for the available 15 codes. From Figures 4.3–4.5 we observe the following trends:

    1. The optimal action a_1 (respectively a_2) is increasing in x_1 (respectively x_2).

    2. There is one switch-over region for every action vector. These regions are depicted by different colors (or shades) and labeled by their corresponding action vector in Figures 4.3–4.5.

    3. When $x_1 + x_2 \leq 15$, then the optimal policy allocates no more codes to a user than those required to empty its queue.

    4. In the symmetrical case and when $x_1 + x_2 > 15$, then the border(s) between neighboring regions is (are) a staircase function that can be approximated by a line with slope 1 (Figure 4.4(a)). Those border lines define parallel stripes that have a constant width.

    5. In the asymmetrical cases and when $x_1 + x_2 > 15$ (Figures 4.3–4.5(b)-(f)), the aforementioned regions, while still equidistant, are no longer delimited by linear borders. The approximate border slope as a function of x_1 and x_2 depends on the difference in arrival rates ΔP_z and the difference in the

connectivities ΔP_γ. For example the approximate slope (the dotted line) in Figure 4.3 is: 1 in the symmetrical case (Figure 4.3(a)), less than 1 when $\Delta P_\gamma = 0.3$ (Figure 4.3(b)) and more than 1 when $\Delta P_z = 0.3$ (Figure 4.3(c)).

From observation (4) above, we can approximate each of the switch-over boundary lines between the policy switch-over regions for the symmetrical case by a straight line with slope 1. This approximation is a good fit in the symmetrical case as shown in Figures 4.3-4.5(a). To extend this approximation to the asymmetrical cases, we introduce the weight vector $\mathbf{w} = (w_1, w_2)$, where $w_i, i = 1, 2$ is a function of ΔP_z and ΔP_γ. Then the boundary lines between the different regions can be linearly approximated by the following equation:

$$w_1 x_1 = w_2 x_2 + C \tag{4.19}$$

where C is a constant. Observations of Figures 4.4 and 4.5 suggest that C is a multiple of $\pm 2c$ ($C = \pm 2ck, k = 0, 1, \ldots, 0.5(15/c - 1)$). Equation (4.19) defines a family of lines that have a slope equal to w_2/w_1. For the symmetrical case, $w_1 = w_2 = 1$ and Equation (4.19) will be reduced to $x_1 = x_2 + C$.

The weight w_1 (respectively w_2) is increasing (respectively decreasing) in ΔP_z and decreasing (respectively increasing) in ΔP_γ (the reader may verify this from the figures, for example Figures 4.3(b) and 4.3(c)). We observed the behavior of the optimal policy under different arrivals and connectivity parameters, by solving the dynamic programming equation (Equation 4.11) numerically for these cases. The results obtained were analogous to the ones in Figures 4.4-4.5. Following these observations,

we approximated w_1 and w_2 as follows

$$w_1 \approx 1 + 1.5[-\Delta P_\gamma]^+ - 0.7[-\Delta P_z]^+ \quad (4.20)$$

$$w_2 \approx 1 + 1.5[\Delta P_\gamma]^+ - 0.7[\Delta P_z]^+ \quad (4.21)$$

The coefficients for ΔP_γ and ΔP_z were chosen empirically based on a variety of computations (value iterations to determine the optimal policy) under different channel connectivity and arrival parameters.

The linear approximation may not be perfect for asymmetric cases (see for example Figures 4.3-4.5(b) and (c)). A non-linear approximation would result in a better fit in those cases (and may be a future research problem). However, (as we will show in Section 4.7.3) the heuristic scheduling policy, resulted from our linear approximation, compares favorably (in throughput and fairness/queuing delay terms) with the optimal policy.

4.6.2 Detailed Characterization of The Heuristic Policy

1. Case for $c = 15$. In this case, the optimal policy is a switch-over policy as depicted in Figure 4.3. We can identify three regions which correspond to the three possible actions: (0,0), (1,0) and (0,1). The heuristic policy is a weighted LCQ and it assigns codes to users according to the following rules:

   - Rule1: when there is only one connected user then assign all the needed codes to that user.
   - Rule2: when both users are not connected (i.e., $\gamma_1 = \gamma_2 = 0$) then no codes will be allocated to any user.
   - Rule3: when both users are connected, allocate code chunks according to

(4.22) below

$$\boldsymbol{a}(t) = \begin{cases} (1,0) & \text{if } w_1 x_1 > w_2 x_2, \\ (0,1) & \text{if } w_1 x_1 \leq w_2 x_2 \end{cases} \tag{4.22}$$

2. Case for $c = 5$. The optimal policy defines ten regions specifying the optimal code allocation. However, only four of these regions are of interest. They lie within the area where the demand exceeds the available resources as shown in Figure 4.4. Based on this observation, the heuristic policy partitions the state space into four major regions that correspond to the actions (3,0), (2,1), (1,2), and (0,3). The policy rules 1 and 2 are the same as before. Rule3 is modified as follows:

   - Rule3: when both users are connected, *if* $x_1 + x_2 < 15$ *then* allocate codes to the two users in proportion to their queue length, *else* allocate the code chunks as follows

$$\boldsymbol{a}(t) = \begin{cases} (3,0) & \text{if } w_1 x_1 > w_2 x_2 + 10, \\ (2,1) & \text{if } w_2 x_2 < w_1 x_1 \leq w_2 x_2 + 10, \\ (1,2) & \text{if } w_2 x_2 - 10 \leq w_1 x_1 \leq w_2 x_2, \\ (0,3) & \text{if } w_1 x_1 < w_2 x_2 - 10, \end{cases} \tag{4.23}$$

3. Case for $c = 3$. There are 21 different regions in the state space as shown in Figure 4.5. The heuristic rules used earlier can be extended to this case. Again only Rule3 needs to be modified as shown below:

   - Rule3: when both users are connected, *if* $x_1 + x_2 < 15$ *then* allocate codes to the two users in proportion to their queue length, *else* allocate the code

chunks as follows

$$\boldsymbol{a}(t) = \begin{cases} (5,0) & \text{if } w_1x_1 > w_2x_2 + 12, \\ (4,1) & \text{if } w_2x_2 + 6 < w_1x_1 \leq w_2x_2 + 12, \\ (3,2) & \text{if } w_2x_2 < w_1x_1 \leq w_2x_2 + 6, \\ (2,3) & \text{if } w_2x_2 - 6 < w_1x_1 \leq w_2x_2, \\ (1,4) & \text{if } w_2x_2 - 12 < w_1x_1 \leq w_2x_2 - 6, \\ (0,5) & \text{if } w_1x_1 \leq w_2x_2 - 12, \end{cases} \tag{4.24}$$

Figures 4.3-4.5(a)-(c) show the heuristic policy (the dotted line) superimposed on the optimal policy from Section 4.4 for different loading and channel quality conditions. From these figures, it is fair to say that the heuristic policy reasonably approximated the slope of border lines between the different regions of the optimal policy.

Motivated by the heuristic policy construction for the two-user case, we can extend the methodology to more than two users.

4.6.3 Extended Heuristic Policy

The optimal policy for three or more users still has a switch-over structure. However, it is not possible to visualize it on a two dimensional plane (e.g., for the case of three users, the possible action can be described by a three-dimensional vector, i.e., $\boldsymbol{a}(\boldsymbol{s}) = (a_1, a_2, a_3)$, therefore, it is not possible to pictorially visualize the structure of the policy in this case). Motivated by the heuristic policy construction for the two-user case, we can extend this heuristic policy (presented in the previous subsection) to any finite number of users. We can do this by ordering all users in the system according to their weighted queue lengths. The weight will be a function of the different users'

arrival probabilities and channel states. We can then allocate the available code chunks to the connected queues according to their weighted queue length using rules analogous to those in the previous section. The added complexity of this extension (compared to the heuristic policy we presented in the previous section) is minimal and is mainly due to the user ordering phase of the algorithm.

4.7 Performance Evaluation

In this section, we study the performance of the optimal policy and compare it with that of the devised heuristic policy. We also compare both policies with the Round Robin fair queuing, keeping the same assumptions as before. The evaluation is based on the simulation of several cases with constant buffer size $B = 50$. All simulations were performed using the C programming language. The simulation ran for 50,000 time slots and 10,000 replications. The 95% confidence interval (CI) was computed for all simulation results. For clarity of presentation, only the CI's with maximum width among simulated performances for each offered load (corresponding to a point on the x-axis) are plotted at the top of each figure.

4.7.1 The Effect of Code Allocation Granularity

The total number of codes available in one TTI is 15 codes according to [43] and in our approach, the scheduler allocates chunks of these codes to active users. The chunk size c was introduced in Section 4.3.1. It ranges from the finest (when $c = 1$) to the coarsest (when $c = 15$); in the former case, the policy can assign as little as 1 code to a user at a time, while in the latter case, all the 15 codes are assigned to one single user at a time. Figures 4.7-4.10 show the effect of the chunk size c on the system performance for various offered load $\rho = \sum_i P_{z_i} u_i / r_\pi$ with r_π is the measured system capacity under the policy π. The channel state probability for the two users

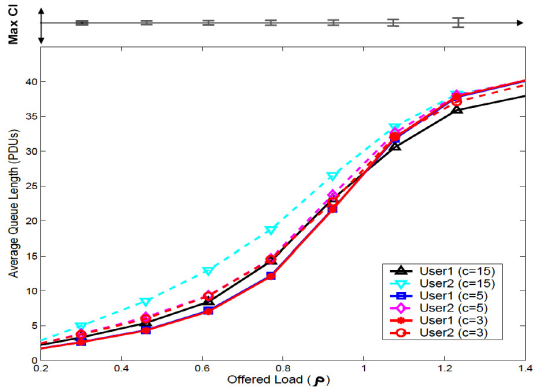


Figure 4.7: The effect of policy granularity on queue length.

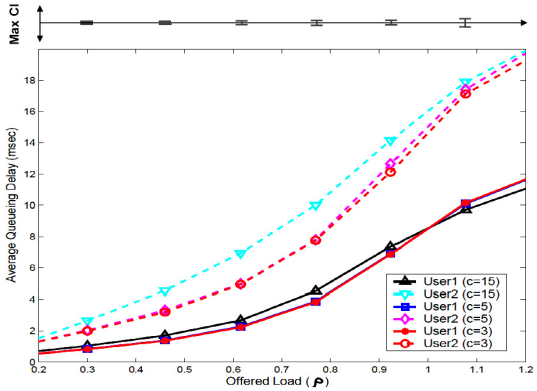


Figure 4.8: The effect of policy granularity on the queuing delay.

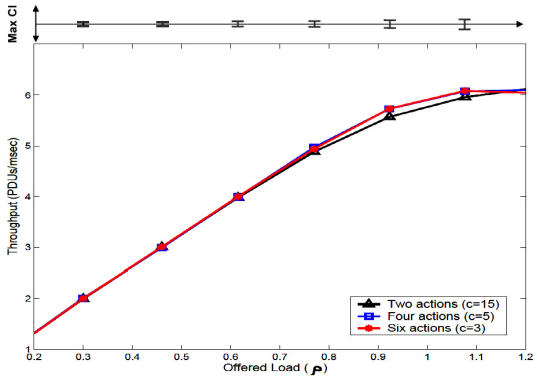


Figure 4.9: The effect of policy granularity on scheduler throughput.

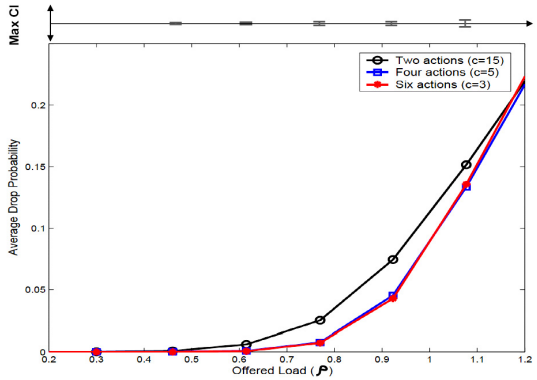


Figure 4.10: The effect of policy granularity on scheduler drop prob.

is set to $P(\gamma_1 = 1) = 0.8$ and $P(\gamma_2 = 1) = 0.5$. Using (4.13), the channel model parameters (α_i and β_i) for each user were selected as follows

$$P_1 = \begin{bmatrix} 0.4 & 0.6 \\ 0.15 & 0.85 \end{bmatrix}, \quad \text{and} \quad P_2 = \begin{bmatrix} 0.4 & 0.6 \\ 0.6 & 0.4 \end{bmatrix}$$

The results show that in light and moderate load conditions, the average queue length is shorter when the assigned code chunks have finer granularity. However, when $\rho \to 1$ the difference becomes smaller (and within the confidence interval) when ρ becomes greater or equal to 1.

Another observation is that the performance gain when moving from $c = 5$ to $c = 3$ is less significant and may not justify the added implementation and computational cost. It is interesting to see that the optimal policy under all of the three values of c achieved approximately the same throughput (see Figure 4.9). The slight throughput loss when $c = 15$ in moderate to high load is within the confidence interval and is due to the increased drops under these particular conditions as it is shown in Figure 4.10. The intuition behind this behavior can be summarized as follows:

1. For light loads ($\rho < 0.7$), the chunk size (c) does not have a significant effect on the dropping probability in the two queues since the drops are unlikely in this case (Figure 4.10). Therefore, the optimal policy for any value of c achieves throughputs that are not substantially different.

2. As the load increases ($\rho = 0.7 \sim 1.1$) the drops become *more likely* to occur. A coarser code allocation granularity ($c = 15$) will serve one queue only at a given TTI thus resulting in more drops for the other, while a finer granularity ($c = 5$) would keep both queue sizes balanced (thus resulting in a lower total loss).

3. For higher load ($\rho > 1.1$), the losses escalate dramatically and chunk granularity has no significant effect on drops between the two queues in this case.

4.7.2 The Effect of Channel Model

The performance of the optimal policy, when using a 3-state FSMC channel model, is evaluated using simulation and compared to the 2-state FSMC model. The channel models used are

- Two-state FSMC with $P(\gamma_1 = 1) = 0.8$, $P(\gamma_2 = 1) = 0.5$.

- Three-state FSMC with $P(\gamma_1 = 1) = 0.4$, $P(\gamma_1 = 2) = 0.4$, $P(\gamma_2 = 1) = 0.25$ and $P(\gamma_2 = 2) = 0.25$.

The two cases above are analogous in the sense that in both cases user 1 channel (respectively user 2 channel) is connected with probability $P(\gamma_1 \geq 1) = 0.8$ (respectively $P(\gamma_2 \geq 1) = 0.5$). To achieve this, the 3-state FSMC model parameters for each user were selected as follows

$$P_1 = \begin{bmatrix} 0.4 & 0.6 & 0.0 \\ 0.3 & 0.3 & 0.4 \\ 0.0 & 0.4 & 0.6 \end{bmatrix}, \quad \text{and} \quad P_2 = \begin{bmatrix} 0.8 & 0.2 & 0.0 \\ 0.4 & 0.3 & 0.3 \\ 0.0 & 0.3 & 0.7 \end{bmatrix}$$

Figures 4.11 and 4.12 show the system throughput and the drop probability versus average arrival rate ($\sum_i P_{z_i} u_i$ for all $i \in \{1, 2\}$). In the 2-state model, the system reaches its capacity at about 6.5 $PDUs/sec$ (Figure 4.11). For the case of a 3-state FSMC model, the saturation performance is better since more PDUs can be transmitted on average (compared to the 2-state FSMC model) when connected (the FSMC model is presented in Section 4.3.2), for the given state transition probabilities.

Figures 4.13 and 4.14 depict the average queue length behavior for both users as a function of the arrival rate; we can see that when the fairness factor $\sigma = 0.5$ the policy keeps almost equal queue occupancy for both users. On the other hand, when $\sigma = 0.0$ the difference in the average queue lengths of the two users is more than 10 PDUs. This will result in unfairness and increased losses for user 2 traffic. In this case, the queuing delay increases and results in poor delay performance for user 2 as shown in Figure 4.15.

The fairness factor (σ), that did not have an effect in the 2-state case, has a significant effect when using the 3-state FSMC model as seen in Figures 4.13 and 4.14. When the load increases, the optimal policy with $\sigma = 0.0$ achieves higher system throughput compared to that with $\sigma = 0.5$. This agrees with intuition since

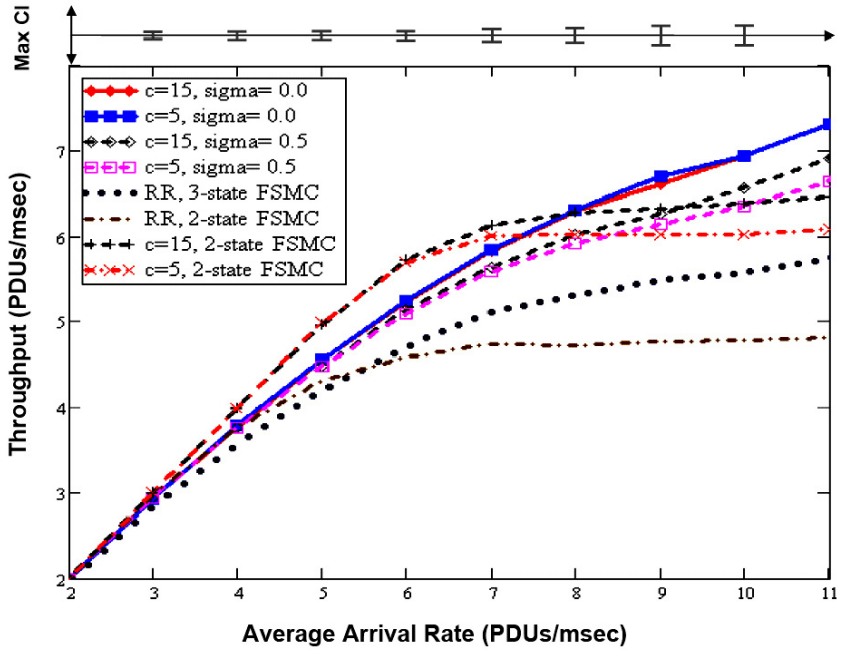


Figure 4.11: System throughput vs. average arrival rate to the system. First four cases in legend corresponds to the optimal policy with 3-state FSMC model.

the policy corresponding to $\sigma = 0.0$ aims at optimizing the throughput at the expense of fairness. Higher σ will result in more fairness at the cost to the overall system throughput as shown in Figures 4.13 and 4.15.

Remark: The 3-state FSMC model obviously is more accurate representation of the HSDPA downlink transmission system than the 2-state model. In 3GPP R'5 standard [43] there are 31 transmission levels (TFRC) which corresponds to 32 channel states. However, in real life only a subset of that range (usually 6 TFRCs) is used (see [46] for an example). □

4.7.3 Heuristic Policy Evaluation

The system throughputs under the heuristic policy we developed in Section 4.6, the optimal and Round Robin policies are shown in Table 4.1 for different loading conditions. The channel model parameters were chosen (2-state FSMC channel is assumed)

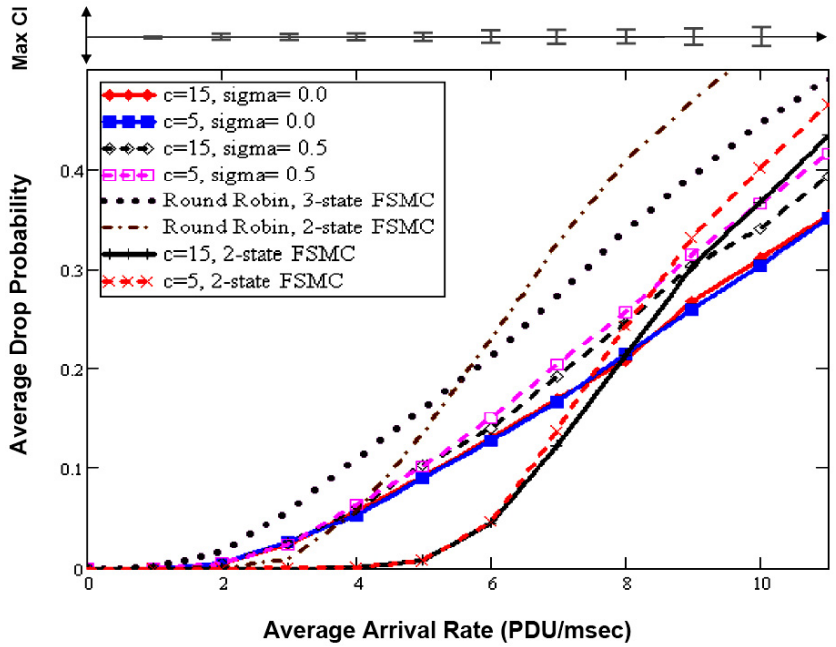


Figure 4.12: Average drop probability (average dropped/average arrived PDUs). First four cases in legend corresponds to the optimal policy with 3-state FSMC model.

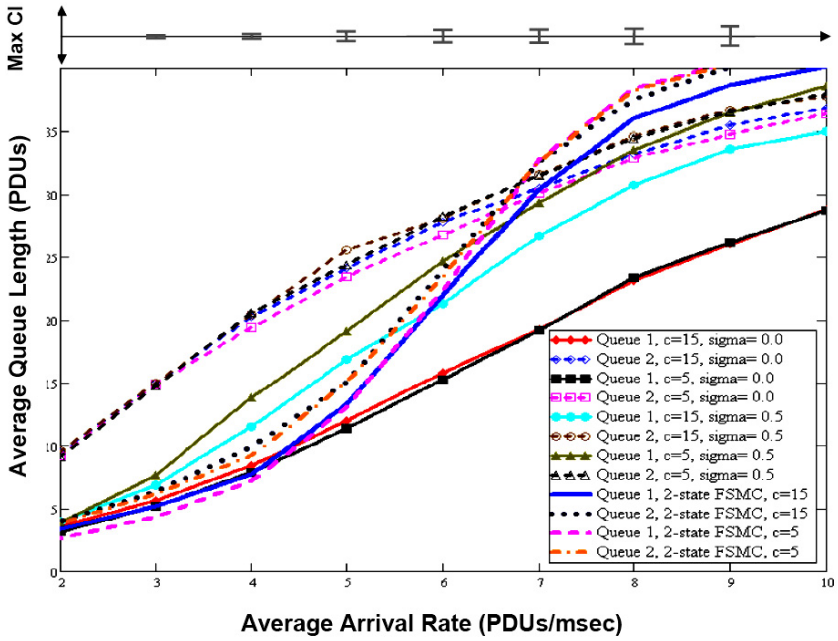


Figure 4.13: Average queue length vs. load compared to the 2-state FSMC model. First eight cases in legend corresponds to the optimal policy with 3-state FSMC model.

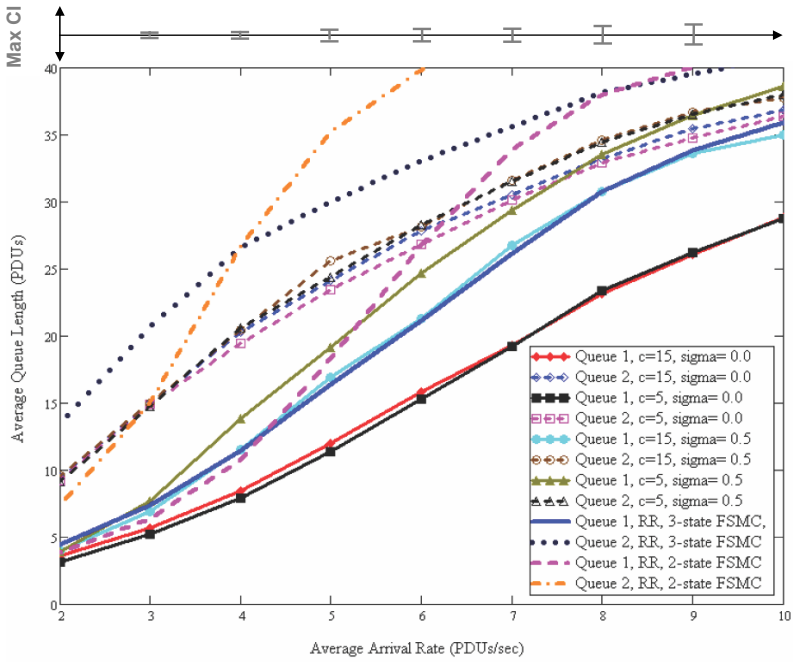


Figure 4.14: Average queue length vs. load compared to Round Robin scheduling. First eight cases in legend corresponds to the optimal policy with 3-state FSMC model.

such that $P(\gamma_1 = 1) = 0.8$ and $P(\gamma_2 = 1) = 0.5$. Table 4.1 shows that the throughput performance of the heuristic policy is very close to that of the optimal policy. It also shows that RR performance is close to that of the optimal policy in case of light loading (e.g., $\rho = 0.5$). However, RR performs around 27% worse than the optimal policy in heavy load conditions when $\rho = 1.2$.

The queuing delay performance of the heuristic policy is compared with that of

Table 4.1: System throughput (PDUs/msec) for different policies and loading conditions.

ρ	*Optimal*	*Heuristic*	*RoundRobin*
0.5	3.25	3.25	3.25
0.8	5.0	4.95	4.45
1.2	6.73	6.71	4.9

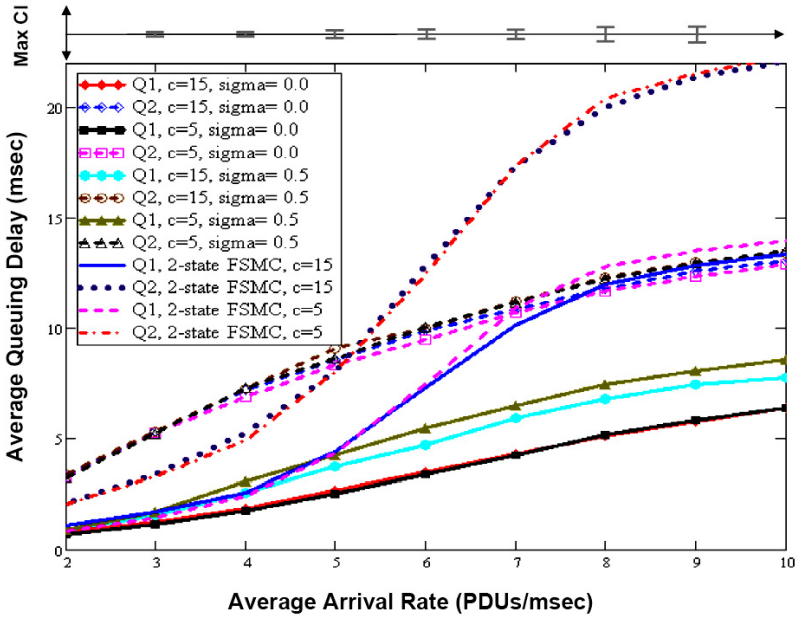


Figure 4.15: Average queuing delay vs. load; 3-state FSMC model compared to 2-state FSMC. First eight cases in legend corresponds to the optimal policy with 3-state FSMC model.

Table 4.2: Queuing delay performance (in milliseconds) for different policies, $P(\gamma_1 = 1) = 0.8$, $P(\gamma_2 = 1) = 0.5$, $q_1 = 0.8$, $q_2 = 0.5$ and $u = 10$.

	Optimal	*Heuristic*	*RoundRobin*
$User1$	8.0	7.4	13.6
$User2$	10.8	12.2	19.9
Difference	2.8	4.8	6.3

Table 4.3: Queue length (PDUs), $\rho = 0.75$, $P(\gamma_1 = 1) = 0.8$, $P(\gamma_2 = 1) = 0.5$, $q_1 = 0.5$, $q_2 = 0.5$ and $u = 10$.

	Optimal	*Heuristic*	*RoundRobin*
$User1$	11.5	11.0	16.5
$User2$	14.5	16.0	34.5
Difference	3.0	5.0	18.0

Table 4.4: Queue length (PDUs), $\rho = 1.1$, $P(\gamma_1 = 1) = 0.6$, $P(\gamma_2 = 1) = 0.6$, $q_1 = 0.8$, $q_2 = 0.5$ and $u = 10$.

	Optimal	*Heuristic*	*RoundRobin*
$User1$	33.0	33.5	45.5
$User2$	28.5	27.5	32.5
Difference	4.5	6.0	13.5

RR and the optimal policy in Table 4.2. Queue lengths for the two users are shown in Tables 4.3 and 4.4 for different channel and arrival parameters. From these tables, the following conclusions can be drawn:

- The proposed heuristic policy compares favorably to the optimal one in terms of throughput and delay performance.

- The optimal policy provides better fairness in comparison to the heuristic and RR policies. This is apparent from the simulation results since the optimal policy achieved the smallest difference between the two users' queuing delay. The heuristic policy provides a comparable performance to that of the optimal policy, while the round robin is the least fair policy among the three.

- The difference in queue lengths of the two users resulted from using the heuristic policy is reasonably close to that of the optimal policy under different channel and arrival parameters (Tables 4.3 and 4.4). RR policy resulted in the largest queue lengths (for the two users) compared to the optimal and heuristic policies.

- The performance of the RR policy is highly dependent on the loading conditions. The results obtained proved that RR has poor performance in wireless channel. A reason for the poor RR performance is that it does not take into account the wireless channel quality variation, while the optimal policy depends on the state of the channel.

Chapter 5

Analytic Evaluation of Downlink Service Rate in 3G Wireless Networks

In this chapter, we use stochastic modeling to develop a server sharing model for the downlink scheduler in 3G wireless networks. This model is used to find the average achievable service rate per user for a given fairness criterion. The wireless channel is modeled by a Finite State Markov Channel (FSMC) to reflect the effect of Adaptive Modulation and Coding used in 3G wireless networks. This model can be used to implement differentiated services in 3G wireless networks. It facilitates QoS control on the last hop, i.e., the wireless link. This methodology provides qualitative and quantitative evaluation of the wireless link sharing mechanisms in 3G wireless networks.

5.1 Introduction

Third generation (3G) wireless networks are IP-based networks that can provide high-speed packet access to their mobile users. Their IP-based infrastructure enables them to benefit from the mature and widely available IP technologies. It also makes available the wide variety of services that an IP-based network can provide to its

users, e.g., web access, file transfer, video streaming and voice over IP. There are two popular categories in 3G wireless networks: Wide-band CDMA (as described by third generation partnership project, 3GPP) and Multi-Carrier CDMA (as described by 3GPP2). An example of the first type is the High Speed Downlink Packet Access (HSDPA) [43]. This network can provide asymmetric data access to its users with a peak downlink rate of 14.4 Mbps (for R'5). An example of MC-CDMA is the CDMA2000 EVDO (EVolution Data Optimized) which is a packet-based network that can provide asymmetric access to its users with a maximum rate of 2.4 Mbps per carrier on the downlink.

From the capacity modeling and scheduling points of view, the third generation wireless networks (and their successors, 4G and beyond) have several characteristics that differentiate them from other networks, mainly, channel variability, resource multiplicity and data rate adaptation. Therefore, the previous models that were used to evaluate service rate and QoS scheduler performance in other networks are no longer sufficient or accurate for the modeling of 3G networks. Therefore, the effort to study the effect of channel (and consequently the data rate per user) variability on the achievable average service rate in emerging wireless networks is well justified. Most of the work done in this area was based on simulation and was limited to specific working scenarios. However, there were also attempts to model wireless access networks, some of which are directly or indirectly related to 3G networks, e.g., [23], [25], [24], [27] as well as the work we presented in the previous chapters of this thesis. In that work, we presented and analyzed an analytical model of the emerging wireless networks (including 3G networks). We mainly studied the structure (or structural characteristics) of the optimal *dynamic* packet scheduling policy in these networks. Although that work provided an important insight to the behavior of the optimal packet scheduling policy in emerging wireless networks, it did not (and was not intended to) compute the average achievable data rate or the maximum QoS

provision that can be met by these networks.

In this part of our work, we develop an analytical model consisting of a set of equations to describe downlink channel sharing in 3G wireless systems. This set of equations is then solved to determine the long-run average share of resources required to meet specific fairness/QoS requirements of users sharing a wireless link in a 3G wireless network. The significance of this methodology rises from its inherent simplicity (computation wise) and its direct approach which make it easy to understand and to deduct conclusions when compared to previously used techniques. The model can be used to represent different service classes competing for the resources in a 3G wireless system.

The methodology and the results presented in this chapter can be used as a benchmarking tool for the dynamic scheduling policies (e.g., to measure how well they achieve their long-run expected throughput).

5.2 System Description and Modeling

The model we present here is an abstraction of the downlink scheduler in 3G wireless access networks. The two main types presented earlier (WCDMA and MC-CDMA) have some subtle differences; however, as much as this model is concerned, they both have the same channel sharing principle where the network resources (codes and/or time slots per cell/sector) have to be shared by the users in that cell. In the following, we will concentrate on the WCDMA HSDPA system as an example of 3G networks. Nevertheless, most of the descriptions are parallel to that of MC-CDMA system.

Radio Access Network (RAN) in HSDPA system consists of Radio Network Controllers (RNC), each of which is connected to one or more Base Stations (BTS) (Figure 5.1). Serving and Gateway GPRS Support Nodes (SGSN and GGSN) are network nodes that support the use of GPRS in the GSM core network. Each BTS resides

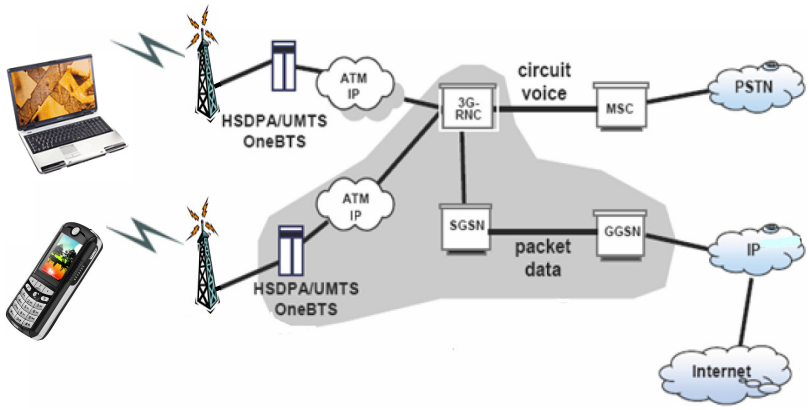


Figure 5.1: 3G WCDMA Network

in the center of a cell in the coverage space of a cellular network. There are one or more sectors per cell each of which uses a single, time-shared channel called HS-DSCH. The system uses a combination of Code and Time Division Multiple Access (CDMA/TDMA). The high-speed downlink shared data channel has 15 codes to be allocated to one or more users every 2 milliseconds. In this system, AMC is used to adapt the transmission rate of a scheduled user to its expected channel conditions in the next transmission time interval. This is done by monitoring the user's signal to noise ratio and mapping it to a set of Modulation and Coding Schemes (MCS) [61]. The higher the order of the selected MCS, the higher the transmission rate. Hence, the user's service rate will depend on his channel conditions and the implemented scheduling policy.

5.2.1 Wireless Channel Model

The system is assumed to have Rayleigh fading channel. Similar to the model in Chapter 4, we model the wireless channel in this system by a Finite-State Markov Channel (FSMC) as shown in Figure 4.2. The states in this channel model represent data rate levels. The channel state of user i during time slot t is denoted by $\gamma_i(t)$, and its associated channel state space is the set $\mathcal{N} = \{0, 1, \ldots, N-1\}$, where N is

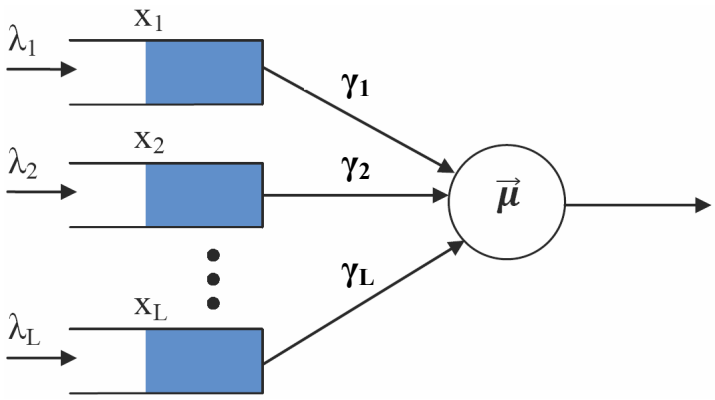


Figure 5.2: A model for L users sharing one HSDPA downlink channel

the total number of available channel states.

5.2.2 A Model for The Downlink Scheduler

A system with L users is modeled by L queues competing for the service of a single server with a base capacity C. C corresponds to the minimum achievable transmission rate when using the lowest order MCS (i.e., $\gamma_i(t) = 1$). The server capacity can be shared, at any given time slot, by the connected users in the system. The achievable capacity for each user depends on the user's current channel condition and directly proportional to $\gamma_i(t)$. We refer to the vector $\vec{\gamma} = [\gamma_1, \gamma_2, \ldots, \gamma_L]$ as the *channel state* vector, where we suppressed the time index to simplify the notation. Each queue is *connected* to the server (i.e., $\gamma_i > 0$) with probability $1 - \pi_{i,0} : i \in I$ and *not-connected* ($\gamma_i = 0$) with probability $\pi_{i,0}$, where $I = \{1, 2, \ldots, L\}$ is the set of all queues in the system and $\pi_{i,j}$ is the probability that user i FSMC (i.e., channel condition) is in state j. Figure 5.2 shows the basic model described above.

5.2.3 Basic Assumptions and Definitions

In the following, we make some basic assumptions in order to facilitate the modeling process and to make the model analytically tractable.

1. Symmetrical arrival processes with arrival rates ($\alpha_i = \alpha$ for all $i \in I$). We assume the system is operating under heavy loading conditions.

2. Channels for different users are independent, i.e., γ_i independent of γ_j for all $i, j \in I, i \neq j$.

3. The achievable transmission rate for queue i (i.e., when the full server capacity is allocated to user i) at time t is $\gamma_i(t)C$.

The first assumption is a valid assumption for heavily-loaded symmetrical systems. The second assumption is also a valid one when the system uses orthogonal code multiple access. The last assumption (i.e., the assumed linear relationship between user's channel state and its data rate) is a simplifying one and it can be relaxed later. We define the following:

- x_i the length of queue i at time t;
- μ_i the average service rate experienced by queue i;
- $\vec{\boldsymbol{\mu}} = [\mu_1, \mu_2, \ldots, \mu_L]$;
- $m_i \in [0, 1]$ the average share of the server assigned to queue i;
- $\vec{\boldsymbol{\pi}}_i = [\pi_{i,0}, \pi_{i,1}, \ldots, \pi_{i,N-1}]$, where $\pi_{i,j}$ is the probability that user i's channel is in state j.

The service rate received by queue i at any given time depends on its channel state. Obviously when $\gamma_i = 0$, the queue is not connected and no server share will be assigned to it. On the other extreme, when only one queue is connected (i.e., $\gamma_i = 1$ and $\gamma_j = 0$ for all $j \in I \backslash \{i\}$) then all the server capacity will be assigned to that queue.

5.3 Service Rate Analysis

In this section we will derive an expression that relates the required server share (m_i) of each user to its channel state stationary distribution ($\vec{\boldsymbol{\pi}}_i$) and the targeted service differentiation between the users in the system. This is done by deriving a set of equations describing the average service rate per user, then solving this set of equations to find the service sharing regime that will achieve fairness or given QoS requirements. We will start by modeling a simple system with 2 users, each of which has a 2-state channel. This step is necessary because in this case it is easier to understand the argument used in the derivation of the average service rate equations. The second step is to extend this model for all values of L and N.

5.3.1 Two-User System with 2-State FSMC

We begin with a simple, 2-user system ($L = 2$) with 2-state channel model ($N = 2$). The system is modeled by two queues sharing one server. Queue 1 and queue 2 are connected with probabilities $\pi_{1,1}$ and $\pi_{2,1}$ respectively and not connected otherwise. The average service rates for the two queues were derived using the following argument:

Queue i can be served only if its channel is in a connected state, i.e., $\gamma_i > 0$. In this case, if user 1 is connected and user 2 is not then the entire server capacity will be assigned to serve queue 1. The probability of this event happening is $\pi_{1,1}\pi_{2,0}$, since we assumed independent FSMCs for different users. When both users are connected (with probability $\pi_{1,1}\pi_{2,1}$), then they will share the server according to the adopted scheduling policy. Hence, the average service rate for queue 1 is given by,

$$\begin{aligned} \mu_1 &= C(\pi_{1,1}\pi_{2,0} + m_1\pi_{1,1}\pi_{2,1}) \qquad \text{packets per second} \\ &= C(\pi_{1,1} - \pi_{1,1}\pi_{2,1} + m_1\pi_{1,1}\pi_{2,1}) \end{aligned} \tag{5.1}$$

where the second line in (5.1) above is obtained using the fact that $\pi_{2,0} + \pi_{2,1} = 1$

Using the same argument we used for queue 1 above, the service rate for queue 2 is give by

$$\begin{aligned} \mu_2 &= C(\pi_{2,1}\pi_{1,0} + m_2\pi_{1,1}\pi_{2,1}) \qquad \text{packets per second} \\ &= C(\pi_{2,1} - \pi_{1,1}\pi_{2,1} + m_2\pi_{1,1}\pi_{2,1}) \end{aligned} \tag{5.2}$$

To solve for m_1 and m_2, we assume differentiated service with rate ν. Such a policy will assign service to both queues according to: $\mu_1 = \nu\mu_2$. Using (5.1) and (5.2) we have

$$m_1 - \nu m_2 = (1-\nu) - \frac{\pi_{1,1} - \nu\pi_{2,1}}{\pi_{1,1}\pi_{2,1}} \tag{5.3}$$

The second equation required is

$$m_1 + m_2 = 1 \qquad \textit{where} \quad 0 \le m_i \le 1 \tag{5.4}$$

since we cannot allocate more that 100% of the server resources. Solving (5.3) and (5.4) for m_1 and m_2 yields

$$m_1 = \frac{1}{1+\nu}(1 - \frac{\pi_{1,1} - \nu\pi_{2,1}}{\pi_{1,1}\pi_{2,1}}) \tag{5.5}$$

$$m_2 = \frac{1}{1+\nu}(\nu + \frac{\pi_{1,1} - \nu\pi_{2,1}}{\pi_{1,1}\pi_{2,1}}) \tag{5.6}$$

Substituting m_1 and m_2 in (5.1) and (5.2) above, we get

$$\mu_1 = \frac{C\nu}{1+\nu}(\pi_{1,1} + \pi_{2,1} - \pi_{1,1}\pi_{2,1}) \tag{5.7}$$

$$\mu_2 = \frac{C}{1+\nu}(\pi_{1,1} + \pi_{2,1} - \pi_{1,1}\pi_{2,1}) \tag{5.8}$$

To achieve fairness in the 2-user case, we choose $\nu = 1$. In this case, the network resources will be distributed in such a way that both users will receive the same average service rate, i.e., $\mu_1 = \mu_2$. On the other hand, to implement QoS, one can use a parameter $\nu \neq 1$ to reflect the relative priority between the two users/classes (this model may be used to represent two classes of service rather than two users). Then the amount of service that each user must receive in order to achieve the required QoS is given by (5.7) and (5.8). The policy used to allocate the server resources to both users must assign all the resources to the connected user, when there is one connected user, and divide the resources between the two users according to (5.5) and (5.6) when both users are connected.

Fair Scheduler

A special and very important case of the previous model is the fair scheduler, when $\nu = 1$, (i.e., $\mu_1 = \mu_2$). In this case, The server is shared by the two users as follows

$$m_1 = \frac{1}{2}(1 - \frac{\pi_{1,1} - \pi_{2,1}}{\pi_{1,1}\pi_{2,1}}) \tag{5.9}$$

$$m_2 = \frac{1}{2}(1 + \frac{\pi_{1,1} - \pi_{2,1}}{\pi_{1,1}\pi_{2,1}}) \tag{5.10}$$

Then (5.7) and (5.8) will be reduced to

$$\mu_1 = \mu_2 = \frac{C}{2}(\pi_{1,1} + \pi_{2,1} - \pi_{1,1}\pi_{2,1}) \tag{5.11}$$

This formula can be used to determine the average service rate experienced by each user when applying a fair scheduling policy. It can also be used to determine the fairness levels of different scheduling policies by comparing the service rate experienced by each user to that in Equation (5.11).

Equal Shares Scheduler

Another special case is when $m_1 = m_2 = 1/2$. This case depicts the policy that divides the resources equally between the two users when connected. Hence, we call this case *equal shares scheduler*. An important example of such policy is the round robin scheduler (RR). In wire-line networks, RR is considered a fair scheduler. To see how the performance of such scheduler is affected by the randomness of the channel, we derive the average service rates for both users. In this case, the service rates are given by:

$$\mu_1 = C\pi_{1,1}(1 - \frac{\pi_{2,1}}{2}) \tag{5.12}$$

$$\mu_2 = C\pi_{2,1}(1 - \frac{\pi_{1,1}}{2}) \tag{5.13}$$

From Equations (5.12) and (5.13) we can see that the policy will result in a service differentiation that depends on the quality of the two users' channels. The user with better channel quality will experience higher service rate than the other user. This policy is fair only when $\pi_{1,1} = \pi_{2,1}$.

5.3.2 Extension to L Users and N-State FSMC

In order to extend the model we presented in the previous section to any number of users and any number of channel states, we need to define the following sets

- I as defined in section 5.2.
- $\mathcal{M}^{(n,i)} \subseteq I$ is a subset of I that contains the element $\{i\}$ plus n other elements of I.

We will extend the formulas we obtained in the 2-users case by using the same argument we used in 5.3.1. Again the independent connectivity assumption is used

here. We also use the assumption that "the achievable transmission rate for queue i is $\gamma_i(t)C$". The average service rate for user i is given by

$$\begin{aligned}
\mu_i &= C\Bigg[\sum_{r=1}^{N-1} r\pi_{i,r} \prod_{l\in I\backslash\{i\}} \pi_{l,0} \\
&+ \sum_{\forall \mathcal{M}^{(1,i)}\subset I}\Bigg[\frac{m_i}{\sum_{j\in\mathcal{M}^{(1,i)}} m_j}\cdot\sum_{r=1}^{N-1} r\pi_{i,r} \prod_{k\in\mathcal{M}^{(1,i)}\backslash\{i\}}\left(\sum_{r=1}^{N-1}\pi_{k,r}\right)\prod_{l\in I\backslash\mathcal{M}^{(1,i)}}\pi_{l,0}\Bigg] \\
&+ \sum_{\forall \mathcal{M}^{(2,i)}\subset I}\Bigg[\frac{m_i}{\sum_{j\in\mathcal{M}^{(2,i)}} m_j}\cdot\sum_{r=1}^{N-1} r\pi_{i,r} \prod_{k\in\mathcal{M}^{(2,i)}\backslash\{i\}}\left(\sum_{r=1}^{N-1}\pi_{k,r}\right)\cdot\prod_{l\in I\backslash\mathcal{M}^{(2,i)}}\pi_{l,0}\Bigg]+\ldots+ \\
&+ \sum_{\forall \mathcal{M}^{(n,i)}\subset I}\Bigg[\frac{m_i}{\sum_{j\in\mathcal{M}^{(n,i)}} m_j}\cdot\sum_{r=1}^{N-1} r\pi_{i,r} \prod_{k\in\mathcal{M}^{(n,i)}\backslash\{i\}}\left(\sum_{r=1}^{N-1}\pi_{k,r}\right)\prod_{l\in I\backslash\mathcal{M}^{(n,i)}}\pi_{l,0}\Bigg]+\ldots+ \\
&+ \frac{m_i}{\sum_{j\in I} m_j}\sum_{r=1}^{N-1} r\pi_{i,r}\prod_{k\in I\backslash\{i\}}\left(\sum_{r=1}^{N-1}\pi_{k,r}\right)\Bigg], \qquad \forall\, i\in I,\, n<L-1
\end{aligned} \tag{5.14}$$

Using a service criterion such as $\mu_1 = \nu_2\mu_2 = \ldots = \nu_L\mu_L$ will result in service differentiation between the L users with parameters $(\nu_2, \nu_3, \ldots, \nu_L)$. Using $\nu_2 = \nu_3 = \ldots = \nu_L = 1$ will result in a fair scheduling between all the L users. Equating μ_1 and μ_j for each $j \in I\backslash\{1\}$ will result in $L-1$ equations with L unknowns. The L^{th} equation needed to solve this system of equations for $m_1, m_2, \ldots, m_L$ is

$$\sum_{i=1}^{L} m_i = 1 \quad \text{where} \quad 0 \le m_i \le 1 \tag{5.15}$$

5.3.3 Example Network; $L = 3$ and $N = 3$

We use the developed methodology to find the server sharing policy in a system with three users ($L = 3$) and $N = 3$. In this case, Equation (5.14) will be reduced to

$$\mu_i = C\Bigg[\sum_{r=1}^{2} r\pi_{i,r} \prod_{l\in I\setminus\{i\}} \pi_{l,0} + \sum_{\forall\mathcal{M}^{(1,i)}\subset I}\Bigg[\frac{m_i}{\sum_{j\in\mathcal{M}^{(1,i)}} m_j}\cdot \sum_{r=1}^{2} r\pi_{i,r} \prod_{k\in\mathcal{M}^{(1,i)}\setminus\{i\}}\left(\sum_{r=1}^{2}\pi_{k,r}\right) \cdot \prod_{l\in I\setminus\mathcal{M}^{(1,i)}} \pi_{l,0}\Bigg] + \frac{m_i}{\sum_{j\in I} m_j}\sum_{r=1}^{2} r\pi_{i,r} \prod_{k\in I\setminus\{i\}}\left(\sum_{r=1}^{2}\pi_{k,r}\right)\Bigg] \tag{5.16}$$

For user 1, the average service rate μ_1 is given by

$$\mu_1 = C\Big[(\pi_{1,1}+2\pi_{1,2})\pi_{2,0}\pi_{3,0} + \frac{m_1}{m_1+m_2}(\pi_{1,1}+2\pi_{1,2})(\pi_{2,1}+\pi_{2,2})\pi_{3,0} + \frac{m_1}{m_1+m_3} (\pi_{1,1}+2\pi_{1,2})(\pi_{3,1}+\pi_{3,2})\pi_{2,0} + \frac{m_1}{m_1+m_2+m_3}(\pi_{1,1}+2\pi_{1,2})(\pi_{2,1}+\pi_{2,2})(\pi_{3,1}+\pi_{3,2})\Big] \tag{5.17}$$

since there are two combinations in I that satisfy ($\forall\mathcal{M}^{(1,i)} \subset I$): those are $\mathcal{M}^{(1,i)} \in \{\{1,2\},\{1,3\}\}$ and depicted as the second and third term in (5.17) respectively. The other two users' service rates can be obtained in a similar manner

$$\mu_2 = C\Big[(\pi_{2,1}+2\pi_{2,2})\pi_{1,0}\pi_{3,0} + \frac{m_2}{m_1+m_2}(\pi_{2,1}+2\pi_{2,2})(\pi_{1,1}+\pi_{1,2})\pi_{3,0} + \frac{m_2}{m_2+m_3} (\pi_{2,1}+2\pi_{2,2})(\pi_{3,1}+\pi_{3,2})\pi_{1,0} + \frac{m_2}{m_1+m_2+m_3}(\pi_{2,1}+2\pi_{2,2})(\pi_{1,1}+\pi_{1,2})(\pi_{3,1}+\pi_{3,2})\Big] \tag{5.18}$$

and

$$\mu_3 = C\Big[(\pi_{3,1}+2\pi_{3,2})\pi_{1,0}\pi_{2,0} + \frac{m_3}{m_1+m_3}(\pi_{3,1}+2\pi_{3,2})(\pi_{1,1}+\pi_{1,2})\pi_{2,0} + \frac{m_3}{m_2+m_3}$$
$$(\pi_{3,1}+2\pi_{3,2})(\pi_{2,1}+\pi_{2,2})\pi_{1,0} + \frac{m_3}{m_1+m_2+m_3}(\pi_{3,1}+2\pi_{3,2})(\pi_{1,1}+\pi_{1,2})(\pi_{2,1}+\pi_{2,2})\Big] \tag{5.19}$$

Substituting (5.17), (5.18) and (5.19) in $\mu_1 = \nu_2\mu_2$ and $\mu_1 = \nu_3\mu_3$ will yield two equations with 3 unknowns. The third equation required to solve for m_1, m_2 and m_3 is $m_1 + m_2 + m_3 = 1$.

Remark: The manual solution of the resultant system of equations can be awkward especially for large L. However, many existing mathematical software packages (e.g., MathCad and MATLAB) can provide numerical or even analytical solutions for such systems. □

5.4 Results and Discussion

In this section, we will use the proposed methodology to model and study a 3G wireless system with two users whose channels are modeled by 3-state FSMC. We derived the server shares m_i and the corresponding average service rates μ_i for the two users, and are given by

$$m_1 = \frac{\pi_{1,1}\pi_{2,1}+\pi_{1,1}\pi_{2,2}+2\pi_{1,2}\pi_{2,1}+2\pi_{1,2}\pi_{2,2}+\nu\pi_{2,1}+2\nu\pi_{2,2}-\pi_{1,1}-2\pi_{1,2}}{(2\nu+1)\pi_{1,1}\pi_{2,2}+(\nu+1)\pi_{1,1}\pi_{2,1}+(\nu+2)\pi_{1,2}\pi_{2,1}+(2\nu+2)\pi_{1,2}\pi_{2,2}} \tag{5.20}$$

$$m_2 = \frac{\nu\pi_{1,1}\pi_{2,1}+2\nu\pi_{1,1}\pi_{2,2}+\nu\pi_{1,2}\pi_{2,1}+2\nu\pi_{1,2}\pi_{2,2}-\nu\pi_{2,1}-2\nu\pi_{2,2}+\pi_{1,1}+2\pi_{1,2}}{(2\nu+1)\pi_{1,1}\pi_{2,2}+(\nu+1)\pi_{1,1}\pi_{2,1}+(\nu+2)\pi_{1,2}\pi_{2,1}+(2\nu+2)\pi_{1,2}\pi_{2,2}} \tag{5.21}$$

where $0 \leq m_i \leq 1$, $\forall\, i \in \{1, 2\}$. The average service rates μ_1 and μ_2 are given by:

$$\mu_1 = C\Big[\pi_{1,1} + 2\pi_{1,2} + (m_1 - 1)(\pi_{1,1}\pi_{2,2} + \pi_{1,1}\pi_{2,1} + 2\pi_{1,2}\pi_{2,1} + 2\pi_{1,2}\pi_{2,2})\Big] \quad (5.22)$$

$$\mu_2 = C\Big[\pi_{2,1} + 2\pi_{2,2} + (m_2 - 1)(2\pi_{1,1}\pi_{2,2} + \pi_{1,1}\pi_{2,1} + \pi_{1,2}\pi_{2,1} + 2\pi_{1,2}\pi_{2,2})\Big] \quad (5.23)$$

In the following, we will study some special cases and present some of the numerical results we computed for this system.

5.4.1 Case 1: Symmetrical Channel Parameters

In this case, both users have identical FSMC stationary distribution, i.e., $\vec{\boldsymbol{\pi}}_i = \vec{\boldsymbol{\pi}}$, $\forall\, i \in \{1, 2\}$. Figure 5.3 depicts the required server shares allocation to user1 and user2 (i.e., m_1 and m_2) to achieve service differentiation with rate (ν) for different channel conditions (i.e., different $\vec{\boldsymbol{\pi}}$). In this figure we can see that $\nu = 1.0$ is a focal point at which both users will get 0.5 of the server capacity no matter what are the values of $\vec{\boldsymbol{\pi}}_i$, as long as they are equal. We can also notice that for $\nu \neq 1$, bigger share is assigned to the user with higher QoS requirements in order to achieve the required service differentiation between users. Figure 5.4 illustrates the fact that using the server sharing policy depicted in Figure 5.3 will achieve the desired service differentiation within certain limits that are dictated by the channel conditions.

5.4.2 Case 2: Fair Scheduler ($\nu = 1.0$)

Fair scheduler is the one that provides all users with the same average service rate (when possible) regardless of their channel conditions. Figures 5.5 and 5.6 illustrate the server sharing policy and the resulting average service rates for both users when $\nu = 1.0$. It can be seen that user's share increases when its channel conditions deteriorate. This policy will achieve the most possible fairness level, on the expense of

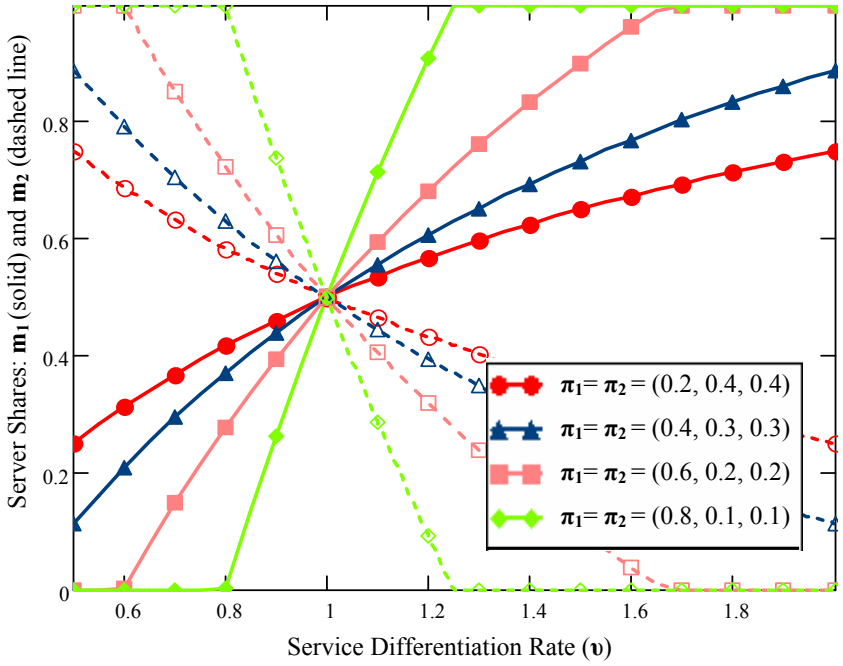


Figure 5.3: Service share m_i vs. ν for two users with symmetrical channel

overall system throughput in some cases (see Figure 5.6). However, fairness between users can only be achieved within certain limits of their channel qualities. These limits depends on the difference $|\vec{\pi}_1 - \vec{\pi}_2|$ (see Figure 5.6).

5.4.3 Case 3: Equal Shares Scheduler ($m_1 = m_2 = 0.5$)

The equal shares scheduler always assigns equal shares of the server capacity to each user, when both are connected, regardless of their expected channel conditions. The users' service rates for such system are shown in Figure 5.7 for different values of $\vec{\pi}_1$ and $\vec{\pi}_2$. For simplicity of presentation, we assume that $\pi_{i,1} = \pi_{i,2}$. It is obvious that this setup is fair only when $\vec{\pi}_1 = \vec{\pi}_2$. The average service rates for the two users diverge linearly when the difference $|\vec{\pi}_1 - \vec{\pi}_2|$ increases. We also include the case when 2-state FSMC model is used (using Equations (5.12) and (5.13)) in this graph for comparison.

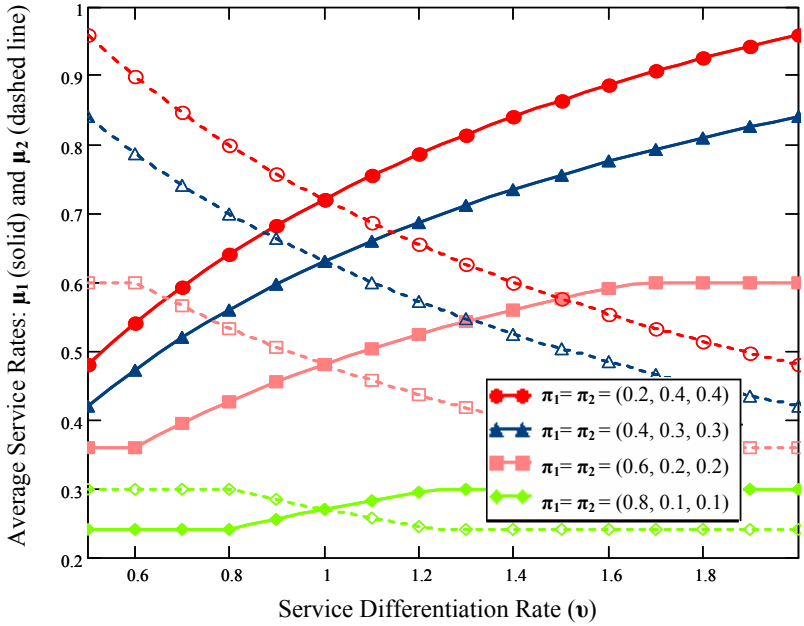


Figure 5.4: Average service rate μ_i vs. ν for two users with symmetrical channel

5.4.4 Case 4: Differentiated Services

Figure 5.8 shows the effect of the service differentiation rate (ν) on the average service rates (μ_i) for two users with different channel conditions. It is obvious that the required service differentiation might not be achievable for certain channel conditions.

The presented technique can be used in conjunction with any dynamic resource allocation policy to aid in providing service differentiation between classes. This can be done by using the dynamic policy to allocate the server instantaneously and using the technique we presented here to monitor the long-run average server shares and adjust the dynamic policy parameters when required. It can also be used to measure the fairness level provided by other existing schedulers. The simplicity and tractability of this technique comes with the price that this server sharing policy does not utilize the dynamic variation of the wireless system to achieve higher system throughput when the system uses rate adaptability. However, it is a powerful tool for understanding server sharing in systems with random channel connectivity such as 3G and 4G wireless systems.

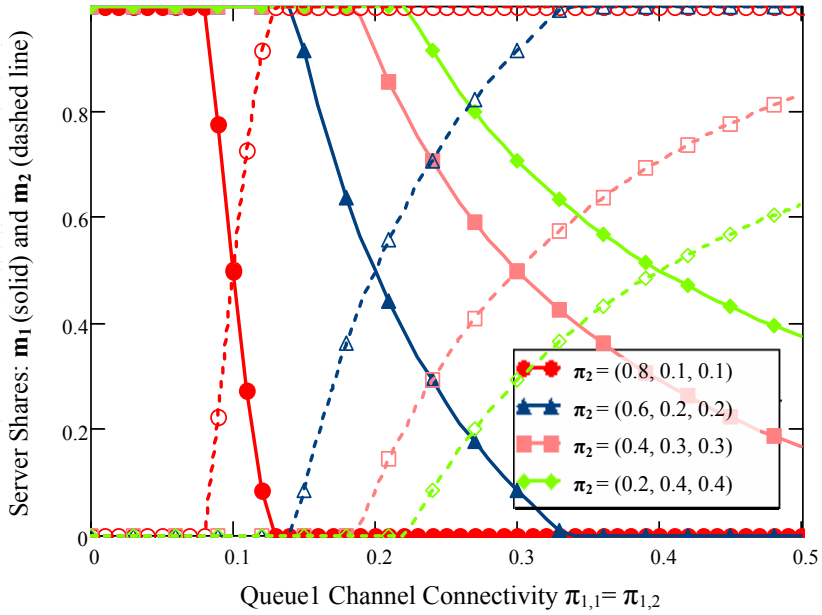


Figure 5.5: Service share (m_i) vs. q_1 when $\nu = 1.0$

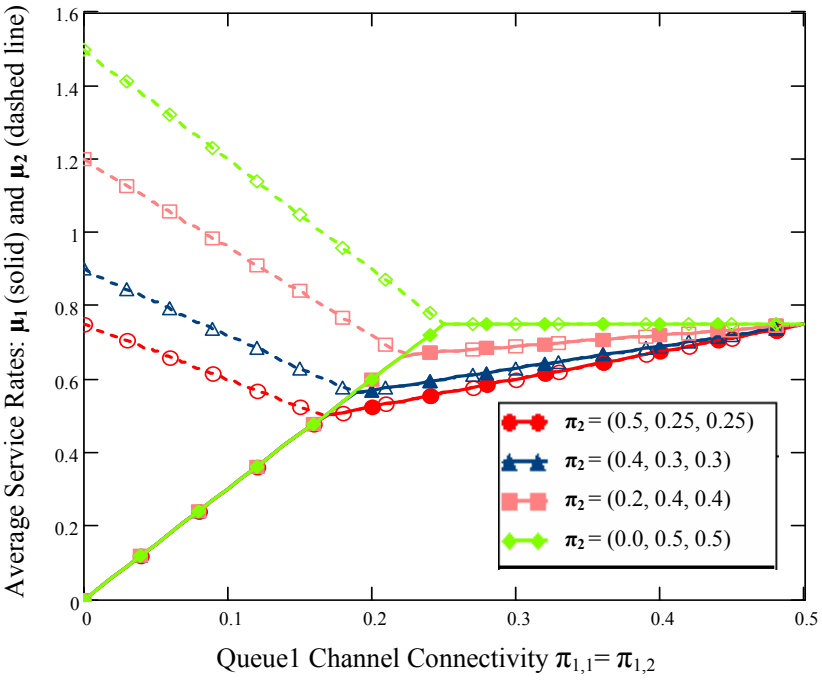


Figure 5.6: Service rate (μ_i) vs. $\vec{\pi}_1$ when $\nu = 1.0$

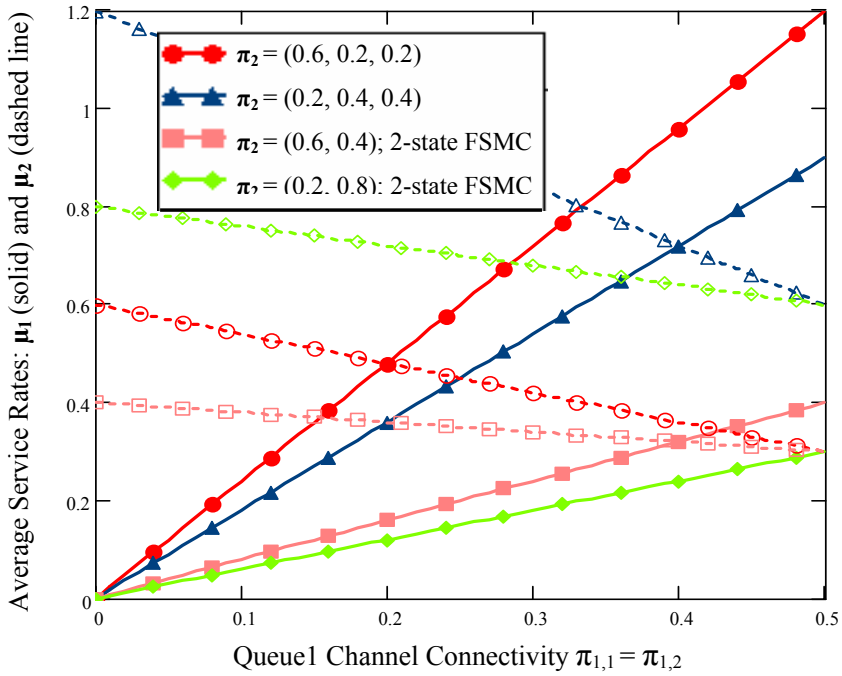


Figure 5.7: Service rate (μ_i) vs. $\vec{\pi}_1$ when $m_1 = m_2 = 0.5$

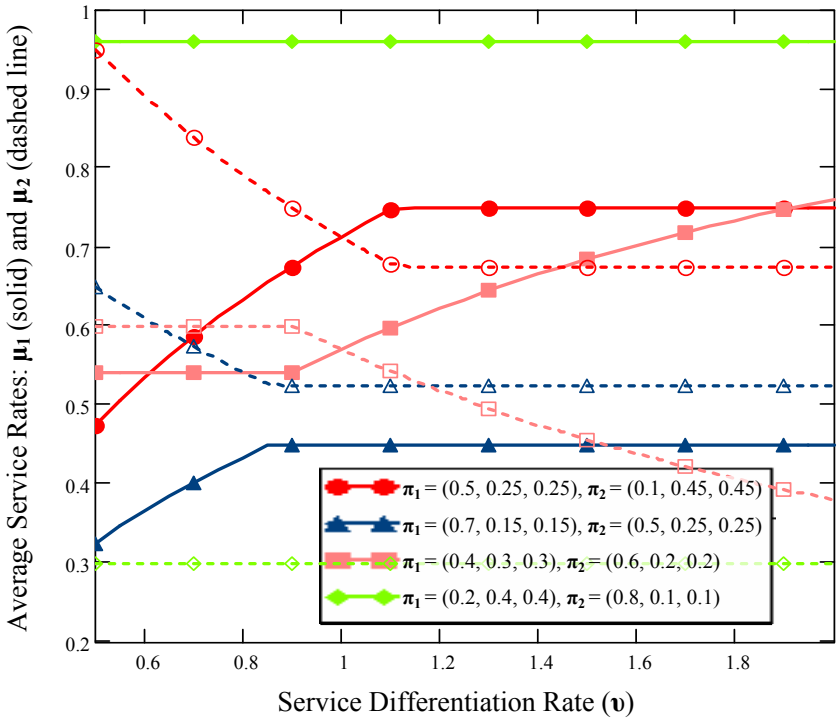


Figure 5.8: Service share (m_i) vs. ν when $\vec{\pi}_1 \neq \vec{\pi}_2$

Chapter 6

Conclusions and Suggestions for Future Work

6.1 Conclusions

6.1.1 Optimal Control in Emerging Wireless Networks

In this part, we presented a model for service sharing in emerging wireless systems. We investigated the class of Most Balancing policies. These policies serve the longest connected queues in the system in an effort to "equalize" the queue occupancies, i.e., minimize the total difference between queue lengths in the system. A theoretical proof of the optimality of MB policies using stochastic dynamic coupling argument was presented. The LCSF/LCQ algorithm was designed as an approximate implementation of MB policies.

A simulation study was conducted to study the performance of five different work conserving policies including the optimal one. The results showed that the Most Balancing policies outperformed all other policies, even when statistical assumptions were relaxed. As expected, the Least Balancing policy performed the worst. We also found that a randomized policy can perform very close to the optimal one in most

situations.

The model was extended by incorporating the effect of retransmitting the unsuccessful transmissions (packets with uncompleted service). Using stochastic coupling argument, we proved that the MB policies are still optimal.

The presented model is an important one for the modeling of emerging wireless systems, since these systems use channel sharing (using a combination of code division multiple access and time division multiple access) to maximize system performance and enhance fairness between users in the system. It also presents a typical scheduling problem that possesses an inherent significance in the stochastic optimization theory as an application for optimal service (or resource) allocation in queuing systems with random server connectivity.

6.1.2 Optimal Scheduling in HSDPA Networks

In this part, we developed an MDP model for the scheduling problem in 3G-HSDPA wireless system. We used dynamic programming and value iteration to determine numerically the optimal scheduling policy in this system. Value iteration is computationally demanding especially for large state space (e.g., larger number of users and/or wireless channel states). To counter this, we developed a heuristic approach to obtain a near-optimal heuristic policy. The suggested approach involves studying the structural and behavioral characteristics of the optimal policy using the MDP model. Then we use this data to determine a near-optimal heuristic scheduling policy that is shown to compare favorably with the optimal one. Therefore, it is a good candidate for practical application and use.

Towards the derivation of the heuristic policy, we approximated the policy switching curves linearly (Equations (4.22)-(4.24)). A non-linear approximation could be used, although it may not necessarily provide substantial improvement.

We also studied (using simulation) the effect of the code allocation granularity on

the optimal policy performance. Our results showed that a policy with finer granularity will perform better in light to moderate loading conditions, while a coarse policy is more desirable in heavy loading conditions. We also showed that the performance gain when using $c < 5$ is marginal and does not justify the added complexity. Our results also proved that RR is undesirable in HSDPA systems due to the poor performance and lack of fairness if deployed in such an environment.

6.1.3 Analytic Evaluation of Downlink Service Rate in 3G Wireless Networks

In this part, a server sharing based stochastic model for 3G wireless networks was presented. A closed-form formula to find the average service rate per user was provided. The channel was modeled by a N-state FSMC (finite state Markov channel). A two-user system with 3-state FSMC channel was modeled by a set of equations. The resultant set of equations was solved to determine the server shares and the corresponding average service rates for some special cases (e.g., Equal shares scheduler and Fair scheduler). It is evident from the results that assigning equal shares of the server capacity to all users in a 3G wireless system with independent random channel connectivity will result in service differentiation that depends on the relative channel quality. We also showed that fairness can only be achieved within a specific range of the channels' parameters. The qualitative results obtained for the 2-state FSMC case seem to hold for the 3-state FSMC case. We can conjecture that these results are extendable to any number of channel states.

6.2 Suggestions for Future Work

For future work, we suggest the following:

1. Investigate the applicability and optimality of the class of MB policies (presented in Chapter 2) in other wireless models. We believe that these policies (or a slightly different flavor of them) are optimal for a large set of scheduling and routing problems in wireless networks.

2. Relax the symmetry assumption we imposed on the arrivals and connectivities in the model presented in Chapter 2. The resultant model in this case will incorporate heterogeneous arrivals and/or heterogeneous connectivities to different queues. An MB policy may not be optimal in this case. However, according to our intuition, the optimal policy should be a weighted version of the MB policies, i.e., a policy that tries to balance a weighted version of the queue lengths in this system. The weight will be a function of the differences in the different queues arrival rates and the differences in their connectivities.

3. Extend the work presented in Chapter 2 to continuous time systems. We believe that the MB policies are still optimal. The coupling argument in this case will be performed in the instances were events (such as arrivals, service completions or connectivity changes) happen.

List of References

[1] 3GPP, "3gpp a global initiative," *Website: http://www.3gpp.org*.

[2] X. Liu, E. K. P. Chong, and N. B. Shroff, "A framework for opportunistic scheduling in wireless networks," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 2003.

[3] X. Liu, E. Chong, and N. Shroff, "Opportunistic transmission scheduling with resource-sharing constraints in wireless networks," *IEEE Journal on Selected Areas in Communications*, 2001.

[4] S. Lu, V. Bharghavan, and R. Srikant, "Fair scheduling in wireless packet networks," *IEEE/ACM Transactions on Networking*, 1999.

[5] S. Shakkottai and A. Stolyar, "Scheduling for multiple flows sharing a time-varying channel: The exponential rule," *Translations of the AMS*, 2001.

[6] T. Nandagopal, S. Lu, and V. Bharghavan, "A unified architecture for the design and evaluation of wireless fair queueing algorithms," in *Proceedings of ACM Mobicom'99*, (Seattle, WA, USA), Aug 1999.

[7] T. Ng, I. Stoica, and H. Zhang, "Packet fair queueing algorithms for wireless networks with location-dependent errors," in *Proceedings of IEEE INFOCOM'98*, (San Francisco, USA), Mar 1998.

[8] S. Shakkottai and R. Srikant, "Scheduling real-time traffic with deadlines over a wireless channel," in *Proceedings of ACM Workshop on Wireless and Mobile Multimedia*, (Seattle, WA, USA), Aug 1999.

[9] S. J. Stidham, "Optimal control of admission to a queueing system," *IEEE Transactions in Automatic Control*, vol. 30, pp. 705–713, 1985.

[10] B. Hajek, "Optimal control of two interacting service stations," *IEEE Trans. Automatic Control*, vol. 29, pp. 491–499, 1984.

[11] Z. Rosberg, P. Varaiya, and J. Walrand, "Optimal control of service in tandem queues," *IEEE Trans. Automatic Control*, vol. 27, pp. 600–610, 1982.

[12] W. Lin and P. Kumar, "Optimal control of a queuing system with two heterogeneous servers," *IEEE Trans. Automatic Control*, vol. 29, pp. 696–703, 1984.

[13] M. Puterman, *Markov Decision Process: Discrete Stochastic Dynamic Programming*. NY, USA: John wiley & Sons Inc., 1994.

[14] S. M. Ross, *Introduction to Stochastic Dynamic Programming*. USA: Academic Press, 1983.

[15] P. R. Kumar and P. P. Varaiya, *Stochastic Systems: Estimation, Identification and Adaptive Control*. Englewood Cliffs, NJ: Prentice Hall, 1986.

[16] A. Schrijver, *Theory of Linear and Integer Programming*. Essex, GB: John wiley & Sons Inc., 1986.

[17] I. Viniotis, *Optimal Control of Integrated Communication Systems Via Linear Programming Techniques*. PhD thesis, University of Maryland, 1988.

[18] P. Luh and I. Viniotis, "Optimality of threshold policies for heterogeneous server systems," in *The 29th Conference on Decision and Control, Honolulu, HI*, (Honolulu, HI, USA), pp. 870–875, 1990.

[19] I. Christidou, I. Lambadaris, and R. Mazumdar, "Optimal control of arrivals to a feedback queueing system," in *Proceedings of 27th IEEE Conference on Decision and Control*, (AUSTIN, TX, USA), pp. 663–667, Dec 1988.

[20] D. Stoyan, *Comparison Methods for Queues and other Stochastic Models*. Chichester: John Wiley and Sons, 1983.

[21] T. Lindvall, *Lectures on the coupling method*. New York: Dover Publications, 2002.

[22] J. Walrand, "A note on optimal control of a queuing system with two heterogeneous servers," *Systems and Control Letters*, 1984.

[23] L. Tassiulas and A. Ephremides, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Transactions on Information Theory*, vol. 39, pp. 466–478, mar 1993.

[24] A. Ganti, E. Modiano, and J. N. Tsitsiklis, "Optimal transmission scheduling in symmetric communication models with intermittent connectivity," *IEEE Transactions on Information Theory*, vol. 53, pp. 998–1008, 2007.

[25] N. Bambos and G. Michailidis, "On the stationary dynamics of parallel queues with random server connectivities," in *Proceedings of 34th Conference on Decision and Control (CDC)*, (New Orleans, LA), 1995.

[26] N. Bambos and G. Michailidis, "On parallel queueing with random server connectivity and routing constraints," *Probability in Engineering and Informational Sciences*, vol. 16, pp. 185–203, 2002.

[27] S. Kittipiyakul and T. Javidi, "Delay-optimal server allocation in multiqueue multi-server systems with time-varying connectivities," *IEEE Transactions on Information Theory*, vol. 55, may 2009.

[28] C. Lott and D. Teneketzis, "On the optimality of an index rule in multichannel allocation for single-hop mobile networks with multiple service classes," *Probability in the Engineering and Information Services*, vol. 14, pp. 259–297, 2000.

[29] G. Koole, Z. Liu, and R. Righter, "Optimal transmission policies for noisy channels," *Operations Research*, vol. 49, pp. 892–899, 2001.

[30] X. Liu, E. Chong, and N. Shroff, "Optimal opportunistic scheduling in wireless networks," in *Proceedings of IEEE 58th Vehicular Technology Conference*, (Orlando, FL, USA), pp. 663–667, Oct 2003.

[31] X. Liu, E. Chong, and N.B.Shroff, "A framework for opportunistic scheduling in wireless networks," *Computer Networks*, vol. 41, p. 451474, 2003.

[32] M. Andrews, "Instability of the proportional fair scheduling algorithm for hdr," *IEEE Transactions on Wireless Communications*, vol. 3, pp. 1422–1426, 2004.

[33] R. Agrawal and V. Subramanian, "Optimality of certain channel aware scheduling policies," in *Proceedings of the 40th Annual Allerton Conference on Communication, Control, and Computing*, (Monticello, Illinois, USA), Oct 2002.

[34] M. Andrews and L. Zhang, "Scheduling over a time-varying user-dependent channel with applications to high speed wireless data," *Journal of the ACM (JACM)*, vol. 52, pp. 809 – 834, 2005.

[35] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length based scheduling and congestion control," in *Proceedings of IEEE INFOCOM 05*, (Miami, FL, USA), Mar 2005.

[36] A. Stolyar, "On the asymptotic optimality of the gradient scheduling algorithm for multiuser throughput allocation," *Operations Research*, vol. 53, p. 1225, 2005.

[37] S. Lu, V. Bharghavan, and R. Srikant, "Fair scheduling in wireless packet networks," *IEEE/ACM Transactions on Networking*, vol. 7, pp. 473–489, 1999.

[38] K. Wasserman, G. Michailidis, and N. Bambos, "Optimal processor allocation to differentiated job flows," *Performance Evaluation, Elsevier*, vol. 63, p. 114, 2006.

[39] P. Sparragis, D. Towsley, and C. Cassandras, "Extremal properties of the shortest/longest non-full queue policies in finite-capacity systems with state-dependent service rates," *Journal of Applied Probability*, vol. 30, pp. 223–236, 1993.

[40] S. Shakkottai and A. Stolyar, "Scheduling for multiple flows sharing a time-varying channel: The exponential rule," *American Mathematical Society Translations*, vol. 207, pp. 185–202, 2002.

[41] S. M. Ross, *Stochastic Processes, 2nd ed.* New York: Wiley, 1996.

[42] R. Lidl and G. Pilz, *Applied abstract algebra, 2nd edition.* New York: Springer, 1998.

[43] 3GPP, "High speed downlink packet access (hsdpa): Overall description (release 5)," *3GPP Technical specification*, vol. 5.7.0, dec 2004.

[44] H. Wei and R. Izmailov, "Channel-aware soft bandwidth guarantee scheduling for wireless packet access," in *IEEE Wireless Communications and Networking Conference (WCNC 04)*, (Atlanta, USA), pp. 1276–1281, mar 2004.

[45] H. R. Shao, D. G. C. Shen, J. Z, and P. Orlik, "Dynamic resource control for high-speed downlink packet access wireless channel," in *23rd International Conference on Distributed Computing Systems*, (Providence, Rhode Island, USA), pp. 838–843, May 2003.

[46] H. Holma and A. Toskala, *WCDMA for UMTS, Radio Access for Third Generation Mobile Communication, 3rd ed.* NY, USA: John Wiley & Sons Inc., 2004.

[47] J. P. Castro, *All IP in 3G CDMA Networks*. NY, USA: John Wiley & Sons Inc., 2004.

[48] T. Bonald, "A score-based opportunistic scheduler for fading radio channels," in *Proceeding of the 5th European Wireless Conference*, (Barcelona, Spain), IEEE, Mar 2004.

[49] W. S. Jeon, D. G. Jeong, and B. Kim, "Packet scheduler for mobile internet services using high speed sownlink packet access," *IEEE Transactions on Wireless Communications*, vol. 3, sep 2004.

[50] H. Jiang, W. Zhuang, and X. S. Shen, "Cross-layer design for resource allocation in 3g wireless networks and beyond," *IEEE Communications Magazine*, 2005.

[51] H. Ekstrom, A. Furuskar, J. Karlsson, M. Meyer, S. Parkvall, J. Torsner, and M. Wahlqvist, "Technical solutions for the 3g long-term. evolution," *IEEE Communications Magazine*, pp. 38–45, mar 2006.

[52] J. Peisa, S. Wager, M. Sagfors, J. Torsner, B. Goransson, T. Fulghum, C. Cozzo, and S. Grant, "High speed packet access evolution - concept and technologies," in *Proceedings of 65th IEEE Vehicular Technology Conference*, (Dublin, Ireland), apr 2007.

[53] P. Kela, J. Puttonen, N. Kolehmainen, T. Ristaniemi, T. Henttonen, and M. Moisio, "Dynamic packet scheduling performance in utra long term evolution downlink," in *Proceedings of the International Symposium on Wireless Pervasive Computing (ISWPC08)*, (Santorini, Greece), may 2008.

[54] G. Monghal, K. Pedersen, I. Kovacs, and P. Mogensen, "Qos oriented time and frequency domain packet schedulers for the utran long term evolution," in *Proceedings of 67th IEEE Vehicular Technology Conference*, (Singapore), pp. 2532–2536, may 2008.

[55] 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 Communication Magazine*, vol. 38, pp. 70–77, jul 2000.

[56] S. Lin and D. Costello, *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[57] P. Frenger, S. Parkvall, and E. Dahlman, "Performance comparison of harq with chase combining and incremental redundancy for hsdpa," in *Proceedings of*

54th IEEE Vehicular Technology Conference, vol. 3, (Atlantic City, NJ, USA), pp. 1829–1833, oct 2001.

[58] H. S. Wang and N. Moayeri, "Finite-state markov channel–a useful model for radio communication channels," *IEEE Transactions on Vehicular Technology*, vol. 44, pp. 163–171, feb 1995.

[59] Q. Zhang and S. Kassam, "Finite-state markov model for rayleigh fading channels," *IEEE Transactions on Communications*, 1999.

[60] M. Hassan, M. Krunz, and I. Matta, "Markov-based channel characterization for tractable performance analysis in wireless packet networks," *IEEE Transactions on Wireless Communications*, 2004.

[61] T. E. Kolding, F. Frederiksen, and P. E. Mogensen, "Performance aspects of wcdma systems with hsdpa," *IEEE 56th IEEE Vehicular Technology Conference (VTC)*, vol. 1, pp. 477–481, sep 2002.

[62] R. Bellman, *Dynamic Programming*. Princeton, USA: Princeton University Press, 1957.

[63] L. I. Sennott, *Stochastic Dynamic Programming and The Control of Queueing Systems*. New York, NY, USA: Wiley Series in Probability and Stochastics, 1999.

Appendix A

The Effect of A Balancing Interchange on The Imbalance Index

In this appendix we present a lemma that quantifies the effect of performing a balancing interchange on the imbalance index $\kappa_n(\pi)$. Recall that the "balancing interchange" is defined in Section 2.6.1.

Lemma 11. *Let* $\mathbf{x}$ *be an* L*-dimensional vector; suppose that* $\mathbf{x}^*$ *is obtained from* $\mathbf{x}$ *by performing a balancing interchange. Then*

$$\sum_{i'=1}^{L} \sum_{j'=i'+1}^{L} (x^*_{[i']} - x^*_{[j']}) \leq \sum_{i=1}^{L} \sum_{j=i+1}^{L} (x_{[i]} - x_{[j]}) \tag{A-1}$$

where $x^*_{[k]}$ *(respectively* $x_{[k]}$*) is the* k^{th} *largest component of vector* $\mathbf{x}^*$ *(respectively* $\mathbf{x}$*).*

Proof. We generate the vector $\mathbf{x}^*$ by performing a *balancing interchange* of two components (the l^{th} and the s^{th} largest components) in the vector $\mathbf{x}$. The resulted vector $\mathbf{x}^*$ is characterized by the following:

$$\begin{aligned} &x^*_{[l']} = x_{[l]} - 1, \quad x^*_{[s']} = x_{[s]} + 1, \quad x_{[l]} > x_{[s]} \\ &x^*_k = x_k, \quad \forall k \neq [l], [s] \end{aligned} \tag{A-2}$$

where $[l']$ (respectively $[s']$) is the new order (i.e., the order in the new vector $\mathbf{x}^*$) of the l^{th} (respectively the s^{th}) component in the original vector $\mathbf{x}$.

From Equation (A-2) we can easily show that

$$\sum_{i'=1}^{L}\sum_{j'=i'+1}^{L}(x^*_{[i']}-x^*_{[j']})=\sum_{i=1}^{L}\sum_{j=i+1}^{L}(x_{[i]}-x_{[j]}),\ \forall i,j\notin\{l,s\}\text{ and }i',j'\notin\{l',s'\} \quad \text{(A-3)}$$

For the remaining cases, i.e., when at least one of the indices i, j belongs to $\{l, s\}$ or i', j' belongs to $\{l', s'\}$, we pair the index i' (respectively j') on the left hand side with the index i (respectively j) on the right hand side of Equation (A-1). We first assume that $x_{[l]} > x_{[s]} + 1$, then we can easily show that $l' \leq s'$. In this case, we have the following five, mutually exclusive, cases to consider:

1. When $i' = l', i = l, j' = s'$ and $j = s$. This case occurs only once, i.e., when decomposing the double sum in Equation (A-1) we can find only one term that satisfies this case. From Equation (A-2) we have

$$x^*_{[l']} - x^*_{[s']} = x_{[l]} - x_{[s]} - 2 \quad \text{(A-4)}$$

2. When $i' = l', i = l, j' \neq s'$ and $j \neq s$. There are $L - l - 1$ terms that satisfy this case. Analogous to case (1) we can determine that

$$x^*_{[l']} - x^*_{[j']} = x_{[l]} - x_{[j]} - 1 \quad \text{(A-5)}$$

3. When $i' \neq l', i \neq l, j' = s'$ and $j = s$. There are $s - 2$ terms that satisfy this case. In this case we can show that

$$x^*_{[i']} - x^*_{[s']} = x_{[i]} - x_{[s]} - 1 \quad \text{(A-6)}$$

4. When $i' \neq l', s', i \neq l, s, j' = l'$ and $j = l$. There are $l - 1$ terms that satisfy this case. In this case we can show that

$$x^*_{[i']} - x^*_{[l']} = x_{[i]} - x_{[l]} + 1 \tag{A-7}$$

5. When $i' = s', i = s, j' \neq l', s'$ and $j \neq l, s$. There are $L - s$ terms that satisfy this case. In this case we have

$$x^*_{[s']} - x^*_{[j']} = x_{[s]} - x_{[j]} + 1 \tag{A-8}$$

The above cases (i.e., Equations (A-3)-(A-8)) cover all the terms in Equation (A-1) when $x_{[l]} > x_{[s]} + 1$. Combining all these terms yields:

$$\begin{aligned}\sum_{i'=1}^{L}\sum_{j'=i'+1}^{L}(x^*_{[i']} - x^*_{[j']}) &= \sum_{i=1}^{L}\sum_{j=i+1}^{L}(x_{[i]} - x_{[j]}) - 2\cdot(1) - 1\cdot(L-l-1) - 1\cdot(s-2) \\ &\qquad +1\cdot(l-1) + 1\cdot(L-s) \\ &= \sum_{i=1}^{L}\sum_{j=i+1}^{L}(x_{[i]} - x_{[j]}) - 2\cdot(s-l)\end{aligned} \tag{A-9}$$

Since $s > l$ by assumption (the vector $\mathbf{x}$ is assumed to be ordered in a descending manner), then we conclude that Equation (A-1) is satisfied with strict inequality.

Furthermore, if $x_{[l]} = x_{[s]} + 1$, then from Equation (A-2) it is clear that $x^*_{[l']} = x_{[s]}$ and $x^*_{[s']} = x_{[l]}$. Therefore, Equation (A-1) is satisfied with strict equality. □

Appendix B

Proofs of The Results of Section 2.5.1

B.1 Proof of Proposition 1:

Proof. Let X' have distribution F. Define $Y' = G^{-1}(F(X'))$, then

$$\begin{aligned} P[G^{-1}(F(X')) \leq z] &= P[F(X') \leq G(z)] \\ &= P[X' \leq F^{-1}(G(z))] \\ &= F[F^{-1}(G(z))] = G(z) \end{aligned}$$

Hence, Y' has distribution G. Since $F \geq G$, then $F^{-1} \leq G^{-1}$ and

$$X' = F^{-1}(F(X')) \leq G^{-1}(F(X')) = Y'$$

and the result follows. □

B.2 Proof of Theorem 1:

Proof. Let $\mathbf{X}$ and $\mathbf{Y}$ have probability distributions F and G respectively and $\mathbf{X} \leq_{st} \mathbf{Y}$. Using Proposition 1, we construct the independent random variables

$X^*(t_1), \ldots X^*(t_n)$, $\forall n, \{t_1, \ldots, t_n\}$, with distribution F, such that $X^*(t_i) \leq Y(t_i)$ for all $1 \leq i \leq n$. Then the process $\mathbf{X}^* = \{X^*(t)\}_{t=1}^{\infty}$ generated by $X^*(t_i)$ has the same probability distribution as $\mathbf{X}$.

Since $X^*(t_i) \leq Y(t_i)$ by construction, then for any monotonically increasing function $f()$ we have:

$$f(X^*(t_1), \ldots, X^*(t_n)) \leq f(Y(t_1), \ldots, Y(t_n))$$

for all $n, \{t_1, \ldots, t_n\}$. For any z we have

$$f(X^*(t_1), \ldots, X^*(t_n)) > z \Rightarrow f(Y(t_1), \ldots, Y(t_n)) > z$$

Hence,

$$P[f(X^*(t_1), \ldots, X^*(t_n)) > z] \leq P[f(Y(t_1), \ldots, Y(t_n)) > z]$$

Since, $\mathbf{X}^*$ has the same distribution as the process $\mathbf{X}$, then

$$P[f(X(t_1), \ldots, X(t_n)) > z] \leq P[f(Y(t_1), \ldots, Y(t_n)) > z] \tag{B.1}$$

Conversely starting from (B.1) above and by integrating both sides, for all monotonically increasing functions $f : \mathcal{R}^n \to \mathcal{R}$ and for any $\mathbf{x} \in \mathcal{R}^n$, we have

$$E(f(\mathbf{X^{(n)}})) = \int_0^{\infty} P[f(\mathbf{x}) > z]\, dz \leq \int_0^{\infty} P[f(\mathbf{y}) > z]\, dz = E(f(\mathbf{Y^{(n)}})) \tag{B.2}$$

where $\mathbf{X^{(n)}}, \mathbf{Y^{(n)}}$ are the finite-dimensional projection of the stochastic processes $\mathbf{X}$ and $\mathbf{Y}$ respectively.

For any $\mathbf{a} \in \mathcal{R}^n$, let f_a denote the increasing function

$$f_{\mathbf{a}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} > \mathbf{a} \\ 0 & \text{if } \mathbf{x} \leq \mathbf{a} \end{cases}$$

therefore,

$$E(f_{\mathbf{a}}(\mathbf{X^{(n)}})) = P[\mathbf{X^{(n)}} > \mathbf{a}] \quad \text{and} \quad E(f_{\mathbf{a}}(\mathbf{Y^{(n)}})) = P[\mathbf{Y^{(n)}} > \mathbf{a}]$$

it follows that $\mathbf{X^{(n)}} \leq_{st} \mathbf{Y^{(n)}}$.

Now, letting n approach the size of the entire time series of the processes $\mathbf{X}$ and $\mathbf{Y}$, we have $\mathbf{X} \leq_{st} \mathbf{Y}$ and the proof is complete. □

Appendix C

Proof for Lemma 7 of Section 2.6.4

We apply the coupling method to our proof of Lemma 7 as follows: Let ω and π be a given sample path and server allocation policy. The values of the sequences $\{X(n)\}$ and $\{Y(n)\}$ can be completely determined by ω and π. We denote the ensemble of all random variables as system S. A new sample path, $\tilde{\omega}$ and a new policy $\tilde{\pi}$ are constructed as we specify in detail in the proof. We employ the tilde notation in all random variables that belong to the new system; we denote the ensemble of all random variables (in the new construction) as system $\tilde{S}$. Then, in the coupling definition, $\hat{\omega} = (\omega, \tilde{\omega})$ and the "coupled" processes of interest in Equation (2.14) will be the queue sizes $\hat{\mathbf{X}} = \{X(n)\}$ and $\hat{\mathbf{X}}' = \{\tilde{X}(n)\}$.

We define ω as the sequence of sample values of the random variables $(\mathbf{X}(1), \mathbf{G}(1), \mathbf{Z}(1), \mathbf{G}(2), \mathbf{Z}(2), \ldots)$, i.e., $\omega \equiv (\mathbf{x}(1), \mathbf{g}(1), \mathbf{z}(1), \mathbf{g}(2), \mathbf{z}(2), \ldots)$. The sample path $\tilde{\omega} \equiv (\tilde{\mathbf{x}}(1), \tilde{\mathbf{g}}(1), \tilde{\mathbf{z}}(1), \tilde{\mathbf{g}}(2), \tilde{\mathbf{z}}(2), \ldots)$ is constructed such that (a) $\tilde{\mathbf{x}}(1) = \mathbf{x}(1)$, (b) $\tilde{\mathbf{g}}(n)$ the same as $\mathbf{g}(n)$ except for two elements that are exchanged, (c) $\tilde{\mathbf{z}}(n)$ the same as $\mathbf{z}(n)$ except for two elements that are exchanged. Which elements are exchanged is detailed in the proof. In the symmetrical system we are studying, $\{\tilde{\mathbf{G}}(n), \tilde{\mathbf{Z}}(n)\}$ has the same distribution as $\{\mathbf{G}(n), \mathbf{Z}(n)\}$, since the distributions of $\mathbf{G}(n)$ and $\mathbf{Z}(n)$ will not change when reordering their elements. The mappings from $\mathbf{G}(n)$ to $\tilde{\mathbf{G}}(n)$ and from $\mathbf{Z}(n)$ to $\tilde{\mathbf{Z}}(n)$ are one-to-one.

The new policy $\tilde{\pi}$ is constructed (by showing how $\tilde{\pi}$ chooses the withdrawal vector $\tilde{\mathbf{y}}(\cdot)$) as detailed in the proof. Then using Equation (2.7), the new states $\mathbf{x}(\cdot), \tilde{\mathbf{x}}(\cdot)$ are determined under π and $\tilde{\pi}$. The goal is to prove that the relation

$$\tilde{\mathbf{x}}(t) \prec_p \mathbf{x}(t) \tag{C.1}$$

is satisfied at all times t. Towards this end, the preferred order (introduced in Section 2.6.1) can be described by the following property:

Property D: $\tilde{\mathbf{x}}$ is preferred over $\mathbf{x}$ ($\tilde{\mathbf{x}} \prec_p \mathbf{x}$) if and only if one of the following statements holds:

(R1) $\tilde{\mathbf{x}} \leq \mathbf{x}$: the two vectors are component-wise ordered;

(R2) $\tilde{\mathbf{x}}$ is a two-component permutation of $\mathbf{x}$ as described in (2) in Section 2.6.1.

(R3) $\tilde{\mathbf{x}}$ is obtained from $\mathbf{x}$ by performing a *"balancing interchange"* as described in (3) in Section 2.6.1.

The proof for Lemma 7 of Section 2.6.4 is given next.

Proof for Lemma 7. Fix an arbitrary policy $\pi \in \Pi^h_\tau$ and a sample path $\omega = (\mathbf{x}(1), \mathbf{g}(1), \mathbf{z}(1), \ldots)$, where $\mathbf{x}(.), \mathbf{g}(.)$ and $\mathbf{z}(.)$ are sample values of the random variables $\mathbf{X}(.), \mathbf{G}(.)$ and $\mathbf{Z}(.)$. Let , $\pi^* \in \Pi^{MB}$ be an MB policy that works on the same system. The policy π^* chooses a withdrawal vector $y^*(t), \forall t$.

The proof has two parts; Part 1 provides constructions for $\tilde{\omega}$ and $\tilde{\pi}$ (as defined by Lemma 7 statement) for times up to $t = \tau$. Part 2 does the same for $t > \tau$.

Part 1: For the construction of $\tilde{\omega}$, we let the arrivals and channel states be the same in both systems at all time slots before τ, i.e., $\tilde{\mathbf{z}}(t) = \mathbf{z}(t)$ and $\tilde{\mathbf{g}}(t) = \mathbf{g}(t)$ for all $t < \tau$. We construct $\tilde{\pi}$ such that it chooses the same withdrawal vector as π, i.e., we set $\tilde{\mathbf{y}}(t) = \mathbf{y}(t)$ for all $t < \tau$. In this case, at $t = \tau$, the resulting queue sizes are

equal, i.e., $\tilde{\mathbf{x}}(\tau) = \mathbf{x}(\tau)$. In the remainder of Part 1, we will construct the policy $\tilde{\pi}$ at time slot τ such that: (a) $\tilde{\pi} \in \Pi_{\tau}^{h-1}$, i.e., $\tilde{\pi}$ is closer to $\pi^* \in \Pi^{MB}$ than π, and (b) the resulting queue length under $\tilde{\pi}$ is preferred over that under the original policy π, i.e., $\tilde{\mathbf{x}}(\tau+1) \prec_p \mathbf{x}(\tau+1)$. Condition (b) is necessary for proving the second part of the lemma, i.e., the domination of policy $\tilde{\pi}$ over π, a result that will be shown in Part 2 of this proof.

At time slot τ, let $\tilde{\omega}$ have the same channel connectivities and arrivals as ω, i.e., let $\tilde{\mathbf{g}}(\tau) = \mathbf{g}(\tau)$ and $\tilde{\mathbf{z}}(\tau) = \mathbf{z}(\tau)$. Furthermore, let $\mathbf{D} = y^*(\tau) - y(\tau)$. Recall that $h = \sum_{i=0}^{L} |D_i|/2$. Then one of the following two cases may apply:

1- During time slot $t = \tau$, the original policy π differs from π^*, the MB policy, by *strictly* less than h balancing interchanges. Then $\pi \in \Pi_{\tau}^{h-1}$ as well, so we set $\tilde{\mathbf{y}}(\tau) = \mathbf{y}(\tau)$. In this case, the resulting queue sizes $\tilde{\mathbf{x}}(\tau+1), \mathbf{x}(\tau+1)$ will be equal, property (R1) holds true and (C.1) is satisfied at $t = \tau + 1$.

2- During time slot $t = \tau$, π differs from the MB policy π^* by *exactly* h balancing interchanges. Since $\pi \in \Pi_{\tau}^{h}$ and $h > 0$ and following Lemma 6, we can identify two queues l and s such that: (a) $D_l \geq 1$, (b) $D_s \leq -1$, and (c) $\mathbf{I}(l, s)$ is feasible.

The construction of $\tilde{\pi}$ is completed in this case by performing the interchange $\mathbf{I}(l, s)$, i.e.,

$$\tilde{\mathbf{y}}(\tau) = \mathbf{y}(\tau) + \mathbf{I}(l, s), \tag{C.2}$$

or equivalently,

$$\tilde{\hat{\mathbf{x}}}(\tau) = \hat{\mathbf{x}}(\tau) - \mathbf{I}(l, s) \tag{C.3}$$

According to Lemma 4, this interchange is balancing. To complete the construction of $\tilde{\omega}$, we examine the arrivals under ω during time slot τ. We set $\tilde{z}_i(\tau) = z_i(\tau), \forall i \neq l, s$. For queues l and s, we do the following: (i) if $x_l(\tau) = x_s(\tau)+1$ and $z_s(\tau) > z_l(\tau)$ then we swap the arrivals for queues l and s, i.e., we let $\tilde{z}_l(\tau) = z_s(\tau)$ and $\tilde{z}_s(\tau) = z_l(\tau)$, (ii) otherwise, let $\tilde{z}_l(\tau) = z_l(\tau)$ and $\tilde{z}_s(\tau) = z_s(\tau)$. The queue

lengths at the beginning of time slot $(\tau+1)$ under the two policies satisfy property (R1) in case (i) and (R3) otherwise. In either case, (C.1) is satisfied at $t=\tau+1$.

Starting from a preferred state at $t=\tau+1$, we will show next, in Part 2 of the proof, that a feasible control at time slot $t>\tau$, such that the constructed policy $\tilde{\pi}$ dominates the original one π, will always exist. We do that by showing one such construction.

Part 2: In this part of the proof, we construct $\tilde{\omega},\tilde{\pi}$ for times $t>\tau$, such that the preferred order $\tilde{\mathbf{x}}(t)\prec_p\mathbf{x}(t)$ is valid for all $t>\tau$. This will insure $\tilde{\pi}$ domination over π. We will use induction to complete our proof. We assume that $\tilde{\pi}$ and $\tilde{\omega}$ are defined up to time $n-1$ and that $\tilde{\mathbf{x}}(n)\prec_p\mathbf{x}(n)$. We will prove that at time slot n, $\tilde{\pi}$ can be constructed so that $\tilde{\mathbf{x}}(n+1)\prec_p\mathbf{x}(n+1)$. Thus, we have to show that either R1, R2 or R3 holds at time slot $n+1$.

The following three cases, corresponding to properties (R1), (R2) and (R3) are considered next.

Case (1) $\tilde{\mathbf{x}}(n)\leq\mathbf{x}(n)$. The construction of $\tilde{\omega}$ is straightforward in this case. We set $\tilde{\mathbf{z}}(n)=\mathbf{z}(n)$ and $\tilde{\mathbf{g}}(n)=\mathbf{g}(n)$. We construct $\tilde{\pi}$ such that $\tilde{\mathbf{y}}(n)=\mathbf{y}(n)$. In this case, its obvious that $\tilde{\mathbf{x}}(n+1)\leq\mathbf{x}(n+1)$ and (C.1) holds at $t=n+1$.

Case (2) $\tilde{\mathbf{x}}(n)$ is a permutation of $\mathbf{x}(n)$, such that $\tilde{\mathbf{x}}(n)$ can be obtained from $\mathbf{x}(n)$ by permuting components i and j (as described in property R2 of the preferred order). For the construction of $\tilde{\omega}$, we set $\tilde{g}_{i,c}(n)=g_{j,c}(n)$ and $\tilde{g}_{j,c}(n)=g_{i,c}(n)$, for all $c=1,2,\ldots,K$; $\tilde{z}_i(n)=z_j(n)$ and $\tilde{z}_j(n)=z_i(n)$ (refer to footnote 7 or [24]); the connectivities and arrivals for each one of the remaining queues are the same as in ω. We construct $\tilde{\pi}$ such that $\tilde{y}_i(n)=y_j(n)$, $\tilde{y}_j(n)=y_i(n)$ and $\tilde{y}_m(n)=y_m(n)$ for all $m\neq i,j$. As a result, $\tilde{\mathbf{x}}(n+1)$ and $\mathbf{x}(n+1)$ satisfy property (R2) and (C.1) is satisfied at $t=n+1$.

Case (3) $\tilde{\mathbf{x}}(n)$ is obtained from $\mathbf{x}(n)$ by performing a balancing interchange for queues i and j as defined in property (R3). In this case $x_i(n)\geq x_j(n)+1$, by the

definition in (R3)[1]. There are three cases to consider:

(3.a) $x_i(n) = x_j(n) + 1$. Therefore, $\tilde{x}_i(n) = x_j(n)$ and $\tilde{x}_j(n) = x_i(n)$, i.e., the vectors $\mathbf{x}(n)$ and $\tilde{\mathbf{x}}(n)$ have components i and j permuted and all other components are the same. This case corresponds to case (2) above.

(3.b) $x_i(n) > x_j(n) + 1$ and $y_i(n) \leq y_j(n)$. We construct $\tilde{\omega}$ as in case (1) above, and we let $\tilde{y}_m(n) = y_m(n), \forall m \neq j$. Note that it is not feasible for policy π to empty queue i in this case. Depending on whether π empties queue j or not at $t = n$, the construction of $\tilde{\pi}$ will follow one of the following two cases:

(i) $y_j(n) < x_j(n)$, i.e., *π does not empty queue j at $t = n$*, then let $\tilde{y}_j(n) = y_j(n)$ (i.e., $\tilde{\pi}$ is identical to π at $t = n$). In this case, property (R3) will be preserved regardless of the arrivals pattern[2], hence (C.1) is satisfied at $t = n + 1$.

(ii) $y_j(n) = x_j(n)$, i.e., *π empties queue j at $t = n$*. Then if under policy π all the servers connected to queue j are allocated, then let $\tilde{y}_j(n) = y_j(n)$. As in case (i) above, property (R3) holds and (C.1) satisfied at $t = n + 1$.

In the event that π empties queue j without exhausting all the servers connected to queue j, then $\tilde{\pi}$ will be constructed such that one of these idling servers is allocated to queue j, i.e., $\tilde{y}_j(n) = y_j(n) + 1$, so that $\tilde{\pi}$ preserves the work conservation property at $t = n$. Since $\tilde{x}_j(n) = x_j(n) + 1$ by property (R3) and $\tilde{z}_j(n) = z_j(n)$ by construction, then we have

$$\tilde{x}_j(n + 1) = x_j(n + 1) = z_j(n)$$

Since $\tilde{x}_i(n) = x_i(n) - 1$ by property (R3), $\tilde{z}_i(n) = z_i(n)$ and $\tilde{y}_i(n) = y_i(n)$ by construction, we have

$$\tilde{x}_i(n + 1) = x_i(n + 1) - 1$$

The rest of the queues will have the same lengths in both systems at $t = n + 1$.

[1]By definition, we have $x_i(n) > x_j(n)$, $\tilde{x}_i(n) = x_i(n) - 1$ and $\tilde{x}_j(n) = x_j(n) + 1$.

[2]Note that if $x_i = x_j + 1$ then property (R2) is a special case of (R3).

Therefore, (R1) holds with strict inequality and (C.1) is satisfied at $t = n + 1$. This case shows that a "more" balancing policy results in a strict enhancement of the original policy.

Cases (i) and (ii) are the only possible ones, since π cannot allocate more servers to queue j than its length.

(3.c) $x_i(n) > x_j(n) + 1$ and $y_i(n) > y_j(n)$. We consider the following two cases:

(i) $y_i(n) = x_i(n)$, i.e., π *empties queue* i at $t = n$. To construct $\tilde{\omega}$ for this case, we set $\tilde{\mathbf{z}}(n) = \mathbf{z}(n)$, $\tilde{g}_{m,c}(n) = g_{m,c}(n)$ for all $m \neq i, j$, and for all c. For queues i and j we do the following:

Let server r be a server that is connected to queue i at time slot n such that $q_r(n) = i$ (i.e., server r is allocated to queue i by policy π at $t = n$). Now, we switch the connectivity of server r to queue i and that of server r to queue j, i.e., we set $\tilde{g}_{j,r}(n) = g_{i,r}(n)$ and $\tilde{g}_{i,r}(n) = g_{j,r}(n)$ (refer to footnote 7 or [24]). The rest of the servers will have the same connectivities to queues i and j under both policies, i.e., we set $\tilde{g}_{i,c}(n) = g_{i,c}(n)$ and $\tilde{g}_{j,c}(n) = g_{j,c}(n)$ for all $c \neq r$.

We construct $\tilde{\pi}$ such that $\tilde{q}_r(n) = j$ and $\tilde{q}_c(n) = q_c(n), \forall c \neq r$. This means that $\tilde{\pi}$ differs from π, at $t = n$, by one server allocation (server r) that is allocated to queue j (under $\tilde{\pi}$) rather than queue i (under π). From equation (2.1), we can easily calculate that the resulting queue lengths at $t = n + 1$ (for any arrivals pattern) will be:

$$\tilde{x}_m(n+1) \;\; = \;\; x_m(n+1), \qquad \forall\, m.$$

It follows that property (R1) is satisfied and therefore (C.1) is satisfied at $t = n+1$.

(ii) $y_i(n) < x_i(n)$, i.e., π *does not empty queue* i at $t = n$. Then consider the following:

If π does not empty queue j at $t = n$ or if π empties queue j and in the process it

exhausts all servers connected to queue j, i.e., π does not idle any server connected to queue j, then we construct $\tilde{\omega}$ and $\tilde{\pi}$ similar to case (3.c(i)) above and the same conclusion holds.

If on the other hand, π empties queue j without exhausting all its connected servers and therefore π is forced to idle some of the servers connected to queue j, then let r' be one such server. We set $\tilde{\mathbf{z}}(n) = \mathbf{z}(n)$, $\tilde{\mathbf{g}}(n) = \mathbf{g}(n)$. We construct $\tilde{\pi}$ such that $\tilde{y}_j(n) = y_j(n) + 1$, by allocating server r' to queue j under $\tilde{\pi}$, i.e., we set $\tilde{q}_{r'}(n) = j$. This is feasible since $\tilde{x}_j(n) = x_j(n) + 1$ by property (R3). We also have $\tilde{x}_i(n) = x_i(n) - 1$ (by property (R3)). Since $\tilde{\mathbf{z}}(n) = \mathbf{z}(n)$ by construction, then similar to case (3.b(ii)), property (R1) holds with strict inequality at $t = n + 1$ for any arrivals pattern, and (C.1) follows.

Since π cannot allocate more servers to queue j than its length, therefore, (i) and (ii) are the only possible cases.

Note that policy $\tilde{\pi}$ belongs to Π_τ^{h-1} by construction in Part 1; its dominance over π follows from relation (2.15). $\square$

Appendix D

Proof of Lemma 10

Let S and $\bar{S}$ refer to the queuing systems under policies π and $\bar{\pi}$ respectively. Let $\mathbf{X} = (\mathbf{X_1}, \mathbf{X_2})$ and $\bar{\mathbf{X}} = (\bar{\mathbf{X}}_\mathbf{1}, \bar{\mathbf{X}}_\mathbf{2})$ be the queue length sequences under π and $\bar{\pi}$ respectively. Define the following indicator variables to represent the queues with the longest and the shortest sizes in S and $\bar{S}$ at time t:

$$\begin{aligned} s(t) &= \underset{m=1,2}{\operatorname{argmin}}\{X_m(t)\}, \qquad & \bar{s}(t) &= \underset{m=1,2}{\operatorname{argmin}}\{\bar{X}_m(t)\} \\ l(t) &= \underset{m=1,2}{\operatorname{argmax}}\{X_m(t)\}, \qquad & \bar{l}(t) &= \underset{m=1,2}{\operatorname{argmax}}\{\bar{X}_m(t)\} \end{aligned} \tag{D.1}$$

Ties are broken arbitrarily. To simplify notation, we will suppress the time argument for the indicator variables when used as a subscript, e.g., we write $X_s(t)$ rather than $X_{s(t)}(t)$. We also define the partial order $\preceq$ on $\mathcal{Z}_+^2$, the set of ordered pairs of nonnegative integers, as follows:

Definition: We say that $\bar{X}(t) \in \mathcal{Z}_+^2$ is preferred to $X(t) \in \mathcal{Z}_+^2$, written as $\bar{X}(t) \preceq X(t)$, if and only if one of the following three relations is satisfied:

$$X_l(t) \geq \bar{X}_{\bar{l}}(t) \quad and \quad X_s(t) \geq \bar{X}_{\bar{s}}(t) \tag{D.2}$$

$$\begin{aligned} &X_l(t) \geq \bar{X}_{\bar{l}}(t) + 1, \quad X_s(t) \geq \bar{X}_{\bar{s}}(t) - 1 \\ &and \quad \bar{X}_{\bar{l}}(t) - \bar{X}_{\bar{s}}(t) \leq X_l(t) - X_s(t) - 1 \end{aligned} \tag{D.3}$$

$$\begin{aligned} &X_l(t) \geq \bar{X}_{\bar{l}}(t) + 2, \quad X_s(t) \geq \bar{X}_{\bar{s}}(t) - 2 \\ &and \quad \bar{X}_{\bar{l}}(t) - \bar{X}_{\bar{s}}(t) \leq X_l(t) - X_s(t) - 3 \end{aligned} \tag{D.4}$$

Relation (D.2) simply states the usual ordering of the queue lengths of queues $l(t), \bar{l}(t), s(t)$ and $\bar{s}(t)$ on $\mathcal{Z}_+$. Relation (D.3) (respectively relation (D.4)) states that the vector $\bar{X}(t)$ can be obtained from $X(t)$ by performing a one-packet (respectively a two-packet) "balancing interchange" and zero or more "packet removals". We can easily check that

$$\bar{X}(t) \preceq X(t) \rightarrow \bar{W}(t) \leq W(t) \tag{D.5}$$

Proof of Lemma 10. We use stochastic coupling to prove the lemma; for the rest of this section, all (in)equalities are in the almost sure sense. The basic idea is to show that $\bar{X}(t) \preceq X(t)$; then from D.5, $\bar{W}(t) \leq W(t)$ and Equation (3.8) follows.

For notational simplicity, we assume that $\tau = 0$. We couple the systems S and $\bar{S}$ by starting them from the same initial state $x_0 = (X_1(0), X_2(0)) = \bar{x}_0$ and giving them the same connectivity variables, i.e., $g_{i,j}(1) = \bar{g}_{i,j}(1), \forall i, j = 1, 2$. Note that $\bar{\pi}$ will take the same actions as MB policy during time slot $t = 1$.

At $t = 1$, one of the following cases may apply: If $g_{i,j}(1) = 0, \forall i, j = 1, 2$, i.e., if both servers are disconnected or $X_l(0) = X_s(0) = 0$, then $\bar{\pi}(1) = (0, 0)$. Clearly,

no service will be rendered under π or $\bar{\pi}$. At $t = 1$, assign the same arrival variables to the corresponding queues in S and $\bar{S}$. In this case, from Equation (5.1) we can easily check that $\bar{X}(t), X(t)$ satisfy relation (D.2) at $t = 1$. Otherwise, we have the following two mutually exclusive, exhaustive cases (denoted as A and B) to consider: *Case A)* $X_l(0) = X_s(0)$. Consider the following sub-cases:

A1) If at $t = 1$, at least server m (respectively server n) is connected to queue 1 (respectively queue 2) $m, n \in \{1, 2\}$ and $n \neq m$ (refer to Figure D.1(a)), then according to Equation (5.4), $\bar{\pi}(t) = (1, 1)$.

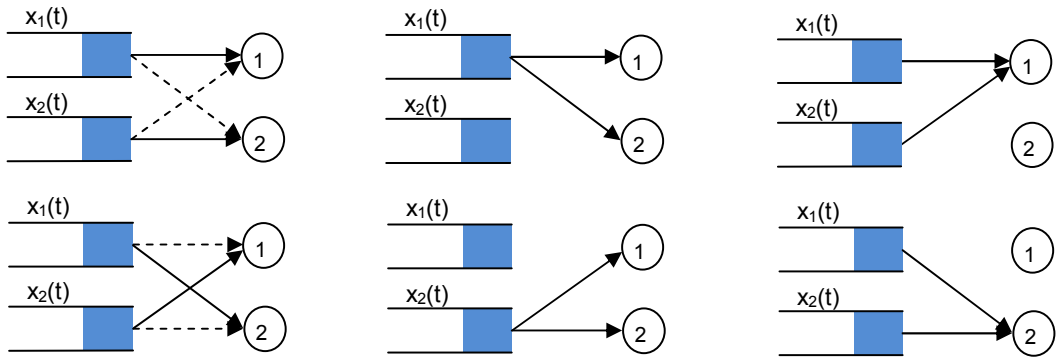
At $t = 1$, let $Z_i(t) = \bar{Z}_i(t)$ and $\mu_k(t) = \bar{\mu}_k(t)$, $i, k = 1, 2$, i.e., the arrival variables for queue $l(t)$ (respectively $s(t)$) are the same as that for queue $\bar{l}(t)$ (respectively $\bar{s}(t)$) and the service completion variable for server k is the same in both S and $\bar{S}$. In this case, the queue length sequences for systems S and $\bar{S}$ may follow one of the following trajectories (depending on the value of $\pi(t)$):

(i) $\pi(t) = (0, 2)$ or $\pi(t) = (2, 0)$ at $t = 1$. Then (D.2) holds if no packet completes its service. If both packets complete their service (i.e., if $\mu_1(n) = \mu_2(n) = 1$), then we can easily verify that (D.2) holds if there is an arrival to the shortest queues (s and $\bar{s}$). For all other arrival patterns, (D.3) holds. If only one packet completes its service then we can verify that (D.2) holds for any arrival pattern.

(ii) Otherwise (D.2) will be satisfied at $t = 1$.

A2) In this sub-case we consider the case where only one queue is connected to either one or both servers (refer to Figure D.1(b)). Assign the same arrival and service variables as in case *A1*. If $\pi(1) = \bar{\pi}(1)$ then (D.2) holds at $t = 1$ with strict equality. Otherwise (π idles servers at $t = 1$), (D.2) holds at $t = 1$ with strict inequality.

A3) If at $t = 1$, $g_{i,m}(t) = g_{j,m}(t) = 1$ and $g_{i,n}(t) = g_{j,n}(t) = 0$, $m, n \in \{1, 2\}$ and $m \neq n$ (a single server is connected to both queues, refer to Figure D.1(c)).



(a) Case A1 (b) Case A2 (c) Case A3

Figure D.1: Different connectivity patterns for Case A (Lemma 10)

Assign the same arrival and service variables as in case *A1*. Then (D.2) holds at $t = 1$ with strict inequality if π idles the connected server and with strict equality otherwise.

Case B) $X_l(0) > X_s(0)$.

We couple the systems S and $\bar{S}$ by assuming they have the same arrival sequences, i.e., $Z_k(t) = \bar{Z}_k(t), k = l(t), s(t)$, and the same service variables for each server. If no packet completes service then (D.2) holds at $t = 1$. Otherwise, the following cases will be considered:

B1) If at $t = 1$, $g_{l,k}(t) = 1$, $k = 1, 2$ then $\bar{\pi}(t) = (\bar{Y}_{\bar{l}}(t), \bar{Y}_{\bar{s}}(t)) = (2, 0)$. One of the following three cases may apply:

(i) If $X_l(0) > X_s(0) + 2$ then at $t = 1$ if $\pi(t) = (1, 1)$ then let $\bar{\mu}_i(t) = \mu_i(t)$ and $\bar{\mu}_j(t) = \mu_j(t)$, i.e., packets served by either server i or server j have the same service completion variables in S and $\bar{S}$. Now if only the packet in queue $l(t)$ completes its service then (D.2) holds; otherwise, we can verify that (D.3) holds for the remaining cases. Else if $\pi(t) = (0, 2)$ then let $\bar{\mu}_i(t) = \mu_i(t)$ and $\bar{\mu}_j(t) = \mu_j(t)$. If both scheduled packets complete service, then (D.4) holds. If only one packet completes its service, e.g., $\mu_i(t) = 0$ and $\mu_j(t) = 1$, then (D.3) holds. Otherwise, $\pi(t) = \bar{\pi}(t)$ and (D.2) holds at $t = 1$.

(ii) If $X_l(0) = X_s(0) + 2$, then, at $t = 1$ if $\pi(t) = (1, 1)$ then let $\bar{\mu}_i(t) = \mu_i(t)$ and $\bar{\mu}_j(t) = \mu_j(t)$. If only the packet in queue $l(t)$ completes its service or if service is completed for both servers and there are arrivals to queues $s(t)$ and $\bar{s}(t)$ only during time slot $t = 1$, then (D.2) holds; otherwise, (D.3) holds at $t = 1$. Else if $\pi(t) = (0, 2)$ then let $\bar{\mu}_i(t) = \mu_i(t)$ and $\bar{\mu}_j(t) = \mu_j(t)$. If service is completed, then (D.4) holds. If only one packet completes its service, then (D.3) holds. Otherwise, $\pi(t) = \bar{\pi}(t)$ and (D.2) holds at $t = 1$.

(iii) If $X_l(0) = X_s(0) + 1$, then, at $t = 1$ if $\pi(t) = (1, 1)$ then let $\bar{\mu}_i(t) = \mu_i(t)$ and $\bar{\mu}_j(t) = \mu_j(t)$. If only the packet in queue $s(t)$ completes service and there are no arrivals to queues $s(t)$ and $\bar{s}(t)$ during time slot $t = 1$, then (D.3) holds; otherwise, (D.2) holds at $t = 1$. Else if $\pi(t) = (0, 2)$ then let $\bar{\mu}_i(t) = \mu_i(t)$ and $\bar{\mu}_j(t) = \mu_j(t)$. If service is completed, then (D.3) holds. If only one packet completes its service and there are arrivals to queues $s(t)$ and $\bar{s}(t)$ only during time slot $t = 1$ then (D.2) holds. Otherwise, $\pi(t) = \bar{\pi}(t)$ and (D.2) holds at $t = 1$.

B2) If at $t = 1$, $g_{l,i}(t) = 1$ and $g_{l,j}(t) = 0, i, j \in \{1, 2\}$, then $\bar{\pi}(t) = (1, 1)$. We choose the same service variables for each pair of the scheduled packets in S and $\bar{S}$. If $\pi(t) = (0, 2)$ at $t = 1$, then if service is not completed then (D.2) holds at $t = 1$. If $\bar{\mu}_i(t) = 0$ and $\bar{\mu}_j(t) = 1$, i.e., the packet in $\bar{s}(t)$ completes its service while the packet in $\bar{l}(t)$ does not, then (D.2) holds at $t = 1$. Otherwise, (D.3) holds at $t = 1$. Else if $\pi(t) = \bar{\pi}(t)$ then (D.2) holds at $t = 1$.

B3) If at $t = 1$, $g_{l,k}(t) = 0,\ k = 1, 2$, then $\pi(t) = \bar{\pi}(t)$. Assign the same service variables for each pair of the scheduled packets in S and $\bar{S}$. In this case, (D.2) holds at $t = 1$.

The above two cases (A and B) are the only possible ones, because of the definition of $\bar{\pi}$. We showed that in both, the total queue length at $t = 1$ satisfies the relation

$$\bar{X}(t) \preceq X(t). \tag{D.6}$$

To complete the induction we show that if the queue lengths at $t = n$ satisfy either (D.2), (D.3), or (D.4) then at $t = n + 1$ we can couple the queue length sequences $\mathbf{X}$ and $\bar{\mathbf{X}}$ by carefully selecting the arrival, connectivity, and service variables and construct $\bar{\pi}$ such that the queue lengths at $t = n+1$ satisfy one of the relations (D.2), (D.3), or (D.4). As a result, (D.6) will hold at $t = n + 1$.

We must consider the following three cases: (1) Relation (D.2) holds at $t = n$; (2) Relation (D.3) holds at $t = n$; (3) Relation (D.4) holds at $t = n$.

Case 1) Relation (D.2) holds at $t = n$. At $t = n + 1$, assign the same arrival, service and connectivity variables to queues $l(t)$ and $\bar{l}(t)$ and similarly for queues $s(t)$ and $\bar{s}(t)$. Let $\bar{\pi}(t) = \pi(t)$ at $t = n + 1$. Then (D.2) will be satisfied again at $t = n + 1$.

Case 2) Relation (D.3) holds at $t = n$. At $t = n+1$, couple the systems by assigning the same arrival variables to queues $l(t)$ and $\bar{l}(t)$ (do the same for queues $s(t)$ and $\bar{s}(t)$). Also assign the same service variables for the servers in S and $\bar{S}$. If service is not completed for both scheduled packets, then let both systems have the same connectivity variables and let $\bar{\pi}(t) = \pi(t)$; then (D.3) holds at $t = n + 1$. Otherwise, we consider the following cases:

2a) $\bar{X}_{\bar{l}}(n) > \bar{X}_{\bar{s}}(n) + 2$. At $t = n + 1$, let $g_{l,k}(t) = \bar{g}_{\bar{l},k}(t)$ and $g_{s,k}(t) = \bar{g}_{\bar{s},k}(t),\ k = 1, 2$. Let $\bar{\pi}(t) = \pi(t)$ at $t = n + 1$. Then (D.3) holds at $t = n + 1$.

2b) $\bar{X}_{\bar{l}}(n) = \bar{X}_{\bar{s}}(n) + 2$. At $t = n + 1$, let $g_{l,k}(t) = \bar{g}_{\bar{l},k}(t)$ and $g_{s,k}(t) = \bar{g}_{\bar{s},k}(t),\ k = 1, 2$. Let $\bar{\pi}(t) = \pi(t)$ at $t = n + 1$. Then, when $\pi(t) = (2, 0)$ at $t = n + 1$ and if both packets complete their service (i.e., $\mu_1(n) = \mu_2(n) = 1$) and there are arrivals to queues $s(t)$ and $\bar{s}(t)$ only, then (D.2) holds; otherwise (D.3) holds at $t = n + 1$.

2c) $\bar{X}_{\bar{l}}(n) = \bar{X}_{\bar{s}}(n) + 1$. At $t = n + 1$, let $g_{l,k}(t) = \bar{g}_{\bar{l},k}(t)$ and $g_{s,k}(t) = \bar{g}_{\bar{s},k}(t),\ k = 1, 2$. Let $\bar{\pi}(t) = \pi(t)$ at $t = n + 1$. If $\pi(t) = (2, 0)$ or if $\pi(t) = (1, 1)$ and service is completed for the packet in queue $l(t)$ only and there is an arrival to queue $s(t)$ only, then (D.2) holds; otherwise (D.3) holds at $t = n + 1$.

2d) $\bar{X}_{\bar{l}}(n) = \bar{X}_{\bar{s}}(n)$. There are several cases to consider:

(i) At $t = n+1$, $\pi(t) = (1,1)$. Let $g_{l,k}(t) = \bar{g}_{\bar{l},k}(t)$ and $g_{s,k}(t) = \bar{g}_{\bar{s},k}(t),\ k = 1,2$. Let $\bar{\pi}(t) = (1,1)$ at $t = n+1$, then if during time slot $t = n+1$ service is completed for both scheduled packets and there are arrivals only to queues $s(t)$ and $\bar{s}(t)$, then (D.2) holds, otherwise (D.3) holds at $t = n+1$.

(ii) At $t = n+1$, $\pi(t) = (2,0)$. Then let $g_{l,i}(t) = \bar{g}_{\bar{l},i}(t)$ and $g_{l,j}(t) = \bar{g}_{\bar{s},j}(t),\ i,j \in \{1,2\},\ i \neq j$. Similarly, let $g_{s,i}(t) = \bar{g}_{\bar{s},i}(t)$ and $g_{s,j}(t) = \bar{g}_{\bar{l},j}(t)$ at $t = n+1$. Let $\bar{\pi}(t) = (1,1)$ at $t = n+1$, then (D.2) holds at $t = n+1$.

(iii) Otherwise, let $\bar{\pi}(t) = \pi(t)$ at $t = n+1$. Use the connectivity assumptions in case $(2d(i))$ in this case. Then (D.3) holds at $t = n+1$.

Case 3) Relation (D.4) holds at $t = n$. At $t = n+1$, let the arrival and service variables be the same as in Case 2 above. In addition let $g_{l,k}(t) = \bar{g}_{\bar{l},k}(t)$ and $g_{s,k}(t) = \bar{g}_{\bar{s},k}(t),\ k = 1,2$ at $t = n+1$. If service is not completed for both scheduled packets then (D.4) holds at $t = n+1$. Otherwise, the following may apply[1]:

3a) $\bar{X}_{\bar{l}}(n) > \bar{X}_{\bar{s}}(n) + 2$. At $t = n+1$, let $\bar{\pi}(t) = \pi(t)$. Then (D.4) holds at $t = n+1$.

3b) $\bar{X}_{\bar{l}}(n) = \bar{X}_{\bar{s}}(n) + 2$. At $t = n+1$, let $\bar{\pi}(t) = \pi(t)$. Then if $\pi(t) = (2,0)$ at $t = n+1$, and if service is completed for both scheduled packets and there are arrivals to queues $s(t)$ and $\bar{s}(t)$ only, then (D.3) holds at $t = n+1$; otherwise (D.4) holds at $t = n+1$.

3c) $\bar{X}_{\bar{l}}(n) = \bar{X}_{\bar{s}}(n) + 1$. At $t = n+1$, let $\bar{\pi}(t) = \pi(t)$. Then either (D.3) or (D.4) holds at $t = n+1$ depending on the arrivals and service completions during that time slot.

3d) $\bar{X}_{\bar{l}}(n) = \bar{X}_{\bar{s}}(n)$. Let $\bar{\pi}(t) = \pi(t)$ at $t = n+1$. There are several cases to consider:

(i) At $t = n+1$, $\pi(t) = (1,1)$. Then either (D.3) or (D.4) holds at $t = n+1$

[1]The arguments in this case are analogous to those in Case 2 above and therefore were summarized when possible.

depending on the arrivals and service completions during that time slot.

(ii) At $t = n + 1$, $\pi(t) = (2, 0)$. Then (D.2), (D.3) or (D.4) may hold at $t = n + 1$ depending on the arrivals and service completions during that time slot.

(iii) Otherwise, (D.4) holds at $t = n + 1$.

The above three cases (i.e., Case 1, Case 2 and Case 3) satisfy equation (D.6) at $t = n + 1$. By forward induction, we can conclude that (D.6) is satisfied for every $t \in T$ and Lemma 10 follows. □

Appendix E

Derivation of Transition Probabilities in Chapter 4

E.1 The System State Transition Probability in Section 4.3.5

To derive the state transition probability ($P_{\mathbf{ss'}}(\mathbf{a})$) that was introduced in section 2.3.2, we start from equation (4.7) as follows

$$\begin{aligned} P'_{\boldsymbol{ss}}(\boldsymbol{a}) &\triangleq Pr\big(\boldsymbol{s}(t+1)=\boldsymbol{s}'|\boldsymbol{s}(t)=\boldsymbol{s},\boldsymbol{a}(t)=\boldsymbol{a}\big) \\ &= Pr(x'_1,\ldots,x'_L,\gamma'_1,\ldots,\gamma'_L|x_1,\ldots,x_L,\gamma_1,\ldots,\gamma_L,a_1,\ldots,a_L) \quad \text{(E.1)} \end{aligned}$$

where x_i denotes the queue size of user i, and γ_i is the FSMC state characterizing the connectivity of user i wireless channel. Using conditioning we can decompose the above joint probability as follows:

$$\begin{aligned} P_{\boldsymbol{ss'}}(\boldsymbol{a}) &= Pr(x'_1,\ldots,x'_L|x_1,\ldots,x_L,\gamma_1,\ldots,\gamma_L,a_1,\ldots,a_L) \\ &\quad \cdot Pr(\gamma'_1,\ldots,\gamma'_L|x'_1,\ldots,x'_L,x_1,\ldots,x_L,\gamma_1,\ldots,\gamma_L,a_1,\ldots,a_L) \quad \text{(E.2)} \end{aligned}$$

Applying conditioning again, the second part of equation (E.2) yields

$$\begin{aligned} P_{\boldsymbol{ss'}}(\boldsymbol{a}) &= Pr(x'_1,\ldots,x'_L|x_1,\ldots,x_L,\gamma_1,\ldots,\gamma_L,a_1,\ldots,a_L) \\ &\cdot Pr(\gamma'_1|x'_1,\ldots,x'_L,x_1,\ldots,x_L,\gamma_1,\ldots,\gamma_L,a_1,\ldots,a_L) \\ &\cdot Pr(\gamma'_2|\gamma'_1,x'_1,\ldots,x'_L,x_1,\ldots,x_L,\gamma_1,\ldots,\gamma_L,a_1,\ldots,a_L)\cdot\ldots \\ &\cdot Pr(\gamma'_L|\gamma'_1,\ldots,\gamma'_{L-1},x'_1,\ldots,x'_L,x_1,\ldots,x_L,\gamma_1,\ldots,\gamma_L,a_1,\ldots,a_L) \end{aligned} \tag{E.3}$$

Since the wireless channel was modeled by means of a Markov process, the channel state γ_i depends only on the most recent channel state. Hence, the channel state transition probability can be written as:

$$Pr(\gamma'_i|\boldsymbol{s}) = Pr(\gamma'_i|\gamma_i) \triangleq P_{\gamma_i\gamma'_i}$$

Accordingly, we can rewrite (E.3) as follows:

$$P_{\boldsymbol{ss'}}(\boldsymbol{a}) = Pr(x'_1,\ldots,x'_L|x_1,\ldots,x_L,\gamma_1,\ldots,\gamma_L,a_1,\ldots,a_L)\cdot\prod_{i=1}^{L}P_{\gamma_i\gamma'_i} \tag{E.4}$$

Following the same approach, the joint probability of the queue size (first term in equation (E.2)) can be decomposed as follows:

$$\begin{aligned} &Pr(x'_1,\ldots,x'_L|x_1,\ldots,x_L,\gamma_1,\ldots,\gamma_L,a_1,\ldots,a_L) = \\ &\qquad = Pr(x'_1|x_1,\ldots,x_L,\gamma_1,\ldots,\gamma_L,a_1,\ldots,a_L) \\ &\qquad \cdot Pr(x'_2|x'_1,x_1,\ldots,x_L,\gamma_1,\ldots,\gamma_L,a_1,\ldots,a_L)\cdot\ldots \\ &\qquad \cdot Pr(x'_L|x'_1,\ldots,x'_{L-1},x_1,\ldots,x_L,\gamma_1,\ldots,\gamma_L,a_1,\ldots,a_L) \end{aligned} \tag{E.5}$$

The evolution of the queue size (x_i) is given by

$$\begin{aligned} x_i' &= \min\left([x_i - y_i]^+ + z_i', B\right) \\ &= \min\left([x_i - a_i\gamma_i c]^+ + z_i', B\right) \end{aligned} \tag{E.6}$$

Since the queue size corresponding to user i at the next time slot (x_i') depends on its current queue size (x_i), the given action a_i, its channel state γ_i and the arrived PDUs z_i' during $(t, t+1]$ and is independent of all other queue sizes, actions and channel conditions corresponding to the remaining users. Hence

$$Pr(x_i'|\boldsymbol{s}) = Pr(x_i'|x_i, \gamma_i, a_i) \triangleq P_{x_i x_i'}(\gamma_i, a_i)$$

Therefore, equation (E.5) reduces to

$$P_{\boldsymbol{s}\boldsymbol{s}'}(\boldsymbol{a}) = \prod_{i=1}^{L} P_{x_i x_i'}(\gamma_i, a_i)\, P_{\gamma_i \gamma_i'} \tag{E.7}$$

The state transition probability will be the product of the individual user queues state transition probabilities and their FSMC channel transition probabilities. The underlining assumption is that $P_{\gamma_i\gamma_i'}$ can in practice be estimated from measurements and provided to the scheduler. The term $P_{x_i x_i'}(\gamma_i, a_i)$ will be derived in the following section.

E.2 The Queue State Transition Probability in Section 4.3.5

The queue transition probability is given by

$$P_{x_i x_i'}(\gamma_i, a_i) = Pr\big(x_i(t+1) = x_i' | x_i(t) = x_i, \gamma_i(t) = \gamma_i, a_i(t) = a_i\big) \tag{E.8}$$

where

$$x_i(t+1) = \min\left([x_i(t) - y_i(t)]^+ + z_i(t+1)\,, B\right) \tag{E.9}$$

with $y_i(t) = a_i(t)\gamma_i(t)c$. From equation (E.9) we can differentiate two cases; (a) $[x_i(t) - y_i(t)]^+ + z_i(t+1) < B$ and (b) $[x_i(t) - y_i(t)]^+ + z_i(t+1) \geq B$. The objective of this section is to derive $P_{x_i x_i'}(\gamma_i, a_i)$ for both cases. Substituting the two cases in equation (E.8) yields the following

Case (a): $[x_i(t) - y_i(t)]^+ + z_i(t+1) < B$ or equivalently

$$x_i(t+1) = [x_i(t) - y_i(t)]^+ + z_i(t+1)$$

Equation (E.8) in this case can be rewritten as:

$$\begin{aligned} P_{x_i x_i'}(\gamma_i, a_i) &= Pr\big([x_i(t) - y_i(t)]^+ + z_i(t+1) = x_i' \,|x_i(t) = x_i, \gamma_i(t) = \gamma_i, a_i(t) = a_i\big) \\ &= Pr\big([x_i - y_i]^+ + z_i(t+1) = x_i'\big) \\ &= Pr(z_i(t+1) = x_i' - [x_i - y_i]^+\big) \end{aligned} \tag{E.10}$$

where $y_i = a_i\gamma_i c$. The arrival process is assumed to be Bernoulli with parameter q_i

for user i (section 4.3.1) and is given by

$$z_i(t) = \begin{cases} u_i & \text{with probability } q_i \\ 0 & \text{with probability } 1 - q_i \end{cases} \tag{E.11}$$

Hence, the queue state transition probability in this case is

$$P_{x_i x_i'}(\gamma_i, a_i) = \begin{cases} q_i & \text{if } x_i' = [x_i - y_i]^+ + u_i \\ 1 - q_i & \text{if } x_i' = [x_i - y_i]^+ \\ 0 & \text{Otherwise} \end{cases} \tag{E.12}$$

Case (b): $[x_i(t) - y_i(t)]^+ + z_i(t+1) \geq B$ or equivalently $x_i(t+1) = B$. In this case, one can conclude that $P_{x_i x_i'}(\gamma_i, a_i) = 0$ when $x_i' \neq B$. The remaining of this section is devoted to the calculation of $P_{x_i B}(\gamma_i, a_i)$ as follows:

$$\begin{aligned} P_{x_i B}(\gamma_i, a_i) &= Pr\big([x_i(t) - y_i(t)]^+ + z_i(t+1) \geq B \,|x_i(t) = x_i, \gamma_i(t) = \gamma_i, a_i(t) = a_i \big) \\ &= Pr\big([x_i - y_i]^+ + z_i(t+1) \geq B \big) \\ &= Pr(z_i(t+1) \geq B - [x_i - y_i]^+) \end{aligned} \tag{E.13}$$

Similar to case (a), we can write equation (E.13) as follows

$$P_{x_i B}(\gamma_i, a_i) = \begin{cases} 1 - q_i & \text{if } [x_i - y_i]^+ \geq B \\ q_i & \text{if } [x_i - y_i]^+ + u_i \geq B \\ 0 & \text{Otherwise} \end{cases} \tag{E.14}$$

Using our knowledge of queue evolution process, we can summarize all the possible

transitions in (E.14) by partitioning the probability space of $P_{x_iB}(\gamma_i, a_i)$ as follows:

$$P_{x_iB}(\gamma_i, a_i) = \begin{cases} 1 - q_i & \text{if } x_i = B,\ \gamma_i a_i = 0 \\ q_i & \text{if } x_i = B,\ \gamma_i a_i = 0 \\ q_i & \text{if } x_i = B,\ 0 < \gamma_i a_i c \leq u_i \\ q_i & \text{if } x_i < B,\ [x_i - \gamma_i a_i c]^+ + u_i \geq B \\ 0 & \text{Otherwise} \end{cases} \tag{E.15}$$

where the first case in equation (E.15) (corresponding to probability $1 - q_i$) is the only possible conditions for the first case in equation (E.14). The other three cases (corresponding to probability q_i) represent all the possible partitions of the probability space of the second case in equation (E.14).

The results obtained above in (E.12) and (E.15) can be summarized as follows

$$P_{x_i x_i'}(\gamma_i, a_i) = \begin{cases} 1 & \text{if } x_i' = x_i = B,\ \gamma_i a_i = 0 \\ q_i & \text{if } x_i' = x_i = B,\ 0 < \gamma_i a_i c \leq u_i \\ q_i & \text{if } x_i' = B,\ x_i < B,\ [x_i - \gamma_i a_i c]^+ + u_i \geq B \\ q_i & \text{if } x_i' < B,\ x_i' = [x_i - \gamma_i a_i c]^+ + u_i \\ 1 - q_i & \text{if } x_i' < B,\ x_i' = [x_i - \gamma_i a_i c]^+ \\ 0 & \text{Otherwise} \end{cases} \tag{E.16}$$

E.3 The Queue State Transition Probability in section 4.5.3

In this section, we present the derivation of queue state transition probabilities when retransmission is considered (section 4.5.3). There are two parts for this derivation;

(a) when the transmission is successful (equation (4.17)) and (b) when the transmission is unsuccessful (equation (4.18)). In case (a), the conditional probability $P_{x_i x_i'|\mu_i=1}(\gamma_i, a_i)$ is similar to that in equation (E.8) and the derivation is analogous to that in the previous section and will not be repeated here.

The remainder of this section is devoted to derive the probability in case (b); namely, the queue transition probability when the transmission is unsuccessful (i.e., $\mu_i(t) = 0$). In this case, no PDUs are removed from the scheduled queues at the end of the current TTI (since the transmission was unsuccessful). The conditional queue state transition probability is given by

$$P_{x_i x_i'|\mu_i=0}(\gamma_i, a_i) \triangleq Pr\big(x_i(t+1) = x_i' | x_i(t) = x_i, \gamma_i(t) = \gamma_i, a_i(t) = a_i, \mu_i = 0\big) \quad \text{(E.17)}$$

where $x_i(t+1)$ is given by equation (4.14). Similar to section E.2, we can distinguish the following two cases: case (b1): $[x_i(t) - y_i(t)\mu_i(t)]^+ + z_i(t+1) < B$; equivalently,

$$x_i(t+1) = [x_i(t) - y_i(t)\mu_i(t)]^+ + z_i(t+1)$$

Equation E.17 can be rewritten as:

$$\begin{aligned} P_{x_i x_i'|\mu_i=0}(\gamma_i, a_i) &= Pr\big([x_i(t) - y_i(t)\mu_i(t)]^+ + z_i(t+1) = x_i' \,|x_i(t) = x_i, \\ &\quad \gamma_i(t) = \gamma_i, a_i(t) = a_i, \mu_i = 0\big) \\ &= Pr(x_i + z_i(t+1) = x_i') \\ &= Pr(z_i(t+1) = x_i' - x_i) \end{aligned} \quad \text{(E.18)}$$

As mentioned previously, $z_i(t)$ has Bernoulli distribution with parameter q_i. Therefore, the marginal queue state transition probability is given by

$$P_{x_i x_i'|\mu_i=0}(\gamma_i, a_i) = \begin{cases} q_i & \text{if } x_i' = x_i + u_i \\ 1 - q_i & \text{if } x_i' = x_i \\ 0 & \text{Otherwise} \end{cases} \tag{E.19}$$

case (b2): $[x_i(t) - y_i(t)\mu_i(t)]^+ + z_i(t+1) \geq B$ equivalently $x_i(t+1) = B$, then equation E.17 can be rewritten as

$$\begin{aligned} P_{x_i B|\mu_i=0}(\gamma_i, a_i) &= Pr([x_i(t) - y_i(t)\mu_i(t)]^+ + z_i(t+1) \geq B \,|x_i(t) = x_i, \\ &\quad \gamma_i(t) = \gamma_i, a_i(t) = a_i, \mu_i = 0) \\ &= Pr(x_i + z_i(t+1) \geq B) \\ &= Pr(z_i(t+1) \geq B - x_i) \end{aligned} \tag{E.20}$$

and $P_{x_i x_i'|\mu_i=0}(\gamma_i, a_i) = 0$ when $x_i' \neq B$.

Using equation (E.11), we can conclude that

$$P_{x_i B|\mu_i=0}(\gamma_i, a_i) = \begin{cases} q_i & \text{if } x_i + u_i \geq B \text{ and } x_i < B \\ 1 & \text{if } x_i = B \\ 0 & \text{Otherwise} \end{cases} \tag{E.21}$$

The second case in equation E.21 is the aggregation of two events (arrival and no arrival), since both events will result in $x_i' = B$ when $x_i = B$, i.e., if the queue is full initially and no PDUs are removed from this queue in the current TTI (due to unsuccessful transmission), then the queue will remain full regardless of the arrival status.

Combining equations (E.19) and (E.21) yields

$$P_{x_i x_i' | \mu_i = 0}(\gamma_i, a_i) = \begin{cases} 1 & \text{if } x_i' = x_i = B \\ q_i & \text{if } x_i' = B, x_i < B, x_i + u_i \geq B \\ q_i & \text{if } x_i' < B, x_i' = x_i + u_i \\ 1 - q_i & \text{if } x_i' < B, x_i' = x_i \\ 0 & \text{Otherwise} \end{cases} \tag{E.22}$$