Packet scheduling in wireless systems by Token Bank Fair Queueing Algorithm

by

WILLIAM KING WONG

B.A.Sc., The University of Western Ontario, 1991
M.A.Sc., The University of Western Ontario, 1993
M.Sc., The University of Western Ontario, 1994

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES

(ELECTRICAL AND COMPUTER ENGINEERING)

THE UNIVERSITY OF BRITISH COLUMBIA

July 2005

ABSTRACT

The amalgamation of wireless systems is a vision for the next generation wireless systems where multimedia applications will dominate. One of the challenges for the operators of next generation networks is the delivery of quality-of-service (QoS) to the end users. Wireless scheduling algorithms play a critical and significant role in the user perception of QoS. However, the scarcity of wireless resources and its unstable channel condition present a challenge to providing QoS. To that end, this thesis proposes a novel scheduling algorithm Token Bank Fair Queuing (TBFQ) that is simple to implement and has the *soft* tolerance characteristic which is suitable under wireless constraints. TBFQ combines both token-based policing mechanism and fair throughput prioritization rule in scheduling packets over packet-switched wireless systems. This results in a simple design and yet achieves better performance and fairness than existing wireless scheduling strategies.

The characterization of the algorithm, which leads to its tight bound performance, is conducted through numerical analysis that recursively determines the steady-state probability distribution of the bandwidth allocation process and ultimately the probability distribution of input queue. We found that the convergence of the recursion is remarkably fast and provides an accurate model and gives a tight bound. The performance of the algorithm is further analyzed through simulation in both TDMA and CDMA systems. In CDMA systems, the algorithm is modified and enhanced to take advantage of the advanced features offered by such system. Furthermore, a novel scheduling framework is proposed as part of the strategy for the deployment of the algorithm within a wireless network. An appropriate call admission control algorithm is also proposed for TBFQ.

TABLE OF CONTENTS

Abstract ii

Table of Contents iii

List of Tables v

List of Figures vi

Glossary viii

Preface xi

Acknowledgement xii

Dedications xiii

CHAPTER 1 INTRODUCTION 1

1.1 Radio resource management 3
1.1.1 Medium access control 5
1.1.2 Scheduling algorithms 9
1.1.3 Call admission control 10
1.2 Heterogeneous services 12
1.3 Thesis objectives and contributions 13
1.4 Organization of the thesis 15

CHAPTER 2 OVERVIEW OF SCHEDULING ALGORITHMS 16

2.1 Wired network algorithms 16
2.1.1 First Come First Serve (FCFS) 16
2.1.2 Round Robin Service Policies 17
2.1.3 Packet-Based Round Robin (PBRR) 17
2.1.4 Deficit Round Robin (DRR) 17
2.1.5 Surplus Round Robin (SRR) 18
2.1.6 Elastic Round Robin (ERR) 18
2.1.7 Packetized GPS (PGPS) or Weighted Fair Queueing (WFQ) 19
2.1.8 Worst-case Fair Weighted Fair Queueing (WF2Q) 20
2.1.9 Self-Clocked Fair Queueing (SCFQ) 20
2.1.10 Earliest Deadline First (EDF) 21
2.1.11 Delay Earliest Due Date (Delay-EDD) 21
2.1.12 Jitter Earliest Due Date (Jitter-EDD) 23
2.2 Wireless network algorithms 24
2.2.1 Channel State Dependent Packet Scheduling (CSDPS) 24
2.2.2 Idealized Wireless Fair Queuing (IWFQ) 25
2.2.3 Channel-Condition Independent Packet Fair Queueing (CIF-Q) 25
2.2.4 Opportunity-Based Deficit Round Robin (ODRR) 26

CHAPTER 3 TOKEN BANK FAIR QUEUING ALGORITHM 27

3.1 TBFQ algorithm 27
3.2 Work Complexity 35
3.3 Rate guarantee 37
3.4 Fairness 37
3.5 Latency Rate Servers 43
3.6 Graceful services 50

CHAPTER 4 MODELING OF TBFQ 55
4.1 Challenges in mathematical modeling of TBFQ 55
4.2 Mathematical modeling 56
4.3 Methodologies 60
4.3.1 Approximating the service process 60
4.3.2 Precedence relation method 60
4.4 Network calculus analysis 61
4.5 Recursive approach 61
4.6 Implementation of the Recursive Approach 63
4.7 Performance analysis 69
CHAPTER 5 TBFQ IN TDMA TDD-BASED HETEROGENEOUS WIRELESS NETWORKS 75
5.1 Desired properties of wireless scheduling algorithms 75
5.2 QoS-based framework and MAC model 77
5.3 Adaptation of TBFQ in TDMA 80
5.4 Call admission control algorithm 82
5.5 Traffic model parameters 84
5.6 Downlink performance 85
5.7 Uplink performance under error conditions 92
5.8 Isolation performance 98
5.9 Location-dependent error conditions 99
CHAPTER 6 TBFQ IN WCDMA HIGH SPEED DOWNLINK PACKET ACCESS (HSDPA) 103
6.1 Introduction 103
6.2 New Node-B Requirements 108
6.3 Protocol Architecture Changes 109
6.4 MAC Layer Architecture 110
6.5 TBFQ Scheduling in HSDPA 114
6.6 Simulation configuration 115
6.7 Performance analysis 118
CHAPTER 7 CONCLUSIONS AND FUTURE RESEAERCH RECOMENDATIONS 121
7.1 Conclusions 121
7.2 Future research recommendations 122
Bibliography 124
Appendix A : Implementation and Analysis using on-off voice traffic 132
A.1 Uniform On-Off Voice connections 132
A.2 Implementation 137

LIST OF TABLES

Table 1.1	3G (UMTS) bearer service classes	12
Table 1.2	Some 3G (UMTS) service attribute/parameter ranges	13
Table 4.1:	Convergence of the computation of mean queue length, $\bar{Q}$ in bytes.	71
Table 4.2:	Average buffer occupancy, measured in bytes.	71
Table 4.3:	Mean packet delay, measured in ms.	72
Table 4.4:	Average buffer occupancy, measured in bytes.	73
Table 4.5:	Mean packet delay, measured in ms.	73
Table 5.1	TDMA/TDD MAC FRAME STRUCTURE PARAMETERS	79
Table 5.2	PARAMETERS OF H.263 VIDEO SOURCE	85
Table 5.3	Scheduler parameters for voice and video traffic	86
Table 5.4	Properties of the 6 flows in the simulations	100
Table 6.1	HSDPA Throughput Rates	105
Table 6.2	Data services traffic model and QoS requirements	117
Table 6.3	Simulation parameters for HSDPA	118
Table 6.4	Selected MCS and parameters [103]	118
Table 6.5	TBFQ Parameters in HSDPA simulations	118

LIST OF FIGURES

Figure 1.1 Characteristics of future mobile broadband wireless networks 2

Figure 1.2 Radio resource management locations in future broadband mobile networks 4

Figure 1.3 Classification of wireless MAC protocols 6

Figure 1.4 Functions of radio resource management for QoS provisioning 9

Figure 1.5 Demonstration of a call admission problem. 10

Figure 2.1 (a) The arrival pattern; (b) GPS servicing order; (c) The WFQ servicing order); (d) WF^2Q servicing 20

Figure 2.2 Scheduler Saturation 22

Figure 2.3 A Jitter-EDD node 23

Figure 3.1 Generic token bank fair queuing model 29

Figure 3.2 An illustration of the token pool operation in TBFQ 30

Figure 3.3 An illustration of TBFQ operation with bursty traffic. 32

Figure 3.4 Pseudo code of generic TBFQ 33

Figure 3.5 Relationship between token balance counter E and *Sent*(t) 41

Figure 3.6 Two busy periods for flow *i*. 45

Figure 3.7 An example of the behaviour of an *LR* server 47

Figure 4.1: Cumulative probabilities of buffer occupancy for n = 5, 10, 15, and 20. r_x=32Kb/s. 72

Figure 4.2: Cumulative probabilities of buffer occupancy for n = 5, 10, 15, and 20. r_x=25Kb/s. 73

Figure 4.3 Cumulative distribution comparison between theoretical analysis and simulation (r_x=32Kb/s, n_x=5). 74

Figure 5.1 Network architecture of multi-access multi-service networks. 78

Figure 5.2 Format of the TDMA/TDD MAC frame. 79

Figure 5.3 Structure of TBFQ for scheduling both uplink and downlink traffic at the Base Station. 81

Figure 5.4 Pseudo-code for TDMA slot allocation algorithm using TBFQ. 82

Figure 5.5 CAC algorithm used in conjunction with TBFQ algorithm. 84

Figure 5.6 Mean packet delay of voice traffic source. No packet discard is used. 87

Figure 5.7 Delay jitter of voice traffic source. No packet discard is used. 88

Figure 5.8 Violation Probability of voice packets that has exceeded the 20ms delay limit. Without CAC. 88

Figure 5.9 Video traffic frame mean delay 89

Figure 5.10 Video frame delay jitter 90

Figure 5.11 Throughput measurement of WWW traffic source. 91

Figure 5.12 Violation Probability versus Token Rates for video traffic source. 92

Figure 5.13 Packet delay CDF for voice traffic source only (load=0.86) 93

Figure 5.14 Frame delay CDF for video traffic source only (load=0.73) 95
Figure 5.15 Throughput CDF for data traffic source only (0.81 loading) 96
Figure 5.16 Voice Packet Mean Delay with video traffic source. 97
Figure 5.17 Admissible region of voice and video traffic source. 98
Figure 5.18 Isolation performance between well-behaved and 'malicious' video traffic source. 99
Figure 5.19 Serviced received by each data traffic source. Note that Data-1 and -3 are at the bottom among the curves. 101
Figure 5.20 Difference between the actual services received by the data traffic source under error conditions in Table 5.4 and the expected amount of service under error-free condition. 102
Figure 6.1 Time and code multiplexing of HSDPA downlink shared channels 106
Figure 6.2 Network Element Layout 108
Figure 6.3 HSDPA Protocol Architecture 109
Figure 6.4 MAC-hs on the UE 111
Figure 6.5 MAC-hs on the Node-B 112
Figure 6.6 MAC-hs Protocol Data Unit 113
Figure 6.7 Model of TBFQ+maxCI or TBFQ+PF 115
Figure 6.8 Cell layout of the simulation experiments 116
Figure 6.9 Data service traffic model parameter definition 117
Figure 6.10 Cumulative Distribution Function of UE throughput in the centre cell. 119
Figure 6.11 Cumulative Distribution Function of packet delay. 120

GLOSSARY

16QAM	16 Quadrature Amplitude Modulation
3G	Third Generation
3G+	Beyond 3G
3GPP	Third Generation Partnership Project
4G	Fourth Generation
ABR	Available-bit-rate
ACK	Acknowledge message
AFB	Absolute Fairness Bound
AMC	Adaptive Modulation and Coding
ARQ	automatic-repeat-request
B3G	Beyond third-generation
BE	Best effort
BS	Base station
C/I	Carrier-to-interference ration
CAC	Call admission control
CBQ	Class-based queueing
CBR	Constant-bit-rate
CC	Congestion control
CC	Chase Combining
CDF	Cumulative distribution function
CDMA	Code-division multiple access
CIF-Q	Channel-Condition Independent Packet Fair Queueing
CPCH	Common Packet Channels
CPRMA	Centralized Packet Reservation Multiple Access
CQI	Channel Quality Indicator
CRC	Cyclic Redundancy checksum
CSDPS	Channel State Dependent Packet Scheduling
DA	Demand Assignment
Delay-EDD	Delay Earliest Due Date
DPDCH	Dedicated packet data channel
DPRMA	Dynamic Packet Reservation Multiple Access

DRR	Deficit Round Robin
EDF	Earliest Deadline First
ERR	Elastic Round Robin
FaSA	Fairness Family of Scheduling Algorithms
FCFS	First Come First Serve
FCSS	Fast Cell Site Selection
FDD	Frequency-division duplexed
FIFO	First-in-first-out
HARQ	Hybrid Automatic Repeat Request
HOQ	Head-of-queue
HSDPA	High Speed Downlink Packet Access
HS-DPCCH	High Speed Dedicated Physical Control Channel
HS-DSCH	High Speed Downlink Shared Channels
HS-PDSCH	High Speed Physical Downlink Shared Channel
IR	Incremental Redundancy
IWFQ	Idealized Wireless Fair Queuing
Jitter-EDD	Jitter Earliest Due Date
LB	Leaky bucket
LR	Latency Rate
MAC	Multiple Access Control
MAC-hs	High speed MAC
MIMO	Multiple input multiple output
MMS	Multimedia messaging service
NAK	Negative acknowledgement
ODRR	Opportunity-Based Deficit Round Robin
OFDM	Orthogonal Frequency Division Multiplexing
PBRR	Packet-Based Round Robin
PC	Power Control
PDU	Protocol Data Unit
PF	Proportional Fair
PGPS	Packet-based Generalized Processor Sharing
PRADOS	Prioritized Regulated Allocation Delay-Oriented Scheduling
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying

RFB	Relative Fairness Bound
RLC	Radio Link Control
RNC	Radio Network Controller
RRA	Random Reservation Access
RRM	Radio resource management
RT	Real-time
SCFQ	Self-Clocked Fair Queueing
SIR	Signal-to-interference ratio
SLA	Service Level Agreement
SRR	Surplus Round Robin
TBFQ	Token Bank Fair Queueing
TDD	Time-division duplex
TDMA	Time-division multiple access
TSN	Transmission Sequence Number
TTI	Transmission Time Interval
UBR	Unspecified-bit-rate
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
UTRA	UMTS terrestrial radio access
VBR	Variable-bit-rate
W-CDMA	Wideband CDMA
WF2Q	Worst-case Fair Weighted Fair Queueing
WFFQ	Wireless fluid fair queueing
WFQ	Weighted Fair Queuein
WLAN	Wireless local area network
WRR	Weight round robin
WT	Wireless terminal
WWW	World wide web

PREFACE

"UNLESS THE LORD BUILDS THE HOUSE, ITS BUILDERS LABOR IN VAIN;

UNLESS THE LORD WATCHES OVER THE CITY, THE WATCHMEN STAND GUARD IN VAIN."

PSALM 127:1

ACKNOWLEDGEMENT

This journey has been a long and lonely one. Many times I have contemplated the idea of giving up. If there was a dollar for every one of them, I would be very well off. I thank God for His grace and continuing spiritual guidance. My gratitude goes to my mother who encouraged me to take on this journey and greatest gratitude to my wife who shown me unconditional love and encouragement so that I could finish what I have started.

I would like to sincerely thank my advisor Dr. Victor Leung, for his time and effort during all these years. Without his help, I would not have finished this research. His patience during my absence for more than three years was very much appreciated.

I want to express my gratitude to the rest of the committee members in my thesis defense, Dr. Cyril Leung, Dr. Son Vuong, Dr. Hussein Alnuweiri, Dr. Resve Saleh, Dr. Alan Wagner, and Dr. Oliver Yang, for their time and valuable suggestions which made this thesis a quality one.

Special thanks go to the Communication Research Centre Canada who supported my research. Gratitude goes to Dr. Gerry Chan for his encouragement to finish this research and to Luc Boucher for his kindness and support. Without their encouragement and support, this research would not be possible.

DEDICATIONS

This thesis is dedicated to my lovely wife, Carole, for her love, encouragement and patience. I also like to dedicate this to my daughter, Madeleine, and all of my children in the future.

Chapter 1 INTRODUCTION

In the not-too-distant future, we expect to experience multimedia services integrated and provided through wireless networks. Today, multimedia services are available through the access to the Internet, and these services include World Wide Web browsing, electronic mail, file transfer, audio and video broadcast and conference, and gaming. True ubiquitous multimedia services can be realized through wireless networks. This gives rise to a tremendous demand on future broadband fixed and mobile networks, where not only integrated service provision must be provided, interoperability will also have to be seamless.

In response to this growing need of wireless/mobile systems diversification, much research focus is needed to efficiently integrate different wireless systems to enable interoperability and maximize wireless utilization for integrated services. Due to the delayed deployment of the third-generation (3G) wireless systems from its huge investment already made by operators and manufacturers worldwide, an opportunity for emerging license-free wireless Local Area Networks and personal area network (Bluetooth) technologies arises. Debate has since been on-going as to which technology will ultimately dominate and replace the other. However, as the situation is starting to stabilize, it is clear that these technologies will co-exist. Such vision is reflected by a research area conducted under the name of "Beyond 3G" (3G+) where breakthrough steps are being studied to integrate all diverse wireless technologies, along with new truly broadband wireless technologies, in the (r)evolutionary path toward the emerging 3G+ and future fourth-generation (4G) heterogeneous networking environment [1]. Various organizations [2], [3] are in the process of defining what future generation wireless networks (such as 4G) encompass and are planning to undertake extensive studies. At this point, most can agree that the vision of 3G+/4G wireless/mobile systems is the provision of broadband access, seamless global roaming, and Internet/data/voice everywhere, utilizing for each the most appropriate "always best connected" technology [1]. We can expect the future mobile broadband wireless networks to offer higher performance, by an order of magnitude or greater improvement in data rates and by more efficient quality-of-service (QoS) provisioning, over current 3G evolution proposals. Such network may comprise of different wireless air interface (3G cellular, 4G cellular, WLAN, Point-to-multipoint, ad-hoc peer-to-peer etc.), as illustrated in Figure 1.1.

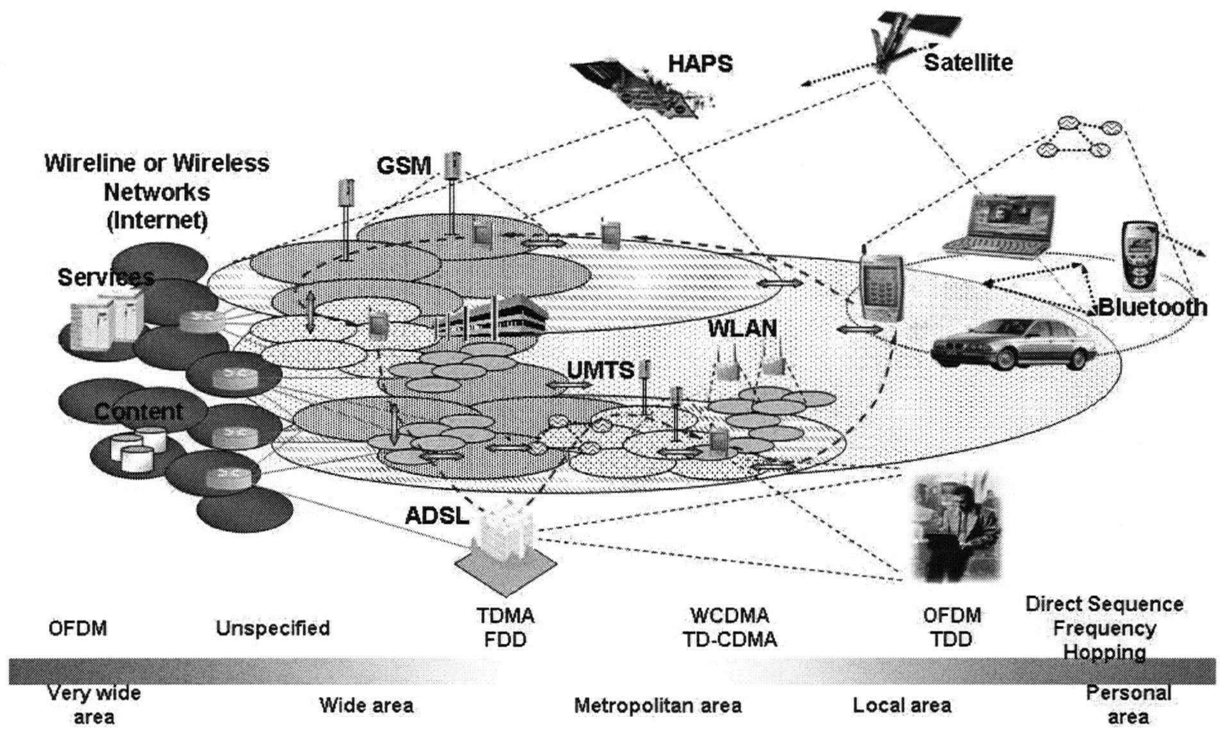

Figure 1.1 Characteristics of future mobile broadband wireless networks

Advances in enabling technologies such as advanced antenna processing, radio access, network infrastructure, or management platforms, as well as the mobile service provisioning environment are all integral parts of the evolution toward 4G systems. There are many challenges to overcome before the vision can be realized. These challenges include:

1. Evolution to a new and integrated wireless access networks that will accommodate different air interfaces as illustrated in Figure 1.1;
2. Link symmetry diversification – the ability to dynamically adapt to applications' needs in terms of uplink and downlink asymmetry;
3. Evolution to a new air interface that will accommodate broadband applications in the order of 100 Mbits/s.
4. Adoption of smart antenna technologies, such as the use of multiple input multiple output (MIMO) antenna processing techniques, which will help to improve spectral efficiency, especially when high date rates are demanded.
5. Multiple Access techniques that enable higher QoS provisioning functions to perform.
6. Evolution in Radio resource management architecture.

Though there are many challenges and each operator has their own vision and requirement in making sure their next generation network will be successful, there are two main factors for the success of 3G+/4G systems that most operators can agree, they are:

1. Cost effective provision of new services at the appropriate QoS levels
2. Efficient handling of versatile service area conditions (e.g., hot spots caused by time-variant traffic loads and mobility)

To address the QoS needs of services in future wireless systems, efficient resource management strategies are therefore critical to the success of these systems. Arguably, effective management of radio resources is not only essential in future wireless networks, but also the key factor in the efficient and affordable deployment and operation of these networks.

1.1 Radio resource management

From an engineering perspective, developers of wireless systems and networks struggle with different fundamental design problems. These problems include path loss, thermal noise, and the limited spectrum [4]. These give rise to scarce resources and therefore the need to provide high and efficient utilization of the system. The objectives of radio resource management (RRM) are high capacity and throughput for a given quality-of-service [5]. The tasks performed often include admission control, channel assignment, power control, and handoff [6].

Radio resource management in future high-performance packet-switched mobile broadband networks require much more features than that of the existing wireless networks. Such network (Figure 1.2) is expected to have RRM functions deployed at various points of the network. These functions can be categorized into the following four locations based on their respective RRM functionalities performed:

(a) at the point of entry into the integrated mobile access network (Network level RRM)

- Dynamic spectrum assignment
- Authentication and admission
- Negotiation of QoS
- Directed retry
- Mobility management attributes
- Policy management
- Traffic spoofing and/or shaping

(b) at the specific access system (Access system RRM)

- Local mobility (handoff)
- Soft handoff processing (optional)
- Network QoS to radio QoS mapping
- Location dependent decisions
- Interference avoidance

(c) at the access node (Access node RRM)

- Dynamic bandwidth management
- Fast scheduling
- QoS Routing four multihop
- Efficient broadcast/ multicast operation

(d) at the mobile (Mobile RRM)

- Handoff
- Diversity
- Multihop
- Peer-to-Peer

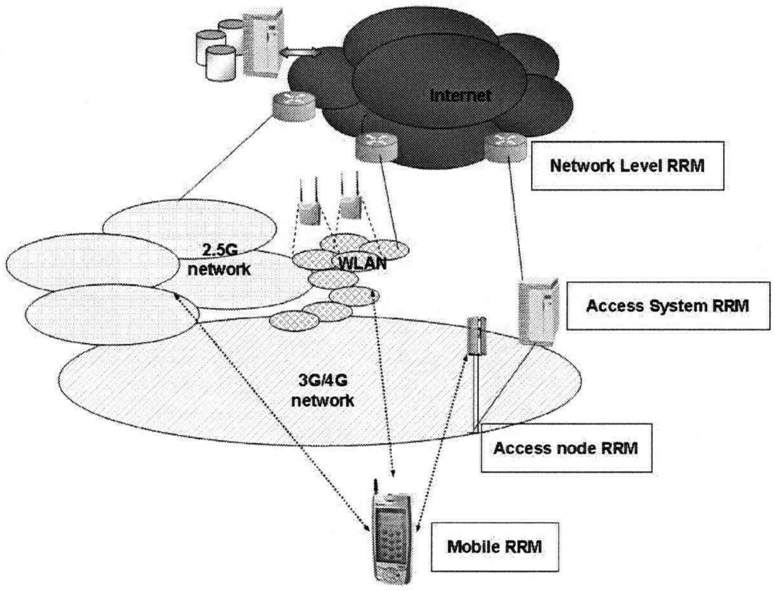

Figure 1.2 Radio resource management locations in future broadband mobile networks

When a mobile user begins to access the network, authentication and admission must be performed; it is conducted at a centralized network level RRM. Admission at this level is done to ensure the appropriate Service Level Agreement (SLA) is met which includes the negotiation of QoS. The admission process also includes the initial base station assignment, channel assignment, and bandwidth assignment. If a base station cannot provide service, the call request is re-directed towards another base station (if within range) different from the one providing best link budget. Once admitted, packet data transfer can proceed, which includes power control, base station re-assignment, re-assignment of the number of base stations, channel re-assignment, bandwidth re-assignment, and of course call release. Bandwidth assignment is a process of choosing the channel to be allocated to the user; in the case of packet-switched systems, the assignment would involve the number of resource units (virtual channel).

Admission control is the process of determining whether a service request can be admitted to the system. A connection could be rejected for reasons related to capacity, or interference levels. Channel assignment is the assignment of a predefined sub-set of channels to the users. This is dealt mainly by the medium access control protocols (MAC) which will be explained below. Power control (PC) is simply the process of setting the transmission power level and can be categorized into fast PC and slow PC. Handoff is the process of changing serving base and/or channel, which is sometimes distinguished between hard- and soft-handoff. Hard handoff means the change of one base station at a time and the process is not seamless; it can have a serious impact on the signaling channels. Soft handoff requires access to more than one base station at a time. This gives the elusion of seamless service through the use of macro diversity, but can have a serious impact on the network capacity.

In the interest of providing QoS to future generation mobile networks that can interoperate, this research is focused on the bottleneck between the access node and the mobile. Mitigating this bottleneck is what many MAC protocols is designed to do. As wireless network access becomes more sophisticated, packet scheduling becomes more significant as demand for maximizing utilization of scarce wireless resource increases. It is desirable to have a fast scheduling algorithm that is not dependant on specific MAC schemes. In the sections below, the functions of MAC and the higher functional blocks are clarified and elaborated, as it will help us understand the significance of decoupling these functions later.

1.1.1 Medium access control

The MAC protocols play a crucial role in QoS guarantees especially in future generation wireless/mobile packet-switched networks. Specifically, the scheduling algorithm in MAC protocols is the key element that enables a MAC to provide QoS guarantees as future applications increasingly demand more QoS support. Wireless MAC protocols have been studied extensively since the 1970s [7]. MAC protocols can be broadly classified into two categories – distributed and centralized, according to the type of network architecture for which they are designed. Protocols can be further classified based on the mode of operation into random access protocols, guaranteed access protocols, and hybrid access protocols as shown in Figure 1.3. In a random access protocol, users contend for access to the medium. Collision results when multiple users make a transmission attempt at the same time. Users resolved that collisions in an orderly manner according to rules defined by the contention resolution algorithm. In a guaranteed access protocol, user access is determined by a scheduling algorithm – usually polled in a round-robin fashion or by token-passing. In terms of suitability to multimedia services, hybrid access protocols are preferred over guaranteed and random access. Most hybrid access protocols are based on request-grant mechanisms. Each user sends a request to the base station indicating how much bandwidth is required to send the data that currently resides in its buffer. The request is sent using a random access protocol. The base station then allocates (based on the scheduling algorithm) uplink resources for the actual data transmission and broadcast the grant decision to all the users. Depending on the intelligence at the base station, the hybrid access protocols can be further classified into Random Reservation Access (RRA) protocols and Demand Assignment (DA) protocols. In an RRA protocol, the BS has implicit rules for reserving upstream bandwidth. On the other hand, in a DA protocol, the BS controls upstream data transmissions according to their QoS requirements.

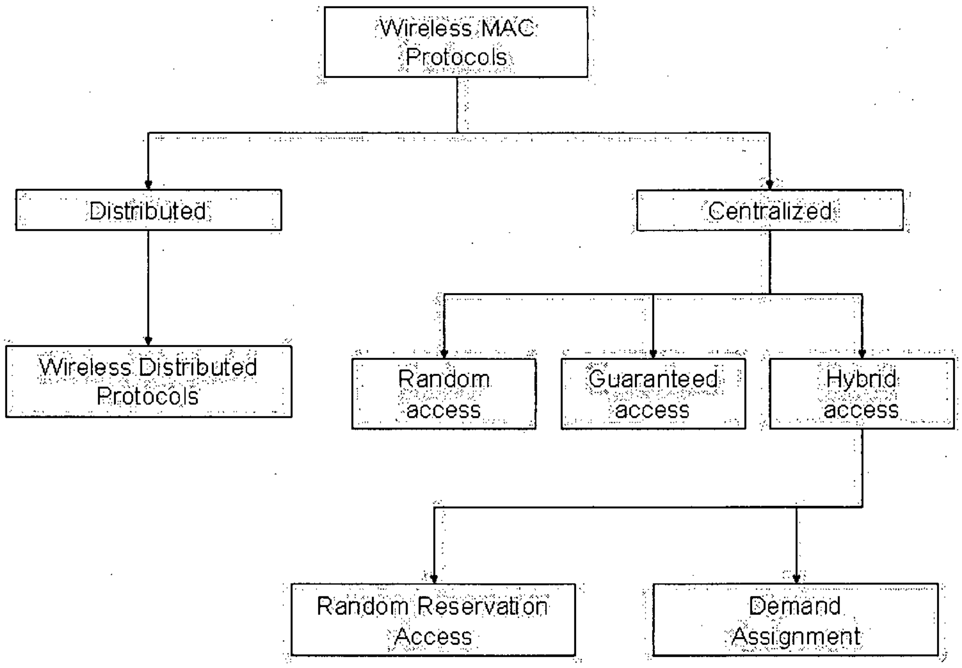

Figure 1.3 Classification of wireless MAC protocols

Both channel access and resource reservation/allocation affects QoS performance. Resource reservation schemes can be divided into *dedicated assignment, random access, and demand-based* *assignment.* Dedicated assignment is the commonly used scheme in first and second generation systems where voice applications dominated. This scheme is not suitable for bursty traffic type, as resources are not utilized fully. The random access scheme would be more suitable for bursty traffic type, as users would be able to send data whenever data arrive but only after successfully competed for channel contention. Due to contention delay, it is not desirable to serve time sensitive traffic using this scheme. The demand-based assignment scheme finds a happy medium by accepting user request before allocating the necessary resources.

Channel access contention is an undesirable effect in applications where QoS requirements, such as delay, are critical. The other function of MAC is to moderate access to the shared medium by defining rules that allow devices to communicate with each other in an orderly fashion with efficient and fair sharing of the scarce wireless bandwidth. In [23], a comprehensive survey on MAC protocols for multimedia traffic in wireless networks is given. Although the survey targets primarily third-generation systems that utilize both time-division multiple access (TDMA) and code-division multiple access (CDMA) schemes, it is important that we understand these schemes and appreciate the significant contribution this research has with respect to providing fast scheduling for heterogeneous access networks.

Both Dynamic Packet Reservation Multiple Access (DPRMA) and Centralized Packet Reservation Multiple Access (CPRMA) are designed for frequency-division duplexed (FDD). They are based on the classical PRMA protocol [25] and use demand-based reservation scheme to serve multimedia services. Demands are submitted using slotted ALOHA for reservations contention periods and slots are dynamically assigned. The results of the contention period are transmitted downlink through reservation acknowledgement bits. Successful users monitor slot reservation through the downlink channel and send data according to the slots assigned. Notable differences of CPRMA include its ability to promptly transmit corrupted packet and to make complex scheduling decision at the base station (BS).

For Internet services, asymmetric usage of the uplink and downlink channels is more appropriate. Time-division duplex (TDD) scheme has the advantage to satisfy this need. Many wireless ATM systems [25]- [30], developed in the late to mid 90's for wireless broadband multimedia communication, make use of dynamic TDMA/TDD frame structure. The protocols often use centralized access where a particular base station controls the resource utilization. TDMA was used because of the high bandwidth requirement, and TDD was adopted to increase the multiplexing gain. Several time slots (mini-slots) are bundled to frames of fixed or variable length. In [29], transmission requests are made to the base station in the dedicated reservation slots using slotted ALOHA random access. Scheduling decision is made at the base station based on the QoS parameters of user traffic and broadcast in the downlink channel. This MAC protocol combines all three resource sharing methods – dedicated, random, and demand assignment. Fixed dedicated allocation is assigned to constant-bit-rate (CBR) traffic. For variable-bit-rate

(VBR) traffic, combination of fixed allocation and the sharing of unused slots were used. In the case of available-bit-rate (ABR) and unspecified-bit-rate (UBR) traffic, such as non-time-critical data and applications like file transfer and e-mail, dynamic allocation is used, whereby any unused slots left over from CBR and VBR traffic can be used. This MAC protocol was quite ingenious and it was believed that the MAC protocol alone was sufficient to satisfy the QoS needs of multimedia traffic, and little emphasis was put on the scheduling aspect. As system loading increases, the appropriate use of scheduling algorithms can increase bandwidth utilization.

The MASCARA protocol [30] on the other hand, placed more emphasis on the scheduling scheme. The scheduling algorithm used is called Prioritized Regulated Allocation Delay-Oriented Scheduling (PRADOS). It assigns priorities for each connection according to its service class. The algorithm combines priorities with a leaky bucket traffic regulator that uses token pool. Tokens are generated at a fixed rate equal to the mean ATM cell rate of each connection. The size of the pool is equal to the maximum number of ATM cells that can be transmitted with the rate greater than the declared mean. One token is consumed when an ATM cell is scheduled for transmission. Another major difference between MASCARA and others is the variable boundaries between the uplink, downlink, and contention period. Due to the complexity of this MAC protocol, it requires a more sophisticated scheduling algorithm.

Wideband CDMA, a pure CDMA standard, is one of the two 3G standards in the world right now and is adopted by the Japanese and Europeans. As pointed out in [23], the WCDMA protocols deal with a complex and very flexible physical layer where many transmission options are possible. A dual mode packet transmission scheme is envisaged within the W-CDMA concept defined in the UMTS terrestrial radio access (UTRA): Common and Dedicated transmission channels. The common channel packet transmission on RACH is typically used for transmission of short infrequent packets, while dedicated channel with closed loop power control is used for large packet transmissions. UTRA specifies packet transmission on Common Packet Channels (CPCH). Transmission on CPCH has two phases of contention: random access phase and contention resolution phase. Although closed power control is allowed on CPCH, a restriction on maximum duration of transmission is needed. The procedure of CPCH access is described in [30]. For large and frequent data packet, transmission on the dedicated channel is considered (DPDCH). The control is performed through a demand assignment protocol to guarantee service multiplexing with QoS requirements. The stability of the protocol depends on higher-layer functions such as call admission control (CAC) and congestion control (CC). Due to the undefined scheduling algorithm in the protocol, higher functions such as the CAC would not be able to evaluate whether a new user could be admitted.

Some of the higher functions in radio resource management, such as scheduling and admission control are inter-related. Figure 1.4 gives a generic QoS management framework of an access node, for example, at the Radio Network Controller in UMSS systems.

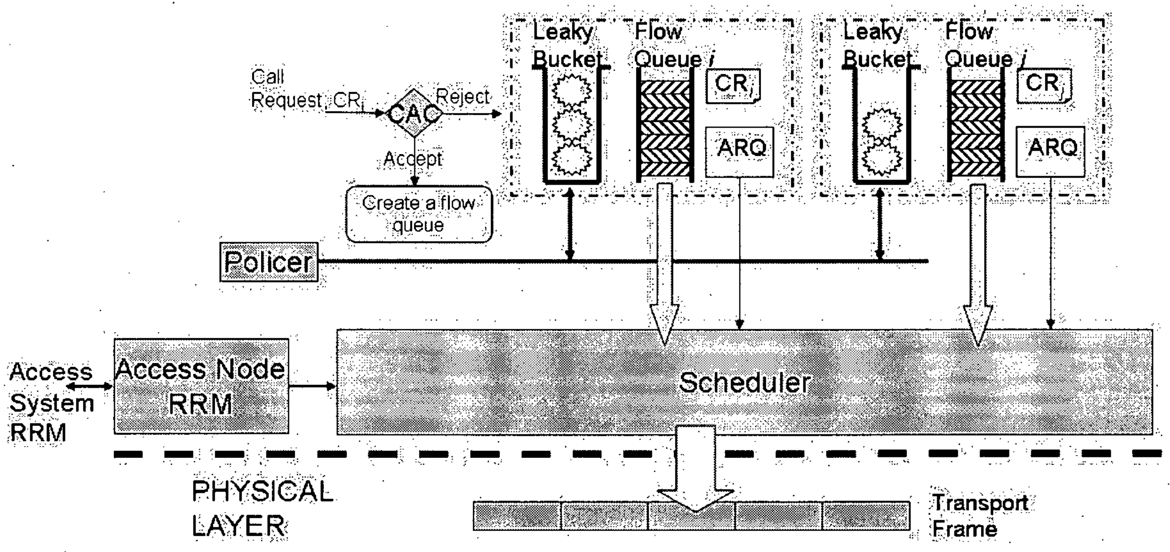

Figure 1.4 Functions of radio resource management for QoS provisioning

1.1.2 Scheduling algorithms

Recently we have begun to see growing importance of scheduling algorithms [8] in MAC protocols, as more sophisticated multimedia applications are being developed and are demanding QoS from their wireless/mobile networks.

A MAC protocol enables scheduling functions to make decisions in terms of QoS provisioning. The function of scheduling algorithms is to decide on the amount of bandwidth allocation to all traffic while providing QoS according to the various loss, delay and bandwidth requirements. Although the two functions are often integrated to provide QoS as we saw in the previous section, it is important to understand the difference between their roles when designing a QoS provision scheme. If a scheduling algorithm is designed for a specific MAC protocol, it is unlikely that it can be exported easily to another MAC system and be served efficiently, let alone interoperable with one another. Therefore, in heterogeneous wireless systems different scheduling algorithms would be needed for different MAC protocols. Different wireless systems often come with its specific MAC protocol [9]-[25], the challenge is to maintain (as much as possible) seamless QoS across various systems. Under the vision of 3G+ (or 4G in particular), once a roaming terminal gains access to a foreign system within a heterogeneous environment, the same QoS should be maintained as much as possible. It is therefore desirable to have one bandwidth allocation algorithm that can easily interoperate with various MAC protocols. Such bandwidth allocation algorithm should also be capable of interoperating with other schedulers within the wireline network.

Under our framework, we envision the decoupling between MAC and scheduling algorithm and focus on the design of a scheduling algorithm that anticipates compatibility and interoperability with other (non-specific) MAC protocols in future generation access systems. One benefit of this framework is that QoS can be provided seamlessly across different wireless systems. The overview of scheduling algorithms is given in Chapter 2.

1.1.3 Call admission control

Consider for a moment a simple uncontrolled TDMA network where the number of channels per cell is 20, and the network consists of 3 cells (Figure 1.5). At time t_0, the number of users, $N_i(t_0)$ = (18, 17, 19), for i=1,2,3. At time t_1, a new user arrived at cell 3, then $N_i(t_1)$ = (18, 17, 20). At time t_2, a user from cell 2 is leaving cell 2 and is heading to cell 3. The user has to be handed over to cell 3, but cell 3 is fully loaded. So the user is rejected by cell 3 and will be dropped. Clearly, admitting the new user in cell 2 at instant t_1 was not a good decision.

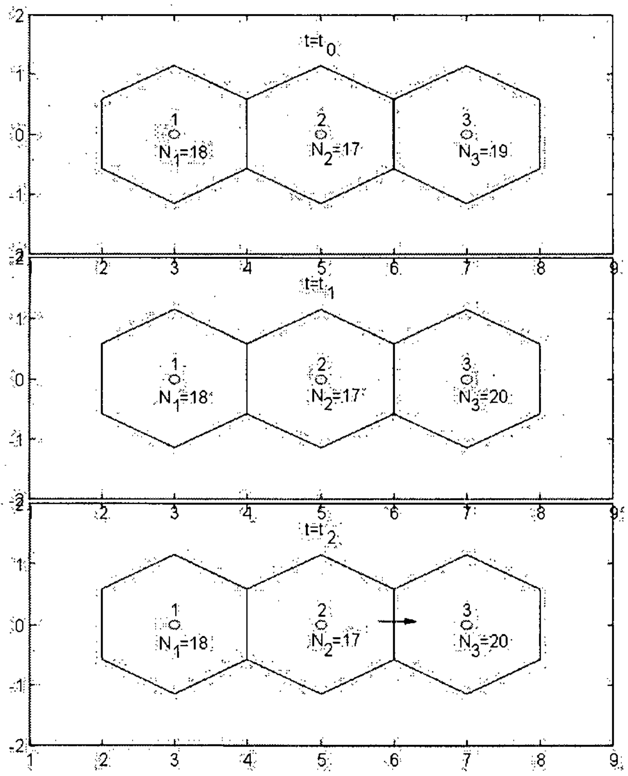

Figure 1.5 Demonstration of a call admission problem.

Mathematically, CAC can be modeled as MAX(f_1) or MIN(f_2), subject to criteria such as $g_1<h_1$, $g_2<h_2$,..., $g_m<h_m$, where h_i is a threshold and f_1 and f_2 are functions to be maximized or minimized. In the example given, the CAC can be modeled as MAX(Users) or MIN(P_b blocking probability), subject to $P_b<h_2$. Sometimes, the optimization part is not explicitly specified. In which case, it is always said that CAC is used to guarantee the QoS. The parameters that are often optimized include:

- Guarantee the signal quality (*SIR*, *Eb/Io*, or *BER*)
- Guarantee the call dropping probability (*Pd*)
- Giving priority to some classes
- Maximize revenue
- Fair resource sharing (Complete Partitioning, Complete Sharing, and threshold)
- Guarantee transmission rate (e.g., *throughput*)
- Guarantee packet-level QoS

Call admission control schemes have been investigated extensively in traditional circuit-switching wireless networks as an important mechanism to support connection-level QoS [32][33][34][35][36][37] where the handoff and mobility management issues are addressed. The connection-level QoS performance parameters are often the call dropping and call blocking probability. At the same time, CAC schemes have also been investigated on packet-level QoS [38][39], such as delay and throughput. Other researchers addressed both connection-level and packet-level QoS for wireless networks [40][41][42].

- Whether the scheme is call-level or packet level, the admission criterion often include:
- Resource availability – in terms of effective bandwidth [43][44]
- Equivalent number of users – relative to voice service users [45][46]
- Signal-to-interference ratio (SIR) or residual capacity [46][47][48]
- Transmitter or receiver power [49][50]
- Dropping probability [51][52]
- Bandwidth utility function [53]
- Degraded area service [54]

Although, CAC is a research topic in itself and is beyond the scope of this research, it is important to understand its relationship to scheduling algorithm performance because our investigation requires a deployment of such. Furthermore, we have developed an effective CAC algorithm to work with our scheduling algorithm.

1.2 Heterogeneous services

The main goal of wireless communication is to allow the user access to the capabilities of the global packet-switched network at any time without regard to location or mobility. Cellular concept is quite suitable and will very likely be the basis of future wireless networks. Users are interested in receiving telecommunication services or heterogeneous information services (video, voice, and data) in the user-perceived sense. Mapping perceived performance on technical parameters in the wireless system is a complex task. There are many factors influencing the user experience, including not only the shortcomings of the wireless system but also factors such as the wireline backbone switch performance, the service providers application software and hardware, and the user interface provided by the service providers and the terminal manufacturers. To rationalize the design of telecommunications systems, 3GPP (UMTS) [55] divided the services provided (also known as *bearer service*) into four classes as outlined in Table 1.1 [4].

Table 1.1 3G (UMTS) bearer service classes

Service Class	Typical Applications	Service Functional Characteristics	Relevant QoS Requirement
Conversational Real Time (RT)	Voice over IP Video conferencing	Preserve time relations between entities Stringent preservation of conversational patterns (low delay)	Low jitter Low delay
Streaming RT	Video/Audio streams	Preserve time relations between entities	Low jitter
Interactive Best Effort (BE)	Web browsing	Request-response pattern Preserve payload (low error rate)	Round trip delay time Low BER
Background BE	File transfer E-mail	Not time critical Preserve payload (low error rate)	Low BER

These classes were categorized based mainly on the delay requirements ranging from very strict delay requirements (e.g. voice services) to the very relaxed requirements in the background best-effort class. Specifically, the services in these classes are characterized by some of the following QoS parameters:

- Maximum data rate
- Guaranteed data rate;
- Maximum packet/message size;
- Residual bit error rate – the undetected error rate after the delivery of the information over the service interface;

- Transfer delay – the time delay the packet spends between the service access points; This could be a guaranteed delay for every message (e.g., in conversational class services) or defined in statistical terms, for example, average delay or X percentile delay;
- Priority – indicates the relative importance of different messages.

There are just too many parameters that can be used to qualify various services, and realistically most systems can take only finite set of bearer services, where each parameter will be allowed to take one (out of a few) discrete values. Table 1.2 [4] provides an indication of what ranges these service parameters can take in a 3G wireless system.

Table 1.2 Some 3G (UMTS) service attribute/parameter ranges

Traffic Class	Conversational	Streaming	Interactive	Background
Max bit rate (kbps)	<2000	<2000	<2000 – overhead	<2000 – overhead
Max SDU size (byte)	<1500	<1500	<1500	<1500
Guaranteed bit rate	<2000	<2000		
Transfer delay (ms)	80 – max value	500 – max value		
Priority	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3
Residual BER	$5*10^{-2}$, 10^{-2}, 10^{-3}, 10^{-6}	$5*10^{-2}$, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}	$4*10^{-3}$, 10^{-5}, $6*10^{-8}$	$4*10^{-3}$, 10^{-5}, $6*10^{-8}$

1.3 Thesis objectives and contributions

The function of MAC is to moderate access to the shared medium by defining rules that allow these devices to communicate with each other in an orderly, efficient and fair sharing of the scarce wireless bandwidth. The function of scheduling algorithms is to determine the amount of bandwidth allocation to all traffic while providing QoS according to the various loss, delay and bandwidth requirements. Although the two functions are often integrated to provide QoS, it is important to understand the difference between their roles when designing a QoS provision scheme. If a scheduling algorithm is designed for a specific MAC protocol, it is unlikely that it can be easily exported to another MAC system and works efficiently. Therefore, different scheduling algorithms would be needed for different MAC protocols. As we can see earlier, different wireless systems often come with specific MAC protocols, but when a mobile terminal roams heterogeneous networks, seamless QoS should be maintained as much as possible. It is therefore desirable to have one bandwidth allocation algorithm that could easily be deployed and interoperate with various MAC protocols. Such bandwidth allocation algorithm should also be capable of interoperating with other schedulers within the wireline network. Under our framework, we envision the decoupling of MAC and scheduling algorithm to focus on the design of a scheduling algorithm that anticipates interoperability with other MAC protocols in the future. One benefit of this framework is that QoS can be provided seamlessly across different wireless systems.

This dissertation focuses on the design of a novel packet scheduling algorithm – Token Bank Fair Queuing (TBFQ) – for future generation wireless systems with heterogeneous services. The analysis of such algorithm include both theoretical and simulation. The theoretical analysis opens other possibilities of fairness algorithms which belong to the same family of scheduling algorithms as the TBFQ. The objective is to provide communication engineers the tools to utilize TBFQ. Ideally, we should provide single closed-form formula to characterize the performance of TBFQ; however, it is not possible. Instead, we treat it as a class of queueing network and utilize numerical techniques in characterizing its performance such as buffer occupancy which would allow us to determine the mean packet delay. In addition to the theoretical analysis, simulation analysis is used to characterize TBFQ performance in wireless systems. The combination of theoretical and simulation analysis allows us to take comfort in cross-referencing the validity of both. Contributions of this thesis to the research community are as follows:

- Proposed a novel, priority-based scheduling algorithm - the TBFQ - which is shown to be effective in the wireless applications and is suitable for heterogeneous wireless systems;
- Proposed the separation of scheduling algorithm from MAC which results in a framework that would be conducive for interoperability and seamless QoS in heterogeneous systems;
- Presented a novel recursive analysis technique that gives tighter bound than conventional analysis methods such as network calculus. This new methodology opens new possibilities to analyze a broad range of priority-based scheduling algorithms;
- Analyzed the theoretical performance bound using On-Off voice traffic sources. This opens up opportunities to obtain bounds for other type of on-off traffic;
- Introduced two modifications and enhancements of the TBFQ algorithm;
- Adoption of the GTH computational technique in the analysis of queues with very large number of states;
- Analyzed the TBFQ algorithm through simulation in both TDMA and CDMA systems.

- Developed a suitable call admission control algorithm that works in conjunction with the TBFQ.

1.4 Organization of the thesis

Chapter 2 provides an overview of scheduling algorithms and their current developments. As it will be pointed out, some of the algorithms would not be applicable to wireless access networks. The traffic models for various services are also introduced as they are important in the analysis of the performance of the algorithm developed.

Chapter 3 proposes the token bank fair queuing algorithm (TBFQ) and presents the features and characteristics of the algorithm. Fairness analysis of the algorithm is explored, as well as the minimum service guarantee and graceful service capabilities.

Chapter 4 further analyzes the input queue distribution of the TBFQ algorithm using the combination of stochastic and recursive algorithm. We give account of the methodologies that are available to analyze the algorithm and make use of these tools to find a performance bound in the queue distribution which would allow us to determine the mean delays.

Chapter 5 applies the TBFQ in a TDMA TDD-based network and presents detail performance analysis of TBFQ in such network access under location-dependent channel error condition. Heterogeneous services are considered in the simulation and comparisons are made to other commonly used algorithms. Performance parameters such as fairness, isolation, delay, jitter, throughput, and utilization are also be explored.

Chapter 6 applies the TBFQ in a next generation wideband CDMA 3G system (WCDMA) called High Speed Downlink Packet Access. It is demonstrated that the TBFQ also performs quite well in CDMA systems. To enhance its performance and utilize the features offered by such systems, modifications of TBFQ are introduced and their performance is demonstrated.

Finally, Chapter 7 concludes the thesis and outlines possible future research topics and directions.

Chapter 2 OVERVIEW OF SCHEDULING ALGORITHMS

2.1 Wired network algorithms

The GPS scheduler, in the scheduling of bandwidth over a link, is an unimplementable but ideally fair scheduler that distribute bandwidths among the various flows [57][58]. During each infinitesimal interval of time, the GPS scheduler visits each backlogged flow once and schedules an equal and infinitesimal amount of data for transmission over the output link. If different weights are imposed on network flows, the amount of data transmitted from each flow by the GPS scheduler is proportional to its weight during each infinitesimal interval of time. By this means, the GPS scheduler achieves max-min fairness. The *Max-Min fair share* policy [58] is a classic notation of fairness in the allocation of a resource among multiple requesting entities with equal rights to the resource but unequal demands.

It is apparent that the GPS scheduler is an ideal policy and cannot be implemented in real systems, where network traffic is packetized and flow packets have different sizes. Many practical scheduling algorithms have been proposed to approximate the ideal GPS scheduler, such as Weighted Fair Queueing (WFQ) [59] or sometimes referred to as Packet-based GPS (PGPS), Self-Clocked Fair Queueing (SCFQ) [60], Worst-case Fair Weighted Fair Queueing (WF2Q) [61], Deficit Round Robin (DRR) [62]and Elastic Round Robin (ERR) [63]. In this section we will review some of these scheduling algorithms and outline their properties, as some of them will be used from performance comparison standpoint.

2.1.1 First Come First Serve (FCFS)

FCFS is one of the most rudimentary queuing algorithms that can be employed in switches. As the name implies, in FCFS, the order of arrival completely determines the bandwidth allocation. FCFS service is trivial to implement, requiring a router or a switch to store only a single head and tail pointer per output link. However, FCFS fails to provide adequate protection from a bursty source that may suddenly send packets at a rate higher than its fair share for brief periods of time. It is easy to see that this rogue flow will capture an arbitrary part of the outgoing bandwidth. Also such a source can significantly increase the upper bound on the queuing delay of packets belonging to a flow from another source. Fairness, however, requires that as long as a source is demanding bandwidth within its rightful share, the delay experienced by packets from this source should not be affected by other traffic in the network.

2.1.2 Round Robin Service Policies

Consider several flows, with packets waiting in the respective queues to be forwarded to another queue or an output link in a round robin fashion. Many implementation techniques for round-robin policy are possible.

2.1.3 Packet-Based Round Robin (PBRR)

Packet-Based Round Robin (PBRR) [64]: the scheduler visits each of the queues in a round-robin fashion, and transmits an entire packet from a queue before beginning transmission from another queue. The PBRR scheduling discipline ignores the packet lengths and would be fair if the average packet size over the interval of a connection were the same for all the traffic flows, in which case each flow would get an equal share of the outbound bandwidth. It is, however, not fair among the flows when the packet sizes in the different flows are not equal. Consequently, flows sending longer packets use up an unfairly high fraction of the available transmission bandwidth. In the worst case, a flow can get *Max/Min* times the bandwidth of another flow, where *Max* is the maximum size of the packet and *Min* is the minimum packet size [58].

2.1.4 Deficit Round Robin (DRR)

Deficit Round Robin (DRR) [62], a less fair but more efficient scheduling discipline with an $O(1)$ per packet work complexity, was proposed by Shreedhar and Varghese in 1996. DRR is not a timestamp-based algorithm, and therefore, avoids the associated computational complexity. DRR achieves $O(1)$ time-complexity because it serves the active flows in a strict round robin order. It succeeds in eliminating the unfairness due to different packet lengths observed in pure PBRR. This is done by keeping a state, associated with each queue called a deficit count (DC) to measure the past unfairness. A *Quantum* is assigned to each of the queues and when a flow is picked for service, its DC is incremented by the quantum value for that flow. A packet is served from a queue only if the packet size at the head is less or equal to the sum of the *quantum* and the deficit counter value; otherwise, the scheduler begins serving the next flow in the round robin sequence. When a packet is transmitted, the DC corresponding to that flow is decremented by the size of the transmitted packet. In DRR, in order that the per-packet work complexity is $O(1)$, one has to make sure that the *quantum* value chosen is no smaller than the size of the largest packet that may potentially arrive at the scheduler. Otherwise the per-packet work complexity increases to $O(\mathrm{n})$ since one may encounter a situation, in which, even after visiting each of the n flows and examining the respective DC values, no packet is eligible for transmission. A per-packet work complexity of $O(1)$ is ensured if we make sure that at least one packet is transmitted from each active flow during each round. This is ensured if the *quantum* is no smaller than the size of the largest possible packet, since this guarantees that the packet size at the head of each queue at the start of its service opportunity will always be less than the sum of the DC value and the quantum value of the flow. In order to achieve a per-packet work-complexity of $O(1)$, therefore, the DRR scheduler requires knowledge of the upper bound on the size of a packet. DRR, thus, is not ideally suitable for IP networks (especially bursty applications) since it requires the knowledge of the size of a packet before making a decision on transmitting it, and in addition requires an upper bound on the size of a packet.

2.1.5 Surplus Round Robin (SRR)

In [65][66], a fair scheduler similar to DRR was proposed. This algorithm, later known as Surplus Round Robin (SRR), has also been used in other contexts such as in [71]. SRR is a modified version of DRR, in which the scheduler continues serving a flow as long as the DC value of the flow is positive. When the DC becomes negative, the scheduler begins serving the next flow in the round robin sequence. Thus, while DRR never allows a flow to overdraw its account but rewards an under-served flow in the next round, SRR allows a flow to overdraw its account but penalizes the flow accordingly in the next round. DRR keeps an account of each flow's deficit in service, while SRR keeps an account of the surplus service received by each flow. SRR does not require the scheduler to know the length of a packet before scheduling it. However, it does require the use of a fixed *quantum* assigned to each flow per round. As in DRR, in order to ensure an $O(1)$ per-packet work complexity, the *quantum* value has to be no smaller than the size of the largest packet that may potentially arrive at the scheduler. SRR, like DRR, does not work well in networks requires knowledge of the upper bound on packet sizes.

2.1.6 Elastic Round Robin (ERR)

All the above schedulers require the knowledge of the maximum packet length to ensure a low computation complexity. The Elastic Round Robin (ERR) [63] uses a method which does not need the maximum packet length to achieve fairness with low complexity. Each flow is assigned an allowance, measured in terms of the number of flits that a flow can transmit in any given round. A flit as the smallest piece of a packet that can be independently scheduled, and the length of a packet is measured in terms of flits. The queues are serviced in round robin order. When a queue is selected for service, ERR scheduler calculates its *allowance*. ERR serves a queue as long as the number of flits transmitted from that queue in any round is less than its allowance in that round. The ERR scheduler, however, allows a flow to exceed its allowance. Associated with each queue is a state, *surplus count* which keeps track of the extra bandwidth that a flow used in any given round. The *surplus count* is used to calculate the allowance for a flow in the following round, so that if a flow overdraws its allowance by some amount, it is penalized by this amount in the next round. A flow that tries to seize a large fraction of bandwidth beyond its fair share will have a large value of surplus count. In the next round, the ERR scheduler allows the other flows to transmit at least as many flits as this flow did in addition to its fair share in the previous round. Thus, the flows that received little service in a round are compensated for in the next round.

2.1.7 Packetized GPS (PGPS) or Weighted Fair Queueing (WFQ)

The Weighted Fair Queueing (WFQ) policy approximates the GPS scheduler in the sense that WFQ tends to serve packets in the order of their finishing time under GPS. The system maintains a variable referred as *round number*, indicating the time under GPS by increasing inversely as the number of active flows. Each packet upon arrival is associated with a tag referred as *finish number*. Assuming the *k*-th packet of flow i, p_i^k, arrives at time τ, its finish number $F(p_i^k)$ is computed as

$$F(p_i^k) = \max\{F(p_i^{k-1}), RN(\tau)\} + \frac{L(p_i^k)}{w_i} \tag{2.1}$$

where $RN(\tau)$ is the round number at time τ, $L(p_i^k)$ is the size of packet p_i^k, and w_i is the weight of flow *i*. In other words, the finish number of packet p_i^k depends on the round number at its arrival time if the previous packet from flow i, p_i^{k-1}, has finished service when p_i^k arrives, or on the finish number of p_i^{k-1} otherwise. When a packet finishes service, the WFQ scheduler selects the packet with the smallest finish number in the system as the packet to be served next. PGPS [59] has been proven to provide upper delay bounds for the flows given they obey the leaky bucket model. Since PGPS is work-conserving and packet-based, a packet that has departed from the GPS system earlier than other packets may not yet have arrived when PGPS switches to this flow. Therefore WFQ (PGPS) has to switch to the next non-empty flow whose packet would depart after or at the same time as the one that has not yet arrived. This makes a WFQ system fall behind the GPS system by no more than the maximum packet transmission time. Let B(t1,t2) be a set of backlogged flows within time interval (t_1,t_2), the ith flow then gets the output link share $\varphi_i^* = \frac{\varphi_i}{\sum_{j\in B(t_1,t_2)} \varphi_j}$. Although the scheduler guarantees that a packet is delayed by no longer than the longest packet transfer time compared to the GPS system, in practice, a packet in a WFQ system can depart much sooner than in the reference one. Consider a GPS system backlogged for a time period τ. Then, if shares of the flows are such that a number of packets of a train from flow i depart from the system before or at the same time with packets belonging to the other flows, the respective WFQ system will serve this train ahead of these packets. Assume that these other packets departed the from GPS system at the same time. The WFQ system will serve them in an order from the first to the last one. Thus, although in the GPS system all the packets are mixed exactly according to their link shares at any time instance, in the WFQ, all the packets except one will be served at least one packet length earlier than in the reference system. Figure 2.1(b) and Figure 2.1(c) show an example of such a case for equally sized packets, where one flow has share 0.5 and four flows have equal shares of 0.125 of the normalized link capacity. The arrival pattern of the flows is shown on Figure 2.1(a).

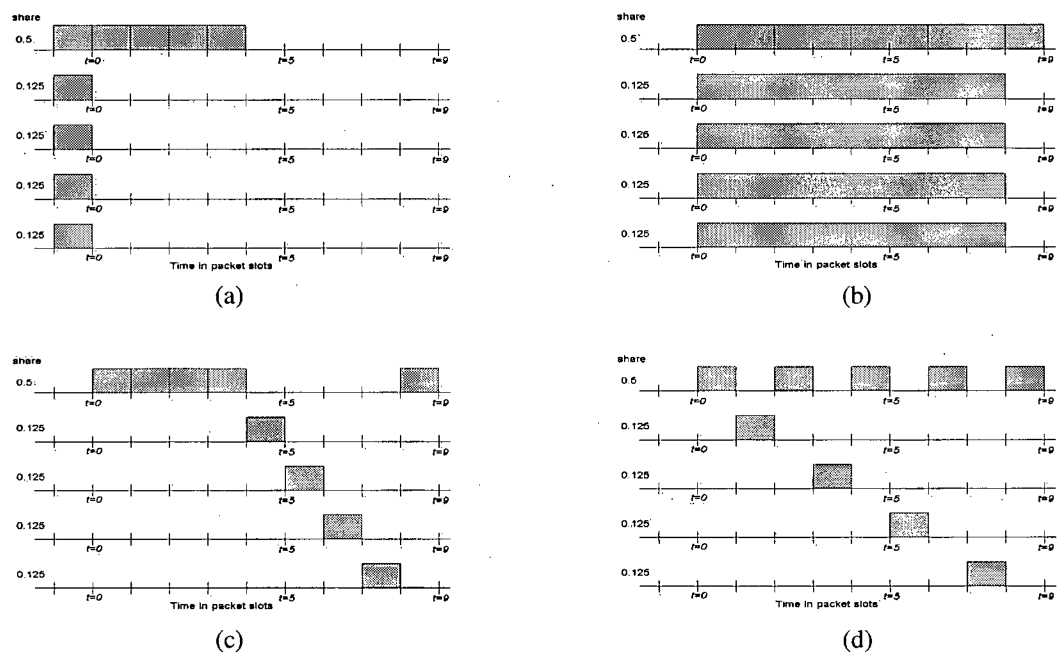

Figure 2.1 (a) The arrival pattern; (b) GPS servicing order; (c) The WFQ servicing order); (d) WF^{2}Q servicing

2.1.8 Worst-case Fair Weighted Fair Queueing (WF2Q)

Unlike WFQ, WF^{2}Q [61] chooses the packets eligible for transmission at time τ based not only on their service finishing times in the GPS system but also on their service starting times. That is, if a packet has started being serviced at time τ in the GPS system, then it is allowed to be served at this time in the WF2Q system. If there is more than one such a packet, it is ordered, just like in WFQ, in increasing order of their departure times in the GPS system. Thus, the arrival pattern used for the WFQ example will be served by the respective WF^{2}Q system as shown in Figure 2.1(c).

2.1.9 Self-Clocked Fair Queueing (SCFQ)

Self-Clocked Fair Queueing (SCFQ) relaxes the WFQ scheduling policy by not strictly emulating the GPS system and not accurately maintaining the round number [7]. Instead, the finish number of the packet currently under service is used as the current round number, i.e.,

$$F(p_i^k) = \max\{F(p_i^{k-1}), CF(\tau)\} + \frac{L(p_i^k)}{w_i} \tag{2.2}$$

where $CF(\tau)$ is the finish number of the packet being served at time τ. SCFQ has a finite relative fairness as follows:

$$RFB^{SCFQ} = 2\frac{L}{w_m} \tag{2.3}$$

2.1.10 Earliest Deadline First (EDF)

The Earliest Deadline First (EDF) scheduling policy, also called Earliest Due Date (EDD), schedules packets in the order of their deadlines. The deadline of a packet is defined to be the arrival time of the packet plus the delay bound assigned to the packet. When a packet arrives at a node, it is inserted into a sorted priority queue based on this deadline. For the single node case, non-preemptive EDF is known to be optimal under the restriction that maximum packet transmission time is identical for all flows [67] [68]. This is consistent with the known optimality of EDF as a real-time scheduling algorithm [69]. Under the restriction that all flows have a deadline equal to their packet inter-arrival time, the flows will be schedulable as long as the link utilization is less than 100%. EDF scheduling alone does not guarantee a specific delay bound at a node, only that if it is possible to schedule all of the packets, then EDF will succeed in scheduling them.

2.1.11 Delay Earliest Due Date (Delay-EDD)

Delay-EDD [70] extends the idea of EDF scheduling into an admission control protocol which determines at each node what minimum delay bound can be met for a new traffic flow specification without violating delay guarantees to existing traffic flows. Flows are regulated by assignment of a deadline equivalent to the deadline that would have been assigned if the packet conformed to its agreed rate. For example, if a flow agrees to a packet rate of 5 packets/time unit, then the deadline assigned to the *j*th packet will be $j * 1/5 + d$, where d is the delay bound guaranteed to the flow, regardless of its actual arrival time at the node. This allows Delay-EDD to accommodate traffic distortion caused by variations in congestion at the earlier nodes. When a packet arrives for a given outgoing link, the packet is assigned a deadline as described above and then placed in a sorted priority queue for the outgoing link based on this deadline.

Dead-EDD requires that the outgoing links must not be saturated, that is the total traffic rate of incoming flows sharing the link must be less than the link's rate. In addition, Delay-EDD must ensure that the scheduler is not saturated. It is possible for the scheduler to be saturated even if the link is not saturated. Consider the node illustrated in Figure 2.2.

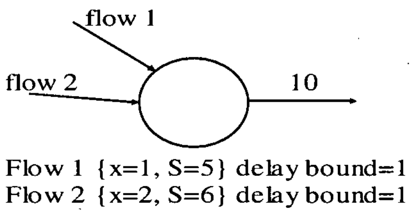
flow 1
flow 2
10
Flow 1 {x=1, S=5} delay bound=1
Flow 2 {x=2, S=6} delay bound=1

Figure 2.2 Scheduler Saturation

The incoming rate of flow 1 is 5 (5 bits every 1 time unit) and flow 2 has a rate of 3 (6 bits every 2 time units). The total incoming rte is 8 which is less than the outgoing link rate of 10, so the outgoing link itself is not saturated. It is still not possible to meet the delay bounds of the flows. If one packet arrives from each flow, the time taken to service both packets is 5/10+6/10 = 11/10 = 2.2 time units. Given both flows have the delay bound of 1, either may be serviced first. The packet that is serviced second will not be sent until 1.1 time units after its arrival. There, it is guaranteed that one of the flows will miss its deadline. Since the ordering of packets with the same deadline is arbitrary, the packet that misses its deadline is also arbitrary. We can not provide a guarantee that either flow will meet its delay bound. A test of node saturation is necessary but not sufficient to guarantee the flows are schedulable. An additional constraint is given in [70] that can be used to guarantee that scheduler saturation does not occur. Given the condition that for a set of J channels each flow i has a minimum interarrival time greater or equal to the time required to service one packet from each of the flows:

$$\forall i \in J : d_i \geq \sum_{k=1}^{J} S_k / r_{out} \tag{2.4}$$

the flows can be scheduled if the delay bound of each flow i, d_i meets the following criteria:

$$\forall i \in J : d_i \geq \sum_{k=1}^{J} S_k / r_{out} + S_{\max} / r_{out} \tag{2.5}$$

The last term is included to allow for the delay caused by a lower priority packet that is currently being serviced and can be preempted.

2.1.12 Jitter Earliest Due Date (Jitter-EDD)

Continuous media applications such as video and voice are affected not only by delay but also by variations in delay. Packets arriving too soon can negatively impact the application by making higher buffer demands on the receiver. This variation in delay is called jitter. Jitter Earliest Due Date [72] extends Delay-EDD to provide lower bounds, as well as upper bounds, on delay and new buffer requirements for flows with the bounded jitter.

Jitter-EDD nodes, as shown in Figure 2.3, are identical to Delay-EDD nodes with the following exceptions:

- In addition to the flow regulators that enforce the minimum packet spacing (i.e. the maximum rate), there are also jitter regulators for each flow that enforce a minimum delay.
- In addition to per node delay bounds, jitter bounds are also defined for each flow at a node.

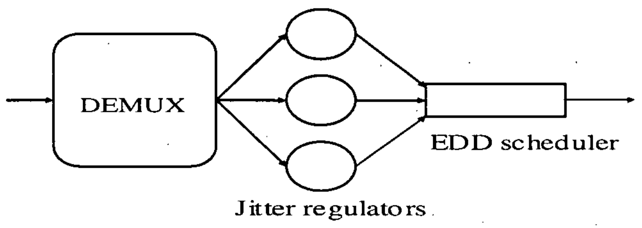

Figure 2.3 A Jitter-EDD node

As packets arrive on an input link, they are separated by flow and passed to a jitter regulator assigned to that flow. The regulator assigns an *eligibility time* to the packet. The packet is not forwarded to the EDD scheduler until this eligibility time has arrived. The eligibility time of a packet arriving at time t with delay bound d and jitter bound J is equal to $t+d-J$. Packets from a flow that bounds jitter will therefore not be sent more than J times units before their deadline. Packets from a flow that does not need to bound jitter would be eligible for scheduling at their time of arrival.

The scheme as described so far will guarantee the jitter behaviour at a single node based on the original specification of the flow. However, the traffic regulator, which ensures that the traffic meets its flow specification, only enforces minimum time between packets. It does not enforce a maximum time between packets, which would force the source to conform to its original jitter behaviour. The flow regulators for Jitter-EDD must be extended to remove any jitter experienced by the packet in the previous node.

A packet can be sent any time τ between $t+d-J$ and $t+d$. To determine the packet jitter at the previous node, the difference between a packet deadline ($t+d$) and the time it is actually sent on the output link of the previous node ($t+d-J \leq \tau \leq t+d$) is stamped on the packet. At the next node, the packet is held at the regulators until this time has elapsed before being sent to the EDD scheduler. This forces packets to be delayed if they arrive early. The jitter EDD policy effectively moves the buffering of packets from the received to the network and reduces the overall buffering required at each node by reducing the number of packets which could potentially arrive early (i.e. reducing the worst case scenario). The constraints on delay are the same as Delay-EDD.

2.2 Wireless network algorithms

Cao [8] provides an in-depth survey on recent research in wireless scheduling. For the completeness and relevance, some of the algorithms are mentioned here as well. The issues that are associated with wireless scheduling are similar to those of the wireline counterpart. However, there are issues in the wireless networks that warrant differentiation. These issues are mainly wireless link variability, which is caused by interference, fading, shadowing in the wireless channels (time-dependent problems), and location-dependent problems. Power constraint in the wireless terminals is also a significant issue in wireless networks. Algorithms should be kept simple in the wireless terminal, and less control or signaling information across the wireless is more desirable.

2.2.1 Channel State Dependent Packet Scheduling (CSDPS)

Channel state dependent packet scheduling (CSDPS) [73] addresses location-dependent and bursty errors in wireless scheduling. When a terminal is marked as bad state (when acknowledgement for data packet is not received), the scheduler no longer serves the queue until it is unmarked. A terminal is unmarked after a time-out period (e.g. average link error duration). Although it outperforms FIFO protocol in throughput and channel utilization, it dos not address the issues of delay guarantee, fairness, and throughput. It gives credits to mobiles in bursty errors rather than schedule bandwidth for them. BS then compensates all the credits later to guarantee the fairness. The fairness issue is addressed in [65] by combining class-based queueing (CBQ) to CSDPS. The authors proposed an enhanced CBQ for multiple classes of traffic with CSDPS to improve channel utilization, but delay bounds and other QoS guarantees such as loss rate are not discussed. The scheduling of VBR video with different rates at different times is not addressed.

2.2.2 Idealized Wireless Fair Queuing (IWFQ)

Idealized wireless fair queueing (IWFQ) [74] is proposed for packet scheduling in cell-structured wireless networks. IWFQ is the packet-based emulation of the unrealistic fluid model, wireless fluid fair queueing (WFFQ). The IWFQ model is defined with reference to an error-free WFQ service system. Each flow in the IWFQ has its own queue. When no link suffers from errors, it works just like the ordinary WFQ in wireless networks. When a packet of sequence number n of flow i arrives, it is tagged with virtual service start time, $S_{i,n}$, and finish time $F_{i,n}$. or specifically

$$\begin{aligned} S_{i,n} &= \max\{v(A(t)), F_{i,n-1}\}; \\ F_{i,n} &= S_{i,n} + \frac{L_{i,n}}{r_i} \end{aligned} \tag{2.6}$$

where

$L_{i,n}$	packet size of the arrived packet;
$v(A(t))$	system virtual time defined in WFQ.
r_i	service allocated to flow i.

The scheduler serves the packets based on non-decreasing finish virtual time (like in the WFQ) only from the terminals that are in good link state. By bounding the amount of *lagging* and *leading* in a flow, throughput and delay guarantees are achievable for IWFQ [74]. However, the limitation is pointed out in both [8]. It's not practical to require BS to know the arrival time of uplink packets, weight round robin (WRR) is used instead of WFQ. However, it is shown that the worst-case performance of the real implementation is much worse than IWFQ, and the average performance is very close to the idealized algorithm. Another limitation is that when a flow is compensated for its previous lagged service all other error-free flows will not be service at all. For the same reason, a lagging flow will receive compensation at a rate independent of its allocated service rate, violating the semantics that a larger guaranteed rate implies better QoS.

2.2.3 Channel-Condition Independent Packet Fair Queueing (CIF-Q)

The channel-condition independent packet fair queueing (CIF-Q) [75] is very similar to IWFQ in that it is an approximation of the ideal error-free systems, and it also defines a flow to be *leading*, *lagging*, or *satisfied* at any time instant if it receives more, less or the same amount of service as it would have received in the corresponding error-free system. CIF-Q addresses mainly the fairness issue (both long-term and short-term), delay and throughput guarantees, and graceful degradation for leading flows. Link error detection, estimation, and implementation issues are not discussed. The core of CIF-Q is Start-time Fair Queueing (SFQ) [76], although other wireline queueing algorithms can also be used.

2.2.4 Opportunity-Based Deficit Round Robin (ODRR)

In addition to DRR, ODRR scheduler incorporates channel status in the decision it makes in serving flows. However, it does not decrement the deficit counter like in DRR. Instead, it uses the quantity, *PanaltyFactor*, which indicates the efficiency with which the allocated resource was used. The *PenaltyFactor* can be greater than 1 if the channel status of a flow is poor, and this introduces exceptions that the algorithm needs to handle.

The rationale behind ODRR is the existing fair scheduling algorithms for wireless packets networks focus on only bandwidth that is successfully distributing to the flows; they do not consider the unsuccessful transmissions by a flows that also consume systems resources such as electrical power, and that the unsuccessful distribution also occupies the shared medium for periods of time during which another flow may have successfully used the medium. In other words, packet of a flow with poor link state, such as due to low signal power, will require a larger number of retransmission attempts and consume more system resources, and thereby, deny other flows the opportunity to transmit during the period.

Chapter 3 TOKEN BANK FAIR QUEUING ALGORITHM

In this chapter, a detailed description of the TBFQ scheduling algorithm, along with the rationale behind it, are presented. The characteristics and properties of the scheduler are explained as well.

TBFQ was first developed for wireless packet scheduling in the downlink channel [77]. It integrates the policing and scheduling functions. Its predecessors, the Leaky Bucket (LB) mechanism and its many multi-level variants, were used to police flows and to conform them to certain traffic profile. The LB mechanism stringently polices the negotiated parameters for individual connection. Such restriction degrades the statistical multiplexing of group connections.

In wireless mobile networks, strict scheduling schemes may not seem appropriate as flows cannot *a priori* determine their exact behavior as most schedulers would require. The environment of wireless scheduling requires *soft* handling of packets. This is the philosophy behind TBFQ [78]. We define *soft-QoS* provision of a session to be the graceful acceptance of traffic profile violation when excess bandwidth is available, provided the session does not exceed its bandwidth allocation in the long-term. This prevents sudden degradation of QoS experienced by end-user as a result of traffic profile violations. TBFQ penalizes violating traffic less severely as it is able to service a packet, which might otherwise be discarded by the per-flow policing mechanism, by distributing unused bandwidth from other connections. TBFQ exploits the statistical multiplexing of group connections to enhance bandwidth utilization - an important factor in wireless links. Used in conjunction with CAC, TBFQ guarantees the minimum rate of a connection in an error-free environment.

In wireless environment characterized by bursty and location-dependent channel condition, the kind of fairness that is endeavored by the algorithms mentioned in the previous chapter may not be appropriate. When a flow comes out of erroneous condition, it should receive service immediately at least at its reserved rate plus it should be compensated for its lack of service during the setback period. But at the same time, the algorithm should not allow the flow to be so greedy that it may affect other flows. TBFQ assures that all flows in the system receive the same normalized service as much as possible, while it is momentarily tolerable to have a flow receive more service if it has just come out of erroneous condition. We first introduce the algorithm as follows.

3.1 TBFQ algorithm

TBFQ is a frame-based algorithm similar to round-robin type algorithms, where each frame can be thought of as a round. It is a work-conserving algorithm. A work-conserving scheduler is never idle while there are packets waiting to be transmitted in service queues. On the other hand, a non-work-conserving scheduler may be idle even if there are packets waiting to be served. In a wireless network, it may be better to postpone a mobile terminal from transmitting if the channel condition is poor, this gives the opportunity for other terminals to utilize the bandwidth while their channel condition may be better. The postponing of scheduling service due to impaired channel condition can turn a work-conserving algorithm to a non-work-conserving one. We will introduce the effect of erroneous channel condition in Chapter 5. In this chapter, we introduce the generic TBFQ algorithm that is work-conserving and describe its characteristics and performance such as the fairness, delay bound, and its graceful services capability.

We will define the parameters of the algorithm with respect to a packet-based system. Each frame in a TBFQ can be considered as a round in a round-robin based system, except that in each frame the order of service will change according to their priority. A round is generally defined as having varied time intervals and the length of which depend on the completion of servicing all the flows in turn in the system. Frames are intervals with fixed time period. When a round-based algorithm is used in a wireless system, it is often converted to a more restricted frame-based usage because of the constraints imposed by many wireless systems. Throughout this chapter, without loss of generality, we define our algorithm under the round-based condition, which means a flow can be serviced during the round. Later on, when the algorithm is applied in a wireless system, frames will be used and specified and a flow can only receive the service at the end of a frame.

Referring to Figure 3.1 for the generic model of TBFQ, each flow i has its own data buffer and a Leaky Bucket associated with it that can be characterized as LB_i, = (r_i, P_i, D_i), for i=0,1,...n_i , where r is the token generation rate (bytes/s), p is the token pool size (bytes), and D is the data buffer size (bytes).

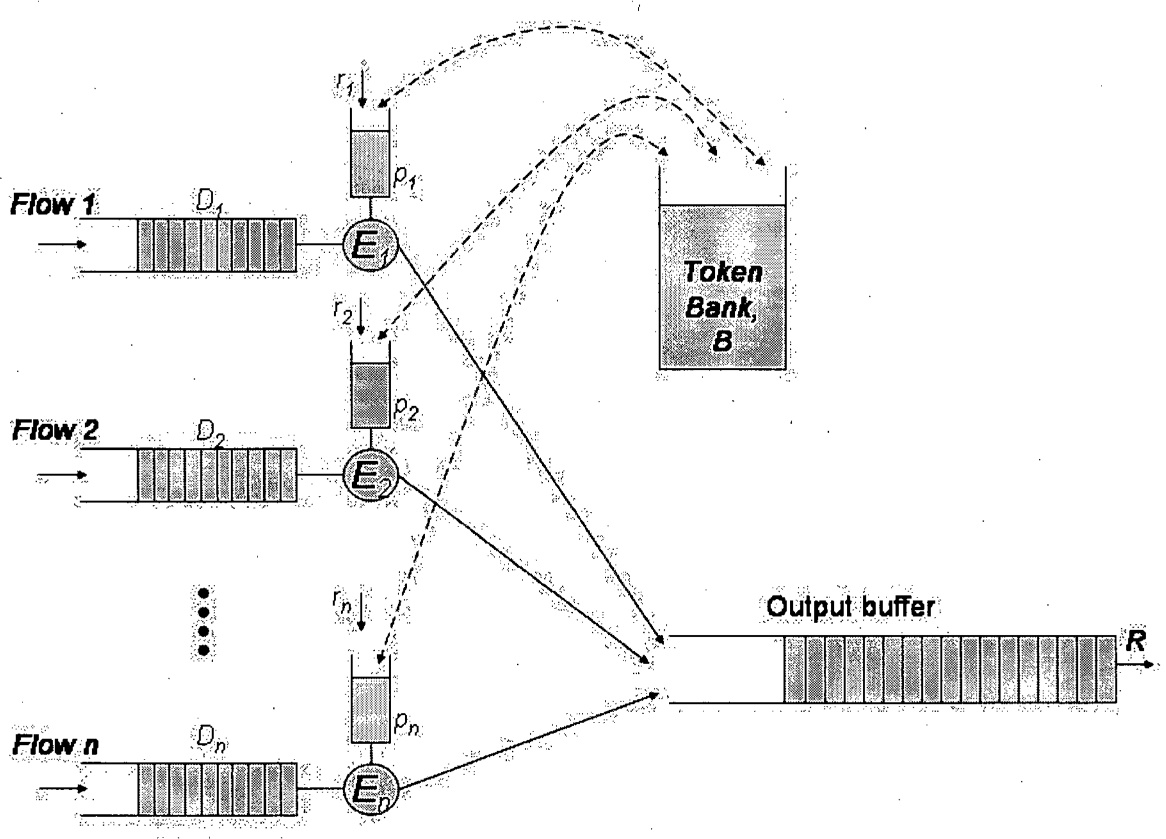

Figure 3.1 Generic token bank fair queuing model

Let's consider fixed sized packet of L bytes only. As data packets are generated from a flow, they are stored in their corresponding data buffer D. A packet can only be served when tokens are available. Each L-byte packet consumes L tokens. Tokens are generated at the rate r and into the token pool p. Whenever there are L tokens accumulated in a token pool, a packet can be admitted from the data buffer into the output data buffer. The rate that data is served in the output data buffer is R, where R>>r, and

$$\sum_{1}^{n} r_i \leq R \tag{3.1}$$

$$r_i \leq \frac{w_i}{\sum w_i} R \tag{3.2}$$

where w_i is the weight assigned to flow i.

Figure 3.2 illustrates the operation of the token pools when the flows are constant-bit-rate. In the top half of the figure, tokens (L packets) are generated at interval time at a predetermined rate. Packets are generated from Flow 1 at a constant bit rate as an example. The first data packet is generated after the first token had arrived which means the token pool is full and the first packet can be served immediately as reflected by the packet service curve.

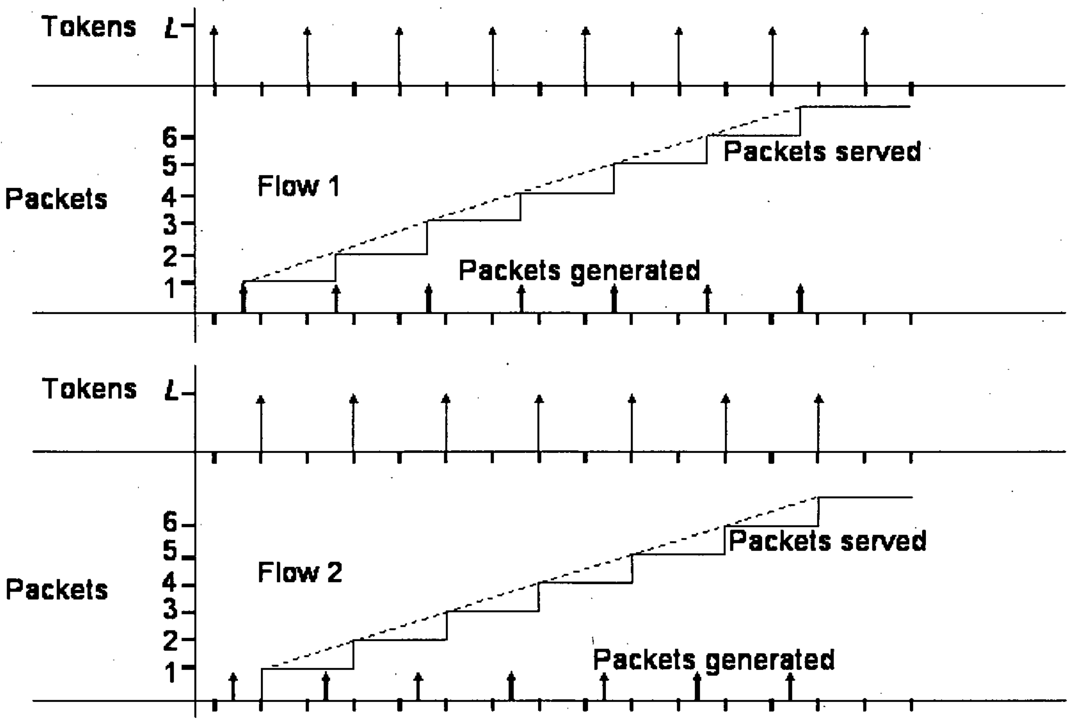

Figure 3.2 An illustration of the token pool operation in TBFQ

In Flow 2 on the other hand, data packets arrive before tokens become available and the first packet will wait until the tokens have arrived and be served, and likewise for the subsequent packets. Both flows have packets generated at the same rate, even though their service order is different, and because their token rates are matched to their data rate, their service received are the same and have the same service slope which equal to their token generation rate.

In addition to the token pools, the TBFQ is governed by few other parameters. For each connection i, the token balance E_i is a counter that keeps track of the number of tokens borrowed from or given to the token bank. As tokens are generated at rate r_i the tokens overflowing from the token pool are added to the token bank and E_i is incremented by the amount of tokens given. When the token pool is depleted due to servicing the flow, and if there are still packets remain to be served in the data buffer D, tokens can be borrowed from the bank by connection i, and E_i is decreased by the number of tokens borrowed. Thus during periods that the incoming traffic rate of connection i is less than its token generation rate, the token pool would always has enough tokens to service arriving packets, and E_i becomes positive and increasing. On the other hand, during periods that the incoming traffic rate of connection i is greater than its token generation rate, the token pool is emptied at a faster rate than it can be refilled with tokens. In this case, the connection may borrow tokens from the bank. This is often necessary under variable-bit-rate or even bursty traffic scenarios.

The priority of a connection that can borrow tokens from the bank is determined by the priority index, (E_i/r_i). The connection that has the highest index value has the highest priority in borrowing tokens from the bank; hence they will be serviced first. The number of token a connection may borrow from the bank at each time should be limited as it affects the burstiness of the outflow as well as fairness to others. To avoid starvation to other connections, '*debt limit*' (d_i) is imposed. If E_i reaches the debt limit d_i, the connection can no longer allow borrowing from the bank. The debt limit, d_i, is a negative value, and is imposed so that a malicious connection cannot affect the QoS of other well behaved connections. We also define '*burst credit*', c_i, as the maximum number of tokens connection i can borrow from the bank each time. For a constant-bit-rate source, r_i equals the source peak rate, and there is no need to borrow tokens from or deposit tokens into the bank. E_i ideally stays zero all the time, and c_i has no relevance what so ever. However, for bursty sources, c_i should be set to a suitably large to accommodate the bursty nature of an application, yet not too large as to affect the fairness of the connections (we will consider the characteristics these parameters have on TBFQ in later sections). A connection may borrow token from the bank until its debt limit is reached, then it must wait until it has deposited enough tokens to the bank to reach the '*creditable threshold*'.

Figure 3.3 illustrates the operation of TBFQ in a bursty traffic scenario. Let L-byte tokens be generated at a regular interval of $1/r$. The first three data packets are served because of the token pools are filled and E remains at 0. The token generated at t=4 again fills the token pool. From t=5 to t=9, each new token accumulates to E, and E increases at the rate of r. Shortly after t=9, a burst of 9 packets arrive. At this point, the token pool is filled but not enough to serve the request and E has accumulated to 5. With *burst credit*, c=10 and *debt limit*, d=-3, we can borrow up to the debt limit, which is E-d=8. However, this can only serve 8 packets. The remaining 1 packet will have to wait until a token becomes available which is at t=10. E continues to stay at the debt limit until all packets are served and then it begins to accumulate at the rate r.

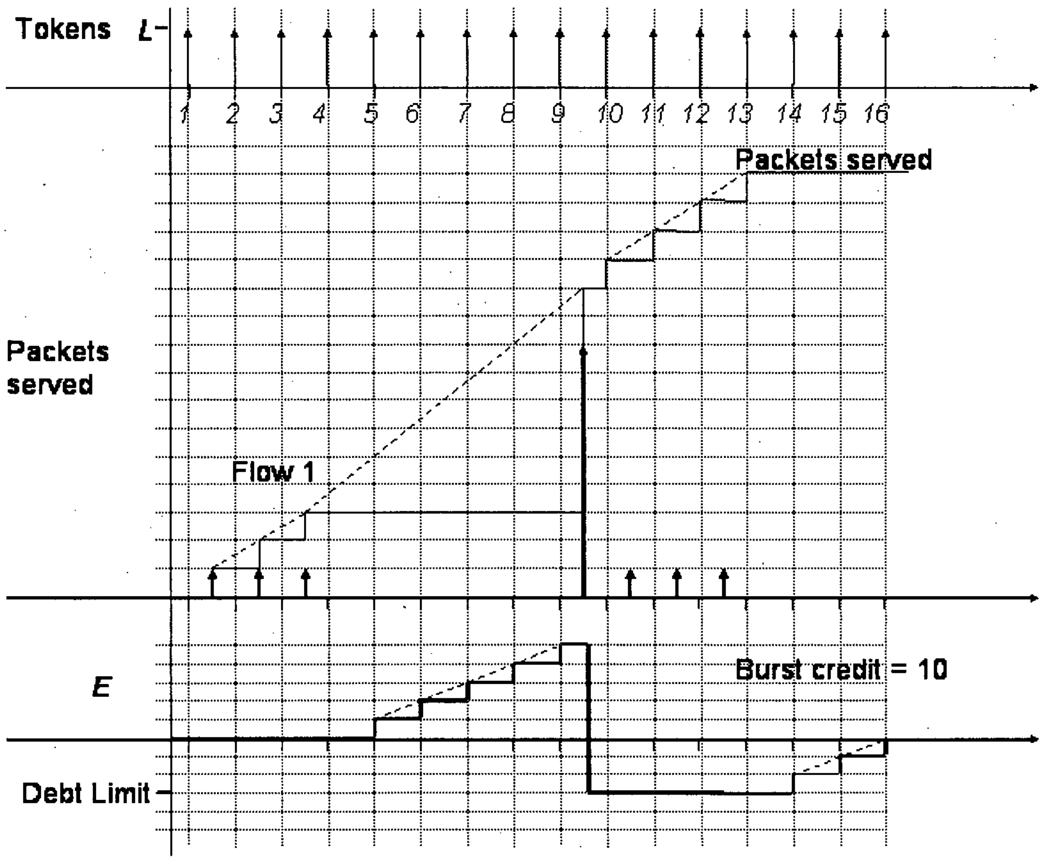

Figure 3.3 An illustration of TBFQ operation with bursty traffic.

A Pseudo-code implementation of the generic TBFQ scheduling algorithm is shown in Figure 3.4. The *Initialize* function is invoked before the TBFQ can be utilized, and it allocates memory for data structures and TBFQ parameters. The *Enqueue* function is invoked when a packet arrives. If a packet belongs to an unknown flow, provision for a new flow is necessary. This includes memory allocation to accommodate the parameters associated with the new flow, adjusting the total number of flows, assigning the appropriate token generation rate, and insert the packet(s) into the new queue. The *Tokengenerated* function is invoked when an event triggered by the generation of a token for flow *i*. Upon activation, if the pool is empty, the new token will fill the pool; otherwise, the overflowing token is added to *E*.

```
Initialize()        //invoked before the scheduler can be used
    Constants: _NUMFLOW, _MAXQSIZE, _MINTOKENRATE, _PACKETSIZE, _BANKSIZE;
    Allocate (InputQueue[]);          // contains incoming packets
    Allocate (Rate[]);                //token generation rate for each flow
    Allocate (TokenPool[]);           //token pool
    Allocate (E[]);                   //keeps track of  token borrowed/replenished
    Allocate (PriorityIndex[]);       //stores the priority of each flow
    Allocate (Priority[]);            // prioritized flow order. Index 0 has the highest priority
    Allocate (Backlogged[]);          //TRUE if there is one or more packet in the queues, else FALSE.
    Allocate (DebtLimit[]);           //keeps the debt limit, d, of all the flows
    Allocate (BurstCredit[]);         //keeps the burst credit, c, of all the flows
    Allocate (CreditableThresh[]);    //keeps the creditable threshold, h, of all the flows
    For (i = 0; i<_NUMFLOW; i += 1)
        Qsize[i] = _MAXQSIZE;
        Rate[i] = _MINTOKENRATE;
        TokenPool[i] = _PACKETSIZE;       // in Bytes
        E[i] = 0;
        Bank = _BANKSIZE;

Enqueue(i, packet, r, d, c, h)          //invoked when a packet arrives
// packet        the actual data packet(s)
//  i            flow ID number
//  r            token generation rate for flow i
//  d            debt limit for flow i
//  c            burst credit for flow i
//  h            creditable threshold for flow i
    If (i >= _NUMFLOW) then
        _NUMFLOW += 1;
        Rate[i] = r;
        DebtLimit[i] = d;
        BurstCredit[i] = c;
        CreditableThresh[i] = h;
        E[i] = 0;
        TokenPool[i] = _PACKETSIZE;
        packet --> InputQueue[i];
    Else
        packet --> InputQueue[i];

Tokengenerated(i)          //invoked when the event of a token is generated for queue i
    If  token pool i is Empty then fill the pool with the new token
    Else E[i] += 1;

TBFQ()          //TBFQ scheduling decision
    // First stage:  admit packets according to token pool
    // This stage scans all the queues and grants service to the queues that is non-empty and token pool is full
    For (i = 0..n-1)
        If  (Backlogged[i] && TokenPool[i] == _PACKETSIZE) then
            Admit one packet from InputQueue[i] to output queue
            TokenPool[i]=0;
            Bank = Bank - _PACKETSIZE;

    // Second stage:  admission by borrowing tokens from Bank
    // This stage lets each flow borrow from token bank according to their Priority Index, as long as there
    // is resources left in the bank.
    QuickSort(E[], r[], Priority[]);          // standard quick sort algorithm for sorting index E/r_i
    While (Bank>0)
        For (i=0..n-1)
            q = Priority[i];
            If Backlogged[q] then Break;

        Admit one packet from InputQueue[q] to output queue
        E[q] -= _PACKETSIZE;
        Bank = Bank - _PACKETSIZE;
        QuickSort(E[], r[], Priority[]);
    // end while
    Bank = _BANKSIZE;
```

Figure 3.4 Pseudo code of generic TBFQ

The *TBFQ* function will be invoked when the scheduling decision is needed, which is at the beginning of each frame. This is the generic TBFQ algorithm, its application in wireless systems and performance will be discussed in detail in Chapter 5 and Chapter 6.

There are two stages in the *TBFQ* decision process. In the first stage, a packet is admitted to the output queue if the corresponding token pool is full and there is backlog in the queue. In the second stage, the remaining bandwidth in the system will be allocated according to the priority index of each flow,

$$\textit{Priority Index},\ \mathrm{P}_i = \frac{E_i}{r_i} \tag{3.3}$$

Ideally, the rate of increase of the priority index at each instant should match the rate of increase of the potential of a connection currently being serviced by the scheduler. In practice, however, a much more relaxed definition of the system potential function is adequate.

In each frame, the total amount of resources that are available to serve all the flows is set to **B.** Thus,

$$R = \frac{B}{T} \tag{3.4}$$

where T is the period of a frame. After each flow is served according to their pool size p, the remaining amount of resources that can be used for distribution is,

$$B' = B - \sum_{i \in n} \left\lfloor \frac{p_i}{L} \right\rfloor \tag{3.5}$$

where p_i the token pool for flow i, *and* $0 \le p_i \le L_i$, and L is the size of a packet.

Using the *QuickSort* function as a standard sorting algorithm, we first calculate the priority index of each flow and then sort through the priority index. The result is stored in the *Priority* array. The first index in the *Priority* array contains the flow ID with the highest priority index. For each backlogged queue in the order of decreasing priority index, the maximum number of packets, or *throttle,* that flow i can potentially be admitted into the output queue is

$$\textit{Throttle} = E_i - d_i \tag{3.6}$$

However, the redistribution of the bank's resources must be done in a fair manner according to their priority P**.** Resources are given first to the backlogged flow with the highest P**.** The backlogged flow with the highest P can continue to borrow token from the bank until it no longer has the highest P. Obviously, no flow can borrow token from the bank when the bank is depleted. However, the bank is replenished at the end of each frame.

After packet(s) are sent to the output queue, E is subtracted by the number of packets admitted. E can be represented by

$$\begin{gathered} E_i(t) = r_i(t) - Sent_i(t) \\ i = \{0,1,\ldots,n\} \\ 0 \le t \le T_{session} \end{gathered} \tag{3.7}$$

In other words,

$$r_i(t) = E_i(t) + Sent_i(t) \tag{3.8}$$

which means that the token generated over time is either consumed by the packets sent or credited into E_i. The number of fix-sized packets sent, $Sent_i(t)$ can be described as

$$\begin{gathered} Sent_i(t) = n \cdot c_i \\ n \in \mathbb{N} = \{0,1,2,\ldots\} \\ 0 \le t \le T_{session} \end{gathered} \tag{3.9}$$

where $c_i \equiv$ *burst credit*, n is the number of times flow i borrows from the bank during $T_{session}$. The variable $T_{session}$ is the duration of a session; it is implementation specific and is determined by the service providers. For example, the definition of a session can be during the time between users initiate a voice or video communication and until its termination. In the case of non-real-time data services, such as web surfing or email, $T_{session}$ may equate the time-out mechanism imposed by either the client-side or server-side application.

From Equations (3.7) and (3.6), the token balance counter is limited by,

$$Debt\_Limit \le E_i(t) \le r_i \cdot T_{session} \tag{3.10}$$

3.2 Work Complexity

Work complexity is another way of evaluating an algorithm. In sorted-priority based queuing, one can qualify the complexity of computing the system virtual time like in the WFQ for example. For WFQ, the worst-case complexity is $O(n)$ where n is the number of flows sharing the same output link. However, in the case of DRR, SRR, and ERR, the complexity of computing the virtual time is $O(1)$. One can also quantify the complexity of maintaining a sorted list of packets based on their timestamps, and the complexity of computing the maximum or the minimum in this list prior to each packet transmission. For n flows, the work complexity of the scheduler prior to each packet transmission is $O(\log n)$.

The work complexity of TBFQ scheduling discipline is defined as follows,

Definition 3.1 *Consider an execution of a scheduling discipline over n flows. We define the work complexity of the scheduler as the order of the time complexity, with respect to each arriving packet into the system.*

Note that, this definition of work complexity does not include the transmission time of the packet. For example, in WFQ the complexity is O(n), where n is the number of connections, which means that for each arriving packet among the connections, a computation for the finishing time stamp is necessary and that computation complexity is O(n).

Definition 3.2 *Consider packets in an input queue waiting to be served. We define backlogs to be the packets that are still waiting in a queue. A connection is said to be backlogged, if there is at least one packet in the queue. Backlogs can occur in the input queue and/or output queue.*

Definition 3.3 *We define a backlogged session to be the time period during which there is backlog in a flow.*

Definition 3.4 *Consider a backlog packet, we define a backlog packet is served when it leaves the queue.*

Theorem 3.1 *The TBFQ scheduling algorithm has time complexity O(1) for scheduling packets within n admitted connections.*

Proof: We prove the theorem by showing that enqueuing and dequeuing a packet are each of time complexity O(1).

The time complexity of enqueuing a packet is the same as the time complexity of the *Enqueue* function in Figure 3.4, which is executed whenever a new packet arrives at a flow. Examining the *Enqueue* function, it is obvious that the work complexity of determining the flow at which the packet arrives is O(1). Whether the packet belongs to an existing queue or to a new one, it will be identified and is appended to the end of the appropriate queue. The operation of appending a packet to the tail of a queue has also a work complexity of O(1).

Let's consider the complexity of the dequeue operation. The process of dequeue operation is independent from the enqueue operation. Although there is a priority operation to be determined by sorting, which is computed at the end of each cycle (or frame), the sorting is not computed on a per packet basis but rather on a per connection basis. The number of arriving packets, M >> n, where n is the number of connections. From the definition of work complexity, the computation is determined at the order of time complexity on a per packet basis. Therefore, WFQ has the complexity of O(n) for each arriving packet. TBFQ priority computation, however, does not involve time-stamping and is computed independent from any arriving packets. Furthermore, during the process of dequeueing a packet, the priority of each connection had already been determined. Therefore, in relationship to each arriving packet, the work complexity of TBFQ is $O(1)$. ■

3.3 Rate guarantee

In this section, we show that the generic TBFQ provides a guaranteed rate service for each flow.

Lemma 3.1 *For all backlogged connections that are served under the TBFQ scheduling discipline, where each connection has a token generation rate of r_i assigned, then each connection is guaranteed the minimum service rate of r_i.*

Proof: Since each connection has its own token pool and tokens from each pool is generated independently. Therefore, at the minimum, the backlogs in each input queue are served by its replenishing token pool and are served independently. A backlogged packet in an input queue can only be served when a token is available. Since token generation rate stays static during a connection, packets in the input queue are served at r_i if there is no token credit to be borrowed from the bank. If tokens can be borrowed from the bank, more packets can be served and the backlogs is said to be served at a faster rate than r_i. Since the worst case is when no tokens can be borrowed, the backlogs are served at the minimum rate of r_i. ■

3.4 Fairness

In attempt to determine the fairness of a fair resource allocation policy, one has to first define a notion of fairness that determines the criteria by which one can judge the fairness achieved by an allocation policy. To solve the problem of what is fair if multiple entities compete for a single shared resource, many fairness criteria have been proposed in the literature. One of the most popular ones is max-min fairness. The max-min fair share policy of allocating a shared resource among multiple flows with equal rights to the resource but unequal demands, follows the following principles [58][79]:

- The shared resource is allocated in order of increasing demand.
- No flow receives a share of the resource larger than its demand.
- Flows with unsatisfied demands receive equal shares of the resource.

The notion of max-min fairness can also be defined in the following equivalent way: no flow can increase its allocation without reducing the allocation of another flow that has less or equal demand.

Under max-min fairness, given no additional resources, an unsatisfied flow cannot increase its allocation by merely demanding more.

GPS is an ideal scheduling discipline that satisfies the above notion of max-min fair allocation. Although GPS is the ideal fair scheduling algorithm that can achieve max-min fairness, it cannot be implemented in practice, fairness for a practical scheduling policy *S* that approximates GPS can be measured by the discrepancy between the service achieved by GPS and that by the practical policy *S*. This quantity, known as the *Absolute Fairness Bound (AFB)* of a scheduler *S*, is defined as the upper bound on the difference between the service received by a flow under *S* and that under GPS over all possible intervals of time. This bound is often difficult to derive analytically. However, it has been shown in [80] that the *AFB* is related by a simple equation to another popular fairness measure known as the *Relative Fairness Bound (RFB)* first proposed in [60]. The *RFB* is much easier to evaluate than *AFB*.

The *RFB* is defined as the maximum difference in the service received by any two flows over all sessions, where $0 \le t \le T_{session}$. The following provides a more rigorous definition.

Definition 3.5 *A session, which has a duration $T_{session}$, is the time period for a flow i which consists one or more backlogged periods.*

Definition 3.6 *For all flows that are of the same type, let $S_i(t1, t2)$ be the number of bits transmitted by flow i during the time interval between t1 and t2. Given an interval (t1, t2), we define the Relative Fairness, RF(t1; t2) for this interval as the maximum value of* $|Sent_i(t_1,t_2) - Sent_j(t_1,t_2)|$ *over all pairs of flows i and j that are active during this interval. Define the relative fairness bound (RFB) as the maximum of RF(t1; t2) for all possible time intervals (t1; t2), and* $0 \le t_1 < t_2 \le T_{session}$.

It is desirable that *RFB* be a small constant. The smaller the *RFB*, the closer the scheduler emulates the GPS scheduler which is considered an ideal fair scheduling algorithm.

Lemma 3.2 *In a continuously backlogged period during the period $[t_1,t_2]$, $RF(t_1,t_2) \le d$.*

Proof: In a continuously backlogged period, there is always packet in the queue waiting to be served. From ***Lemma* 3.1**,

$$Sent_i(t_1,t_2) \ge r_i \cdot (t_2 - t_1) \tag{3.11}$$

Since it is continuously backlogged,

$$\begin{gathered} 0 \ge E(t_1) \ge E(t_2) \ge d_i \\ 0 \le t_1 \le t_2 \le T_{session} \end{gathered} \tag{3.12}$$

The maximum flow i can borrow is d_i. So, if flow j is allowed to have the maximum service,

$$Sent_j(t_1,t_2) \le r_j \cdot (t_2 - t_1) + d_j \tag{3.13}$$

RF(t1, t2) equals to,

$$\left|Sent_j(t_1,t_2) - Sent_i(t_1,t_2)\right| = r_j \cdot (t_2 - t_1) + d_j - r_i \cdot (t_2 - t_1) \tag{3.14}$$

Since $r_i = r_j$ and $d_i = d_j$, Equation (3.14) becomes

$$\left|Sent_j(t_1,t_2) - Sent_i(t_1,t_2)\right| = d_j \tag{3.15}$$

Since this is the maximum difference in service, therefore $RF(t1, t2) \le$ d. ■

This tells us that two flows that are within the same class and have the same token rate, r_i, will receive fair service. However, what would be the fairness if two flows are from different service classes and have different token rates? To answer this question, we must first normalize the service received by their contracted token rate and then compare the difference in service.

Definition 3.7 *Let us define a fairness index, FI, such that*

$$FI[t_1,t_2] = \left|\frac{Sent_i(t)}{r_i(t)} - \frac{Sent_j(t)}{r_j(t)}\right|_{t=t_1}^{t=t_2} \tag{3.16}$$

A service discipline is said to be fair if *FI* is bounded for all periods of t. Perfect fairness is achieved when *FI=0.*

Theorem 3.2 *For any two flows, i and j,* $FI[t_1,t_2] = \left|\frac{E_i[t_1,t_2]}{r_i[t_1,t_2]} - \frac{E_j[t_1,t_2]}{r_j[t_1,t_2]}\right|$.

Proof: From Equation (3.8),

$$Sent_i(t) = r_i(t) - E_i(t) \tag{3.17}$$

$$r_i(t) = r_i \cdot t \tag{3.18}$$

During the period $[t_1, t_2]$,

$$Sent_i[t_1,t_2] = r_i \cdot (t_2 - t_1) - E_i[t_1,t_2] \tag{3.19}$$

Normalized by $r_i(t)$, Equation (3.19) becomes

$$\frac{Sent_i[t_1,t_2]}{r_i[t_1,t_2]} = \frac{r_i \cdot (t_2 - t_1)}{r_i \cdot (t_2 - t_1)} - \frac{E_i[t_1,t_2]}{r_i \cdot (t_2 - t_1)} \tag{3.20}$$

$$\frac{Sent_i[t_1,t_2]}{r_i[t_1,t_2]} = 1 - \frac{E_i[t_1,t_2]}{r_i \cdot (t_2 - t_1)} \tag{3.21}$$

For flow *j,* the service received is

$$\frac{Sent_j[t_1,t_2]}{r_j[t_1,t_2]} = 1 - \frac{E_j[t_1,t_2]}{r_j \cdot (t_2 - t_1)} \tag{3.22}$$

From Equations (3.21) and (3.22),

$$\left| \frac{Sent_i[t_1,t_2]}{r_i[t_1,t_2]} - \frac{Sent_j[t_1,t_2]}{r_j[t_1,t_2]} \right| = \left| \frac{E_i[t_1,t_2]}{r_i \cdot (t_2 - t_1)} - \frac{E_j[t_1,t_2]}{r_j \cdot (t_2 - t_1)} \right| \tag{3.23}$$

$$FI[t_1,t_2] = \left| \frac{E_i[t_1,t_2]}{r_i[t_1,t_2]} - \frac{E_j[t_1,t_2]}{r_j[t_1,t_2]} \right| \tag{3.24}$$

■

From Theorem *3.2*, we can determine that in a period when the system is backlogged, the fairness index is bounded. To show that the fairness index *FI* is bounded over all time interval [t_1, t_2], we must examine the change in *E* over all possible scenarios in time as in (3.24).

Case 1: *E(t)* is constant, but > d_i, the *debt_limit.*

This occurs when the incoming traffic equates token rate, therefore, the service rate is the same as the token rate (See Figure 3.5), i.e.

$$\frac{d}{dt} Sent_i(t) = \frac{d}{dt} r_i(t) \tag{3.25}$$

and,

$$\frac{d}{dt} E_i(t) = 0 \tag{3.26}$$

$$E_i[t_1,t_2] = E_i(t_2) - E_i(t_1) = 0 \tag{3.27}$$

Case 2: $E(t) = d_i$.

This occurs when the flow continues to borrow tokens and reaches the *debt limit.* Again, $E_i[t_1,t_2] = E_i(t_2) - E_i(t_1) = 0$.

Case 3: *E(t)* is increasing.

E(t) can only increase when there is no more packets to be served, and the input queue is empty. Therefore, from (3.17),

$$E_i(t) = r_i(t) - Sent_i(t) = r_i(t) \tag{3.28}$$

$$E_i[t_1,t_2] = r_i[t_1,t_2] = r_i(t_2) - r_i(t_1) \tag{3.29}$$

$$E_i\left[t_1,t_2\right] = r_i(t_2 - t_1) \tag{3.30}$$

This shows that when E_i increases, it increases at the rate of r_i.

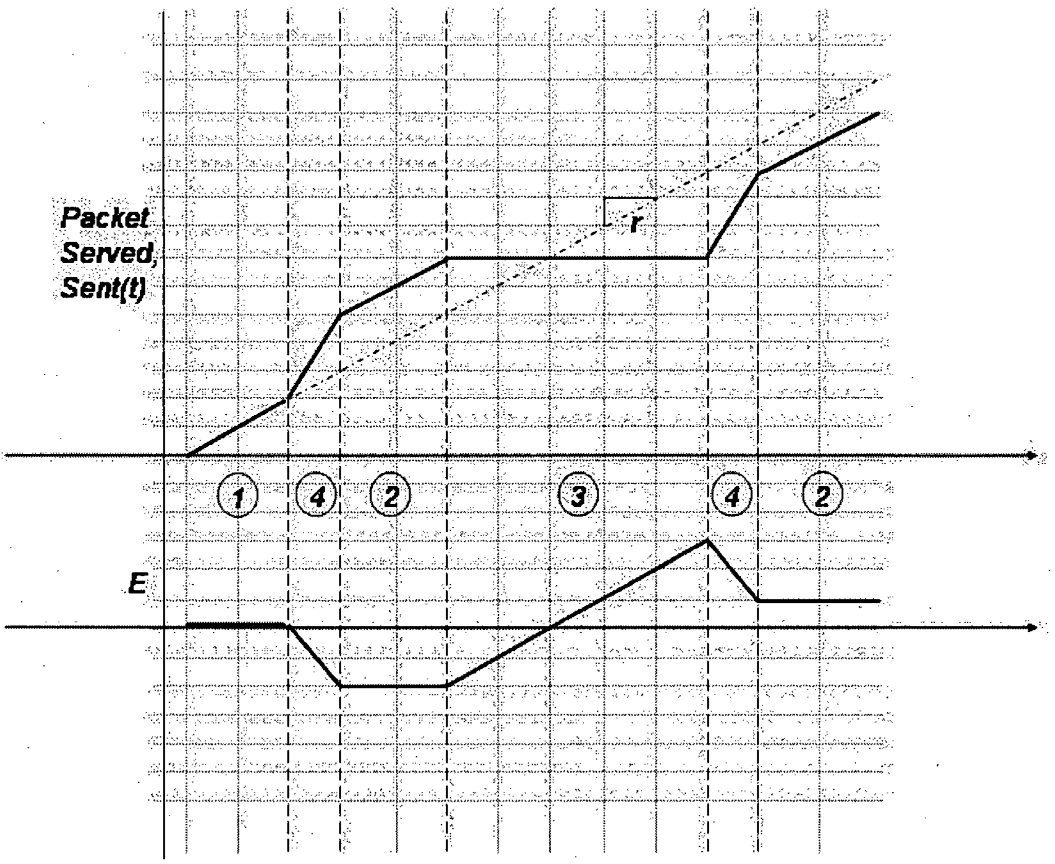

Figure 3.5 Relationship between token balance counter *E* and *Sent*(t)

Case 4: *E(t)* is decreasing.

From (3.17),

$$\frac{d}{dt}E_i(t) = \frac{d}{dt}r_i(t) - \frac{d}{dt}Sent_i(t) < 0 \tag{3.31}$$

This occurs when a flow borrows tokens from the bank. In addition to the token in the token pool, the flow is allowed to borrow an amount of token equal to the burst credit, c_i. A flow may be allowed to borrow more than one c_i within a round. Therefore, from Lemma *3.1*, the service rate is at least $r_i(t)$ plus the resources that come from the borrowed tokens, i.e.

$$Sent_i(t) = r_i(t) + n \cdot c_i \cdot t \tag{3.32}$$

where $n \cdot c_i$ is the largest number of c_i borrowed. So, from (3.31) and (3.32),

$$E_i[t_1, t_2] = -n \cdot c_i \cdot (t_2 - t_1) \tag{3.33}$$

Theorem *3.3 In TBFQ, when the system is backlogged, the fairness index $FI[t_1, t_2]$ is bounded in all time intervals $[t_1, t_2]$.*

Proof: From (3.24), we know that $FI[t_1, t_2]$ is the difference between the normalized token balance of each flow. When token balance E remains constant, as in cases 1 and 2, $E_i[t_1, t_2] = E_i(t_2) - E_i(t_1) = 0$, therefore

$$\frac{E_i[t_1, t_2]}{r_i[t_1, t_2]} = 0 \tag{3.34}$$

$FI[t_1, t_2]$ reaches its maximum when flow *i* is in either case 1 or 2, and flow *j* is in case 4, i.e.,

$$FI[t_1, t_2] = \left| 0 - \frac{-n \cdot c_j \cdot (t_2 - t_1)}{r_j[t_1, t_2]} \right| = \frac{n \cdot c_j}{r_j} \tag{3.35}$$

Both c_j and r_j are constant, and *n* is finite. Therefore, the fairness index is bounded between 0 and $\frac{n \cdot c_j}{r_j}$. ■

Fairness is often defined with respect to how excess bandwidth is distributed to the backlogged flows. In an error-free wired network, that is a reasonable measure. For example, WFQ and other time-stamped based algorithms try to achieve instantaneous fairness by calculating the *finish time* of a packet served under the fluid model. The *finish time-stamp* is calculated by distributing excess bandwidth proportionally according to the weighted share of a flow. However, this calculation is done only during active (or backlogged) periods; in other words, the inactive time of a flow has no significance in the fairness. We could also say that there is no *memory* on the quality of service received in the previous active sessions. In Deficit Round Robin, fairness is enforced to make sure variable-sized packets are treated fairly. So, the arrival of a very large packet from one flow would not affect the service of a small packet in another flow. An *allowance* (in number of bits) for a flow in the a round is proportional to how much *deficit* (in bits) it has conceded relative to the *surplus* given to another flow (due to larger packets) in the previous round. In a way, a history (or memory) is kept for the *over service* due to the service of larger-sized packets. However, this memory mechanism applies only to the variation in packet size and to backlogged period only; when the flows become inactive (or when backlogged is cleared), their memory is reset.

We argue that the inactivity, on the contrary, is of significant importance in wireless systems. It is the cause of the inactivity that separates wireless scheduling from the wired counterpart. This inactivity is often caused by the location-dependent channel error condition in wireless networks. To measure fairness in a wireless link without taking into account of the inactivity that is caused by the very nature of such link, would seem to be *unfair*. TBFQ has an inherent memory mechanism that takes into consideration of the inactivity through the token balance counter E which increases at the rate of r_i (as shown in (3.30)) during the inactivity. Fairness is user-centric in TBFQ as oppose to system-centric found in many of exiting scheduling algorithms. User-centric takes the perspective from users who have an expectation in:

- Receiving at least the guaranteed minimum throughput (in error-free condition) offered by the service provider;
- Receiving more than the guaranteed minimum throughput if the system has excessive capacity;
- Rolling forward resources previously unused and redeeming them whenever needed because users have memory capacity longer than that of the system;
- Isolating from the effect of their neighboring users.

User-perception is very difficult, if not impossible, to quantify. In a wireless environment, instantaneous fairness may not be appropriate. Scheduling algorithms that has longer memory in tracking fairness is more suitable in wireless systems.

3.5 Latency Rate Servers

In this section we begin with a brief overview of the concept of *Latency Rate* servers, first introduced in [81]. The theory of *Latency Rate (LR)* servers provides a means to describe the worst-case behavior of a broad range of scheduling algorithms in a simple manner. From the *LR* concept, we analyze the latency rate performance of TBFQ.

The two key parameters that determine the behavior of a *LR* server are the *latency* and the *reserved rate* of each flow. The latency of a *LR* server is a measure of the cumulative time that a flow has to wait until it begins receiving service at its guaranteed rate. The latency of a particular scheduling algorithm may depend on a number of factors such as internal parameters of the scheduling discipline, the reserved rates of the other flows multiplexed on the same output link and the transmission rate of the flow on the output link. It has been shown in [81] and [82] that several well-known scheduling disciplines such as Weighted Fair Queuing (WFQ), Self-Clocked Fair Queuing (SCFQ), Virtual Clock and Deficit Round Robin (DRR) belong to the class of *LR* servers. We first present some definitions and notations which will be useful in understanding the concept of *LR* servers.

Definition 3.8 *A system busy period is defined as the maximal time interval during which the server is continuously transmitting packets.*

Definition 3.9 *We define a flow as active during an interval of time, if at all instants of time during this interval, it has at least one packet awaiting service or being served.*

We now define the notion of a *busy period* which is an essential component of the concept of *LR* servers.

Definition 3.10 *A busy period of a flow is defined as the maximal time interval during which the flow is active if it served at exactly its reserved rate.*

Let ρ_i be the reserved rate for flow *i*. Also let $A_i(t_1, t_2)$ denote the total number of bits of flow *i* that arrive at the scheduler during the time interval (t_1, t_2). Consider an interval of time (τ_1, τ_2) which represents a busy period for flow *i*. Then for any time interval (τ_1, t) such that $t \in (\tau_1, \tau_2)$, the number of bits that arrive during this interval is greater than or equal to the number of bits that would exit the scheduler if the flow received service at its reserved rate, ρ_i. In other words, for all $t \in (\tau_1, \tau_2)$,

$$A_i(\tau_1, t) > \rho_i(t - \tau_1) \qquad (3.36)$$

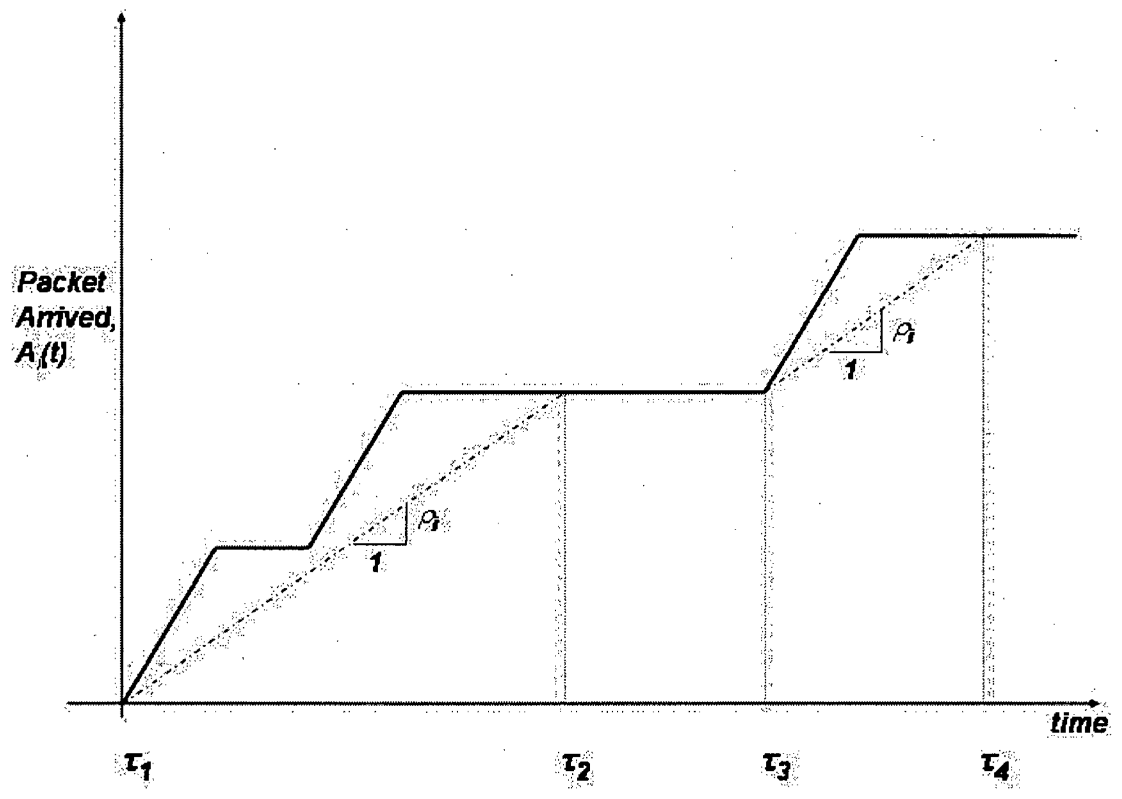

Figure 3.6 Two busy periods for flow *i*.

A graph of $A_i(\tau_1,t)$ against time is plotted in Figure 3.6. Figure 3.6 illustrates two busy periods, (t_1,t_2) and (t_3,t_4) for flow *i*. It is important to understand the basic difference between a session busy period and a session active period. The definition of the busy period supposes that flow *i* is served at the constant reserved rate, and therefore, depends only on the reserved rate of the flow and the packet arrival pattern of the flow. An active period of a flow, however, reflects the actual behavior of the scheduler where the instantaneous service offered to flow *i* varies according to the number of active flows. If during a busy period of flow *i*, the instantaneous service rate offered to flow *i* is greater than the allocated rate, then the flow may cease to be active. Thus, a busy period of a flow may include multiple active periods for that flow. The start of a busy period of a flow is always caused by the arrival of a packet belonging to the flow.

Note that, when the same traffic distribution is applied to two different schedulers with identical reserved rates, the ensuing active periods of the flows can be quite different. This makes it difficult to make use of active periods to analyze a broad class of schedulers. On the other hand, the busy period of a flow depends only on the arrival rate of the flow and its reserved rate. Therefore, the busy period can be used as an invariant in the analysis of different schedulers. It is because of this important property that the definition of an *LR* server is based on the service received by a flow during a busy period.

The following definitions lead to a formal notion of latency in the case of guaranteed rate servers. The reader is referred to [82] for a more detailed discussion.

Definition 3.11 *Define $Sent_i(t_1,t_2)$ as the amount of service received by flow i during the interval (t_1,t_2).*

Definition 3.12 *Let time instant α_i represent the start of a certain busy period for flow i. Let $t > \alpha_i$ be such that the flow is continuously busy during the time interval (α_i,t). Define $S_i(\alpha_i,t)$ as the number of bits belonging to packets in flow i that arrive after time α_i and are scheduled during the time interval (α_i,t).*

Note that, $Sent_i(\alpha_i,t)$ is not necessarily equal to $S_i(\alpha_i,t)$. This is because, during this interval of time, the scheduler may still be serving packets that arrived during a previous busy period. $Sent_i(\alpha_i,t)$, therefore, is not necessarily the same as the total number of bits scheduled from flow i in this interval. We are now prepared to present the definition of latency in *LR* servers.

Definition 3.13 *The latency of a flow is defined as the minimum non-negative constant Ω_i that satisfies the following for all possible busy periods of the flow,*

$$S_i(\alpha_i,t) \geq \max\{0, \rho_i(t-\alpha_i-\Theta_i)\} \tag{3.37}$$

As defined in [82], a scheduler which satisfies Equation (3.37) for some non-negative constant value of Θ_i is said to belong to the class of *Latency Rate* (*LR*) servers. The above definition captures the fact that the latency of a guaranteed-rate scheduler should not merely be the time it takes for the first packet of a flow to get scheduled, but should be a measure of the cumulative time that a flow has to wait until it begins receiving service at its guaranteed rate. A graph of $S_i(\alpha_i,t)$ against time is plotted in Figure 3.7. The RHS of the above equation defines an envelop which bounds the minimum service received by a flow i during the busy period (α_i,t). The dashed line in Figure 3.7 corresponds to this envelop. For a particular scheduling algorithm several parameters such as its transmission rate on the output link, the number of the other flows sharing the link and their reserved rate may influence the latency. It has been proved in [82] that the maximum end-to-end delay experienced by a packet in a network of schedulers can be calculated from only the latencies of the individual schedulers on the path of the connection and the traffic parameters on the connection that generated the packet. Since the end-to-end delay increases directly in proportion to the latency of the schedulers, the model highlights the significance of using low-latency schedulers for achieving low end-to-end delays.

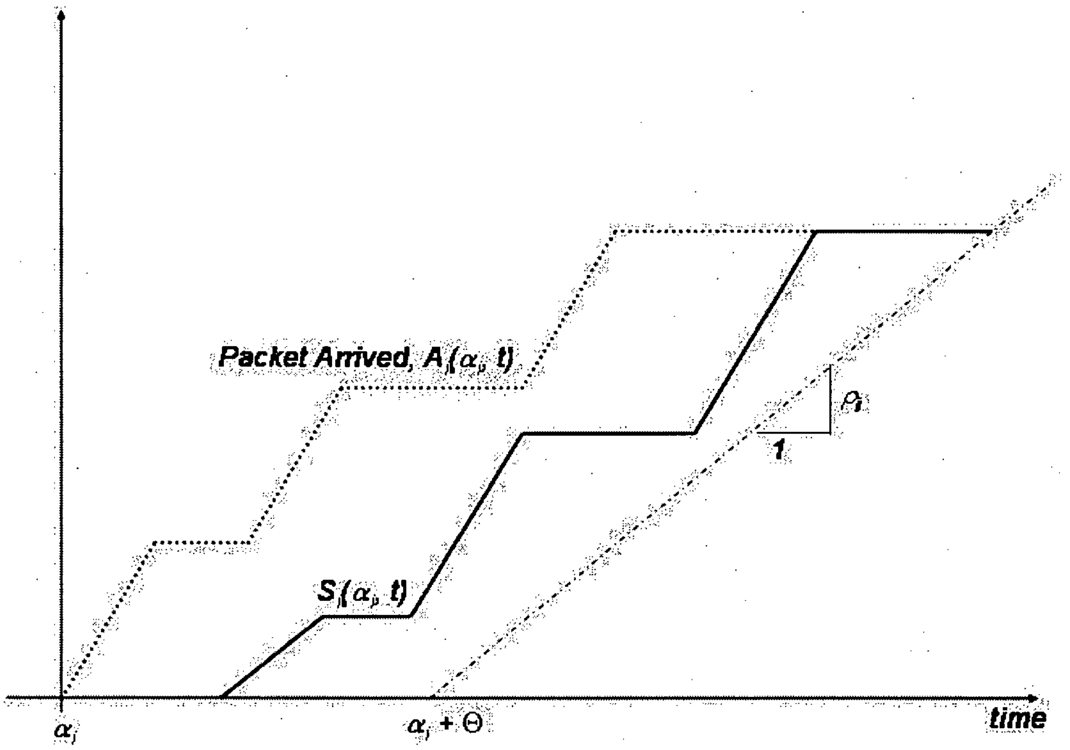

Figure 3.7 An example of the behaviour of an *LR* server

In [82], the following Lemma is also proved and is one that will allow us to easily analyze the latency of TBFQ. This result allows us to obtain a bound on the latency achieved by a flow as given by **Definition 3.13** by considering only the active periods of a flow.

Lemma 3.3 *Let τ_i be an instant of time when flow i becomes active. Let $t > \tau_i$ be some instant of time such that the flow is continuously active during the time interval (τ_i, t). Let Θ_i be the smallest non-negative number such that the following is satisfied for all t.*

$$Sent_i(\alpha_i, t) \geq \max\{0, \rho_i(t - \tau_i - \Theta_i')\} \qquad (3.38)$$

Even though (τ_i, t) may not be a continuously busy period for flow i, the latency as defined by **Definition 3.13** *is bounded by Θ_i'.*

For TBFQ, the definition of a busy period and an active period can be used interchangeably due to ***Lemma*** **3.1**, which stated that a flow will be serviced at a guaranteed minimum rate that is equivalent to the token rate, r_i. The token rate r_i is the same as ρ_i in Lemma **3.3** . We can proceed to determine the latency bound of the TBFQ scheduler. The latency bound is then the time when flow *i* becomes active and when a flow *i* begins to receive service from the TBFQ scheduler. The instant of time when a flow *i* becomes active, τ_i, may or may not coincide with token generation rate. In the following, we prove that a tighter upper bound on the latency of the TBFQ scheduler can be obtained by considering τ_i belonging to only a subset of all possible instants. This subset is the set of all time instants that coincide with its token generation rate.

Definition 3.14 *Define* $\mathbf{T}_i$ *as the set of all time instants,* $\{t_{i,0}, t_{i,1}, t_{i,2}, \ldots\}$ *at which the token pool for flow i becomes full, and* $\Delta t_i = t_{i,k} - t_{i,k+1}$*, for k={0, 1, 2, ...}.*

Lemma 3.4 *The latency experienced by flow i in an TBFQ scheduler will reach its upper bound* $\Theta_i^{'}$*, only if the time instant,* τ_i*, at which flow i becomes active, coincides with a time instant in* $\mathbf{T}_i$*.*

Proof: Assume that flow *i* becomes active at time instant τ_i. Let $t_{i,1} \le \tau_i < t_{i,2}$, where $t_{i,1}$ and $t_{i,2}$ belong to the set $\mathbf{T}_i$. Consider the case when τ_i does not coincide with $\mathbf{T}_i$ i.e., $\tau_i > t_{i,1}$. The time interval $(t_{i,1}, \tau_i)$ will not contribute to the service latency experienced by flow *i*. The latency will be $(\tau_i, t_{i,2})$. On the other hand, consider the case when τ_i coincides with $t_{i,1}$. Now, the time for which flow *i* has to wait before receiving any service will include the entire time interval (t_1, t_2). Clearly, the latency experienced by flow *i* is always greater when τ_i coincides with $\mathbf{T}_i$. The statement of the lemma follows from this observation. ■

Lemma **3.4** tells us that in order to reach the upper bound of $\Theta_i^{'}$, we only need to consider when τ_i coincides with $\mathbf{T}_i$.

Theorem 3.4 *The TBFQ belongs to the class of LR servers, and for any flow i, there is an upper bound on the latency given by,*

$$\Theta_i^{'} \le \frac{(n-1)L}{R} + \Delta t_i \qquad (3.39)$$

where n is the total number of flows in system, L is the size of a packet, R is the output link transmission rate, and $\Delta t_i = \dfrac{1}{r_i}$.

Proof: Consider when flow i becomes active at time τ_i, where $t_{i,k} \le \tau_i < t_{i,k+1}$, where $t_{i,k}$ and $t_{i,k+1}$ belong to the set **T**$_i$. One way to prove the statement of this theorem is to use the approach developed in [82] which determines the equation $Sent_i(\alpha_i, t)$ over the period (τ_i, τ_k) which is the period of many rounds of service, and then arrange the equation in such a form that it becomes $\rho_i(t - \tau_i - \Theta_i^{'})$ and then extract the $\Theta_i^{'}$ from the equation. However, since the reserved rate is guaranteed (from ***Lemma* 3.1**), we will only need to determine the time between when the flow becomes active and when the service begins. According to the definition of *LR* servers, this time difference is the latency bound.

As determined in Lemma **3.4**, we only need to consider a limited number of time instants, i.e. when τ_i coincides with the set **T**$_i$. We know that the members in the set {**T**$_1$, **T**$_2$, ..., **T**$_n$} are independent from each other. We also know that,

$$\begin{aligned} & t_{i,k} + \Delta t_i = t_{i,k+1} \\ & \Delta t_i = \frac{1}{r_i} \\ & \forall k = \{0,1,2,...\}, \forall i = \{1,2,...,n\} \end{aligned} \tag{3.40}$$

As a corollary to ***Lemma* 3.4**, it can easily be seen that $\Theta_i^{'}$ reaches the largest bound when τ_i coincides with all of {**T**$_1$, **T**$_2$, ..., **T**$_n$}, i.e. when

$$\begin{aligned} & t_{1,0} = t_{2,0} = ... = t_{n,0} \\ & \Delta t_1 = \Delta t_2 = ... = \Delta t_1 \end{aligned} \tag{3.41}$$

When this occurs, flow i could wait for the other *n-1* flows to receive service first before its turn. The amount of time to serve the *n-1* flows at the outgoing rate of *R*, would simply be

$$\frac{(n-1)L}{R} \tag{3.42}$$

The latency becomes,

$$\frac{(n-1)L}{R} + (t_{i,k+1} - \tau_j) \tag{3.43}$$

Therefore, the latency bound is

$$\Theta_i^{'} \le \Delta t_i \tag{3.44}$$

■

From ***Theorem* 3.4**, we know that when the tokens generation periods of each flow coincide with each other, the system would result in the maximum latency. By having tokens generation perfectly interleaved among the flows, we minimize the latency.

***Corollary** 3.1 When the token generation periods interleaved perfectly among the flows, the latency is given by,*

$$0 \le \Theta_i^{'} \le \Delta t_i$$

where $\Delta t = \frac{1}{r_i}$.

Proof: Consider when flow i becomes active at time τ_i, where $t_{i,k} \le \tau_i < t_{i,k+1}$. When the token generation periods for flows do not coincide with each other, i.e.

$$\begin{gathered} t_{1,0} \ne t_{2,0} \ne \ldots \ne t_{n,0} \\ \Delta t_1 = \Delta t_2 = \ldots = \Delta t_1 \end{gathered} \tag{3.45}$$

and $\mathbf{T}_i \not\subseteq \{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_n\}$, then service can be received starting at $\{t_{i,1}, t_{i,2}, t_{i,3}, \ldots\}$. So the maximum latency that can be experienced is when $\tau_i = t_{i,k}$, then flow i will have to wait until $t_{i,k+1}$ before it can start to receive service. Therefore,

$$\Theta_i^{'} \le \Delta t$$

Furthermore, it is possible that the token pool is full when flow i becomes active. In that case, the flow is served immediately and $\Theta_i^{'} = 0$. The corollary is proved. ■

3.6 Graceful services

As we explained earlier, TBFQ has a *soft* handling of violating traffic. We will demonstrate this through simulation in the next chapter. In this section, we would like to determine the kind of service the end-user would perceive when a flow exits from erroneous condition. As a flow has been receiving *lagging* service during the period when the channel condition is erroneous, when it exits from such condition to an error-free one it should receive services that will gradually return to its reserved service rate. This gradual returning to normal service, as oppose to an abrupt manner which would cause significant fairness issue, would take finite amount of time. We shall call this the *grace period* which can be used to quantify user perception of the service. The grace period is a measure of how quickly the service can return to its norm, assuming error-free condition is maintained throughout the period.

Definition 3.15 *A flow i is said to be leading in terms of the services received when its priority* $P_i < 0$. *It is lagging when* $P_i > 0$. *And when* $P_i = 0$, *the service is at its norm.*

Definition 3.16 *When a lagging flow i is experiencing channel errors, such as in location-dependent error conditions, and it comes out of such condition to an error-free condition at* τ_e. *An error-free condition is one where the flow can enjoy a normal rate of service at least at the reserved rate or better. We define grace period, T_G, as* $\tau_e' - \tau_e$, *where* τ_e' *is the time when the service received by the flow i reaches its norm, i.e. neither lagging nor leading.*

Lemma 3.5 *In an error-free condition, if there are extra resources in the system, lagging flows will receive the additional resources first before the leading flows. And leading flows will only be given the additional resources when the lagging flows have no packets to be served.*

Proof: Assuming the system is continuously backlogged. Let $\Psi_>$ be the set of lagging flows $\{i \in \Psi_> \mid P_i > 0\}$; $\Psi_<$ be the set of leading flows $\{j \in \Psi_< \mid P_j < 0\}$, and $\Psi_=$ be the flows that are at their norm $\{k \in \Psi_= \mid P_k = 0\}$. By virtue of the algorithm as stated in equation (3.3), excess resources will first be given to flow i_1 first where $P_{i_1} > P_{i_2} > \ldots > P_{i_n}$ and $i_1 \in \Psi_>$. It is easy to see that as long as $\{\Psi_>\} \neq \{\,\}$ and the flows in $\Psi_>$ are continuously backlogged, excess resources will be given to flows in $\Psi_>$. The only case when flows in both $\Psi_>$ and $\Psi_=$ are not being served is when their queues are empty, therefore the excess resources can be given to leading flows in $\Psi_<$. ■

Theorem 3.5 *Assuming that all flows continue to be backlogged in the system during the grace period and all flows, with the exception of flow i, are in their error-free condition. When the lagging flow i has just came out of a channel error condition into a condition that is considered error-free at* τ_e, *the grace period* T_G *experienced by flow i is bounded by,*

$$T_G \leq \frac{\sum_{j=1}^{m} E_j}{(R - \sum_{j=1}^{m} r_j)} \tag{3.46}$$

where there are *m* lagging flows, *r* is the respective reserved rate, *E* is the token balance counter, and *R* is the link output rate,. Assuming all flows stay error-free for the duration of the grace period.

Proof: Let us first consider the priority index of the flows at τ_e. They are,

$$
\begin{aligned}
&P_1(\tau_e) = E_1(\tau_e)/r_1\\
&\vdots\\
&P_{i-1}(\tau_e) = E_{i-1}(\tau_e)/r_{i-1}\\
&P_i(\tau_e) = E_i(\tau_e)/r_i\\
&P_{i+1}(\tau_e) = E_{i+1}(\tau_e)/r_{i+1}\\
&\vdots\\
&P_n(\tau_e) = E_n(\tau_e)/r_n
\end{aligned}
\tag{3.47}
$$

Let $\Psi_>$ be the set of m lagging flows where $i \in \Psi_>$, $\Psi_<$ be the set of leading flows, and $\Psi_=$ be the flows that are at their norm. The grace period T_G is $\tau_e' - \tau_e$, where at τ_e', $i \in \Psi'$.

T_G reaches its maximum if flow i is the last flow that reaches $\Psi_=$, i.e. $\Psi_< = \{\}$. Since from Lemma 3.5, we only need to consider flows in $\Psi_>$. Let the flows in the set $\Psi_>$ be $\{f_1, ..., f_m\}$ with their respective priorities $\{P_{f_1}, ..., P_{f_m}\}$ where $P_{f_1} > ... > P_{f_m}$. Therefore,

$$
\begin{aligned}
&P_{f_1} = \frac{E_{f_1}}{r_{f_1}}\\
&\vdots\\
&P_{f_m} = \frac{E_{f_m}}{r_{f_m}}
\end{aligned}
\tag{3.48}
$$

In order for all flows in $\{f_1, ..., f_m\}$ to reach $\Psi_=$, where,

$$
\begin{aligned}
&E_{f_1} = 0\\
&\vdots\\
&E_{f_m} = 0
\end{aligned}
\tag{3.49}
$$

the system has to utilize extra bandwidth to serve an extra amount of packets equivalent to $\sum_{j=1}^{m} E_{f_j}$. This extra bandwidth is

$$(R - \sum_{j=1}^{m} r_{f_j}) \tag{3.50}$$

Therefore, the extra amount of packets served during $(\tau_e' - \tau_e)$ is,

$$\sum_{j=1}^{m} E_{f_j} = (R - \sum_{j=1}^{m} r_{f_j})(\tau_e' - \tau_e) \tag{3.51}$$

So,

$$(\tau_e{}'-\tau_e)=\frac{\sum_{j=1}^{m}E_{f_j}}{(R-\sum_{j=1}^{m}r_{f_j})} \tag{3.52}$$

Since this is the worst case scenario, the grace period is therefore bounded by,

$$T_G \leq \frac{\sum_{j=1}^{m}E_{f_j}}{(R-\sum_{j=1}^{m}r_{f_j})} \tag{3.53}$$

■

From this theorem, the lower limit of the grace period can be easily deduced.

***Corollary** 3.2 Assuming that all flows continue to be backlogged in the system during the grace period and all flows, with the exception of flow i, are in their error-free condition. When the lagging flow i has just came out of a channel error condition into a condition that is considered error-free at τ_e, the grace period T_G experienced by flow i is at least,*

$$T_{G\min} \geq \frac{\sum_{j=1}^{m}E_j - mL + 1}{(R-\sum_{j=1}^{m}r_j)} \tag{3.54}$$

Proof: In ***Theorem* 3.5**, the upper bound of the grace period is found when flow *i* is the last flow to reach $\Psi_=$. The lower bound is found if flow *i* is the first flow to reach $\Psi_=$.

Before all flows in $\{f_1,\ldots,f_m\}$ can reach $\Psi_=$, their priorities must first reach the levels,

$$\begin{aligned} P_{f_1} &= \frac{L}{r_{f_1}} \\ &\vdots \\ P_{f_m} &= \frac{L}{r_{f_m}} \end{aligned} \tag{3.55}$$

Which means the extra services received by the extra bandwidth must be,

$$\sum_{j=1}^{m}E_j - mL \tag{3.56}$$

If flow i is the first flow to reach $\Psi_{=}$, then the next time, at time τ_e'', when the extra service becomes available, the total extra services received would be,

$$\sum_{j=1}^{m} E_j - mL + 1 \tag{3.57}$$

And,

$$\sum_{j=1}^{m} E_{f_j} - mL + 1 = (R - \sum_{j=1}^{m} r_{f_j})(\tau_e'' - \tau_e) \tag{3.58}$$

Therefore, the lower bound grace period is $(\tau_e'' - \tau_e)$, and,

$$T_{G\min} \geq \frac{\sum_{j=1}^{m} E_j - mL + 1}{(R - \sum_{j=1}^{m} r_j)} \tag{3.59}$$

■

Chapter 4 MODELING OF TBFQ

4.1 Challenges in mathematical modeling of TBFQ

The most direct and natural mathematical modeling of TBFQ would lead to a discrete time stochastic process with multi-dimensional state space. However, this model cannot be mathematically tractable for any possible theoretical analysis leading to exact performance evaluation

A number of challenges in this mathematical model exist, among which the most significant ones are (a) multi-dimensionality in state space; (b) dependence between queues; and (c) non-Markovian property in the process. With all these three properties present in the model, there exist no approaches available in the literature such that a joint distribution for the performance could possibly be expressed explicitly.

In order to set up a more tractable mathematical model, we first analyze where the three above mentioned challenges come from.

(1) The property of multi-dimensionality is a direct consequence of studying integrated wireless systems with multi-users. Moreover, multimedia applications in current and future wireless systems result in a multi-dimensional stochastic system with heterogeneous traffic types. This is one of the most important natures in the current and especially in the next generation packet-switched wireless networks. It would make the mathematical model unrealistic and non-attractive if such a nature were dropped.

(2) The dependence is also an immediate consequence of the TBFQ. The concept of the fairness characterized in TBFQ was introduced to address the long-term fairness issue in bandwidth assignment among all wireless connections. Without dependence, this can be hardly achieved. Any modification towards independence in modeling would lead to an undesired algorithm.

(3) The non-Markovian property in the original TBFQ is the consequence of a couple of assumptions in modeling: (a) traffic models, which means non-Markovian properties of packet generating processes; and (b) the introduction of the *creditable threshold* parameter. Each time when the *debt limit* d is reached, no token can be borrowed by the connection until the token balance of the connection has been increased to the level of the *creditable threshold*. Therefore, the future assignment of the available bandwidth, or future state of the system, not only depends on the current state of the system, but also the state information in the past to determine if any bandwidth could be borrowed by a connection whose token balance is above its *debt limit* but below its *creditable threshold*. The supplementary variable method [83] is the common technique to convert a non-Markovian model into Markovian one. To make

the system Markovian, one or more supplementary variables must be introduced into the model, which will add further complexity into the already very complicated one.

(4) Besides the above mentioned challenges, there exists other complexities in the original TBFQ, including the *burst credit*, the token bank size B, and others.

4.2 Mathematical modeling

In this section, a mathematical model for a family of scheduling schemes is defined, which includes the TBFQ scheduling algorithm as its member. The mathematical model is a discrete time stochastic system, which will be referred to as the Fairness Family of Scheduling Algorithms (FaSA). In this system, time is divided into slots [k −1, k) such that the mathematical model is used to represent frame-based scheduling algorithms. Various scheduling algorithms can be derived from FaSA by specifying the fairness function introduced in the process. All of these algorithms bear a principle to address the long term fairness in allocating bandwidth, the same as in the original generic TBFQ described in chapter 3.1. TBFQ is a subset of the FaSA.

Arrival process: The packet generating pattern of all wireless connections is characterized by a stochastic process, referred to as the arrival process. To define this process, let x be an index variable, which takes a value of a, d or v, representing voice, data or video traffic. For each value of x, assume that n_x is a finite positive integer. Therefore, n_a, n_d or n_v is the total number of wireless connections requiring voice, data or video type of service. The total number of wireless connections in the system is $N = n_a + n_d + n_v$. For a fixed value of x, let $WT_x^i(k)$ be a discrete time stochastic process, representing the number of total packets generated by the i-th wireless terminal in type x service group in the time interval [0, k). We assume that all $WT_x^i(k)$ are independent, each of which either is a Markov chain or can be converted in terms of supplementary method to a Markov chain. This is a reasonable model since wireless connections can often be assumed to independently generate their packets, and the traffic model is either Markovian or can be converted into Markovian in terms of the supplementary variable method. Usually, all processes $WT_x^i(k)$ belonging to the same type of traffic can be assumed to be identically distributed to reflect the fact the same type of connections could generate the same type of packets. However, we will not introduce this restriction in the definition of FaSA. The packets generated by each process $WT_x^i(k)$ are put in its own data buffer for competing service resources of the system in the next time slot with other service requirements generated by other connections. The size of each data buffer is assumed to be infinite. For a fixed value of x, packets generated from all connections with the same type of traffic are assumed to have a common packet size L_x. Packet generated from a different type of traffic may have a different packet size. Packets generated by the same type of connections are served according to FIFO queueing discipline.

Queue length process: The queue length process is used to denote the number of packets in each data buffer observed at time epoch *k*. This number includes the possible new arrivals during the time slot or frame [*k* – 1, *k*) and any backlogged packets in the buffer, which have not been transferred in frame [*k* –1, *k*). The queue length process is one of the fundamental stochastic quantities, which often plays a key role in the analysis of the system. Let $Q_x^i(k)$ be the number of packets in the data buffer for the *i*th wireless connection of *x* type of traffic at time epoch *k*. If the number of packets generated by WT_x^i in slot [*k* – 1, *k*) is denoted by $\lambda_x^i(k)$ and the bandwidth (including the basic assignment and the extra borrowing), measured in the number of packets per frame, allocated to connection *i* of *x* type of traffic in frame [*k* – 1, *k*) is denoted by $\mu_x^i(k)$, then the queue length process can be expressed as

$$Q_x^i(k)=\left[Q_x^i(k-1)-L_x^i\mu_x^i(k)+\lambda_x^i(k)\right]^+ \tag{4.1}$$

where $[c]^+=\max(c,0)$. To model the fairness in allocating resources, we need to assume that $\mu_x^i(k)$ depends on the information about other wireless connections.

Token generation rate and token balance process: The definition for the token generation rate and the token balance process was motivated by the fairness mechanism described in the TBFQ scheduling algorithm. TBFQ aims at the fairness of scheduling resources to wireless connections in longer term instead of shorter term. There are two basic components in TBFQ service: (a) guaranteed minimum bandwidth assigned to each connection that requires transmitting packets; and (b) excess bandwidth assignment based on long term fairness principle. Mathematically, the token generation rate r_x^i, measured in packets per frame instead in bytes per frame in the original TBFQ model, is assumed to be a constant for each fixed connection *i* in the fixed type *x* of traffic. For a connection of a different type of traffic or even for a different connection in the same type of traffic, the token generation rate would be different, which models the difference in the QoS requirement by each wireless connection. To achieve fairness in allocating excess resources, for each connection, the information about its cumulative usage of bandwidth over time must be recorded. Scheduling of the available bandwidth depends on this information. Mathematically, we introduce the token balance process $E_x^i(k)$ for each connection, which denotes the token balance in packets observed at time epoch *k*. The cumulative token balance at time epoch *k* for a connection can be calculated based on the cumulative balance at time epoch *k* – 1 and the usage of resource in time slot [*k* – 1, *k*). It increases by r_i^x in each time slot and depletes L_i^x bytes if a packet has been transmitted in slot $[k-1, k)$. Therefore, the token balance process for a connection can be expressed as

$$E_x^i(k) = \left[E_x^i(k-1) + r_x^i - L_x^i \mu_x^i(k)\right]^+ \tag{4.2}$$

Resource process: Since the total bandwidth in a frame is usually a constant, we use B to denote total available bytes in a frame. The total extra available resources, after the minimum bandwidth assignment to all connections, available for borrowing in each frame is not a constant since the requirement in service by the connections varies from frame to frame, which is modeled as the resource process. Define $B(k)$, to be the total extra resources for borrow measured in bytes in time slot $[k-1, k)$ available for borrow by connections. If $B \geq \sum_{i,x} L_x^i r_x^i$ is assumed, then $B(k)$ can be expressed as

$$B(k) = B - \sum_{i,x} \theta_x^i(k). \tag{4.3}$$

Obviously, $B(k)$ depends on the information about all queue lengths at time epoch $k-1$.

Fairness function: As we already mentioned earlier, the fairness allocation of extra bandwidth not only depends on the token balance process $E_x^i(k)$ for all connections, but also on other parameters, such as the debt limit d_x^i and the creditable threshold t_x^i for each connection. Since each time when the token balance process $E_x^i(k)$ has reached its debt limit d_x^i, often assumed to be a negative value, the connection will not be allowed to borrow until the token balance reaches the creditable threshold, the allocation of the extra resources have to depend on history of the token balance. This destroys the Markovian property in the mathematical modeling. To make the FaSA Markovian, the same supplementary variable method can be used here, defined as the token status process:

$$S_x^i(k) \begin{cases} 0, & \text{if the connection is not allowed to borrow in slot (k,k+1],} \\ 1, & \text{if the connection is allowed to borrow in slot (k,K+1].} \end{cases} \tag{4.4}$$

In other words, $S_x^i(k) = 0$ if and only if (i) $S_x^i(k-1) = 0$ and $E_x^i(k) < t_x^i$, or (ii) $S_x^i(k-1) = 1$ and $E_x^i(k) \leq d_x^i$. Alternatively, $S_x^i(k) = 1$ if and only if (i) $S_x^i(k-1) = 0$ and $E_x^i(k) \geq t_x^i$, or (ii) $S_x^i(k-1) = 1$ and $E_x^i(k) > d_x^i$.

In terms of this supplementary variable, we can define $\theta_x^i(k)$ as a Markovian function, called the fairness function, which is a stochastic process, depending on the values of token generation rate, the queue length and the token balance for all connections. For convenience, we introduce the following vectors:

$$\vec{r} = (r_x^i)_{\{x=a,d,v;i=1,2,\ldots,n_x\}}, \tag{4.5}$$

$$\vec{E}(k-1) = (E_x^i(k-1))_{\{x=a,d,v;i=1,2,\ldots,n_x\}}, \tag{4.6}$$

$$\vec{Q}(k-1) = (Q_x^i(k-1))_{\{x=a,d,v;i=1,2,\ldots,n_x\}}, \tag{4.7}$$

For every fixed connection i from a fixed type x, the fairness function $\theta_x^i(k)$ is a function, measured in packets per frame, of $\vec{r}$, $\vec{E}(k-1)$, $\vec{Q}(k-1)$, the token status process $S_x^i(k-1)$, and the total extra available resource $B(k)$. Since the token status process $S_x^i(k-1)$ is a function of the token balance process $E_x^i(k-1)$, we can write

$$\theta_x^i(k) = f\left(\vec{r}, \vec{E}(k-1), \vec{Q}(k-1), B(k)\right). \tag{4.8}$$

Hence, the token balance process can be expressed as

$$E_x^i(k) = E_x^i(k-1) + r_x^i - f\left(\vec{r}, \vec{Q}(k-1), \vec{E}(k-1), B(k)\right) \tag{4.9}$$

A specific definition of the function f determines a specific scheduling algorithm. The original TBFQ algorithm is derived when the fairness function $\theta_x^i(k)$ is defined as follows. At each time k, let all connections be indexed according to the descending order of the priority index value defined as $E_x^i(k)/r_x^i$, say $i_{(1)}, i_{(2)}, \ldots, i_{(N)}$. The fairness function for the connection with the highest priority index is defined first, and the fairness function for all other connection are defined recursively as: for $i_{(1)}$,

$$\theta^{i_{(1)}}(k) = S^{i_{(1)}}(k-1)\min\left\{\max\left(0, Q^{i_{(1)}}(k-1) - r_x^{i_{(1)}}\right), b^{i_{(1)}}, B(k)/L_x^{i_{(1)}}\right\}, \tag{4.10}$$

where $b^{i_{(1)}}$ is the burst credit for connection $i_{(1)}$, and for $j = 2, \ldots, N$,

$$\theta^{i_{(j)}}(k) = S^{i_{(j)}}(k-1)\min\left\{\max\left(0, Q^{i_{(j)}}(k-1) - r_x^{i_{(j)}}\right), b^{i_{(j)}}, \left(B(k) - \sum_{l<j}\theta^{i_{(l)}}(k)L^{i_{(l)}}\right)/L_x^{i_{(j)}}\right\}. \tag{4.11}$$

4.3 Methodologies

4.3.1 Approximating the service process

The idea is to approximate/simplify the multi-dimensional dependent process into a number of one-dimensional processes, which can be analyzed one at a time. Consider the queue length process defined earlier

$$Q_x^i(k) = [Q_x^i(k-1) - L_x^i \mu_x^i(k) + \lambda_x^i(k)]^+ \tag{4.12}$$

According to the assumption that $WT_x^i(k)$ are independent, each of which either is a Markov chain or can be converted in terms of supplementary method to a Markov chain, $Q_x^i(k)$ is a Markov chain, either immediately or in terms of supplementary variables if $\mu_x^i(k)$ are independent of *i, x* and *k*. Unfortunately, this is not the case. It might not be a good idea to decouple the FaSA with dependence among connections into *N* independent single connection models, since the fundamental fairness principle would not be effectively captured in the simplified model. And since the fairness function can be recursively determined one by one, we may recursively analyze all queue lengths one by one. However, this can be done for the analysis of the transient queue behaviour, but cannot be easily implemented in stationary analysis. The key step in this method is to develop an approximation for the service process such that the stationary queue behaviour can be analyzed one by one.

The advantage of this is many well-developed approaches could be used to analyze each approximating queue process independently. The challenge is to introduce a new mechanism to integrate all the demands from all connections from all different traffic types into one service variable, which is i.i.d. for all values of k. The total bandwidth constrain in a frame has to be reflected in the service model. Furthermore, it is possible that the approximation can be justified by statistical fitting of field data to the model.

4.3.2 Precedence relation method

This method was developed by [104][105] to provide both lower and upper bounds of a multi-dimensional dependent Markov chain, which reveal a special type of property, called precedence relation. It has been proved that if the system possesses the precedence relation, one can construct a lower bound model and an upper bound model of the original one with any desired precision. This may lead to an approximation model, which is mathematically tractable for theoretical analysis. Analysis for one-dimensional independent queues is well-documented, and various techniques could be used to reduce modeling error. Although for the system that possesses the precedence relation, tight bounds, both lower and upper, can be obtained, not all systems can have the precedence relation. It is often very challenging to confirm the existence of such a property.

4.4 Network calculus analysis

NetCal analysis [6] provides the worst case analysis and provides upper bounds for the performance. Instead of analysis involving random variables or stochastic processes, it is a deterministic method. We expected that this method can also be used in TBFQ model. However, we can not directly apply NetCal to the traffic models mentioned earlier. A leaky bucket process has to be applied for obtaining arrival curve and the service curve. Although this method gives deterministic analysis, the disadvantages of using network calculus analysis method are:

- The original TBFQ model will be modified, the modification would not be realistic in wireless access;
- The bound will not be tight enough, which would not provide useful performance results.

4.5 Recursive approach

The recursive approach is a technique novel to the analysis of scheduling algorithms, and it allows us to provide much tighter bound. In the following, the recursive approach implementation is demonstrated in detail and the results are shown.

The queue length process

$$Q_x^i(k) = [Q_x^i(k-1) - L_x^i \mu_x^i(k) + \lambda_x^i(k)]^+ \tag{4.13}$$

given in (4.1) determines the average packet delay and the throughput of each wireless connection. We are interested in determining the probability distribution of the queue length in equilibrium or in steady-state. Mathematically, we want to determine

$$\pi_x^i(n) = \lim_{k\to\infty} P(Q_x^i(k) = n), \qquad n = 0,1,\ldots \tag{4.14}$$

for each wireless terminal i in the type x connection.

We assume that for a fixed type of connection x, all the terminals behave the same way to each other, that means they have the same packet generation rate, the same bandwidth allocation mechanism, the same buffer size, and etc. This is a reasonable assumption and under this assumption, we have the following property.

Property 4.1 *For a fixed type of connection x, $\mu_x^i(k)$ has the same distribution for all wireless terminals i. Moreover, for a fixed wireless terminal i in a fixed type of connection x, $\mu_x^i(k)$ are i.i.d. random variables for all k.*

According to **Property 4.1**, the resulting queue length process determined by (4.1) is a standard *GI/GI/*1 queue. If the service variable in Property 4.1 can be determined, then we can numerically determine the stationary queue length distribution $\pi_x^i(n)$ using existing algorithms given in the literature. Other average performance metrics, including the average queue length and the average packet delay, can be computed based on this distribution. The goal and challenge in the analysis of TBFQ is determining the service variable under the fairness principle.

The recursive approach proposed here is to compute the queue length distribution recursively. This procedure starts from a lower bound of $\mu_x^i(k)$ and then update it recursively until a satisfactory accuracy in estimating performance metrics has been reached. To update $\mu_x^i(k)$, we need to specify the fairness function $\theta_x^i(k)$ and the updated token balance process $E_x^i(k)$. In general,

$$\theta_x^i(k) = f\left(\vec{r}, \vec{Q}(k-1), \vec{E}(k-1), B(k)\right)$$

is a function of many variables, including the token balance process $\vec{E}(k)$, which will be a very complex function.

In order for the mathematical model to be practically tractable, we need to simplify the fairness function and at the same time we expect that the simplified version still captures the property implied by the TBFQ principle. Various versions of the simplification can be proposed to present variations in the fairness principle. The following are two examples.

Simplification 1 (Random Allocation) For a fixed wireless terminal i in a fixed type of connection x, the fairness function is defined as

$$\theta_x^i(k) = \begin{cases} B(k)/L_x^i \ \textit{with probability } 1/m_k, & \textit{if } E_x^i(k-1) = \max_{j,y} E_y^j(k-1) \\ 0 & , \textit{if } E_x^i(k-1) < \max_{j,y} E_y^j(k-1) \end{cases} \tag{4.15}$$

where m_k is the number of terminals with the largest token balance.

Simplification 2 (Uniform Allocation) For a fixed wireless terminal i in a fixed type of connection x, the fairness function is defined as

$$\theta_x^i(k) = \begin{cases} B(k)/L_x^i m_k & , \text{if } E_x^i(k-1) = \max_{j,y} E_y^j(k-1) \\ 0 & , \text{if } E_x^i(k-1) < \max_{j,y} E_y^j(k-1) \end{cases} \tag{4.16}$$

where m_k is the number of terminals with the largest token balance.

The numerical procedure for computing the stationary queue length probability distribution is given in the following.

4.6 Implementation of the Recursive Approach

In this subsection, we provide details on the implementation of the introduced recursive approach.

Step 0: In this step, we find a lower bound $\tilde{\mu}_x^i(k)$ to the bandwidth $\mu_x^i(k)$ allocated to the connection *i* of type *x* traffic in frame [*k* – 1, *k*), where $\mu_x^i(k)$ is measured in packets per frame including both the basic assignment and the extra borrowing. Various lower bounds for the service rate can be proposed. Here, we provide one of them.

One simple lower bound can be obtained by not allowing borrow. In this case,

$$\tilde{\mu}_x^i(k) = \begin{cases} 1, & \text{with probability } r_x^i / L \\ 0, & \text{with probability } 1\text{-}(r_x^i / L) \end{cases} \tag{4.17}$$

We assume that the packets generated from all connections have the same size $L = L_x^i$ [bytes].

Step 1: When the bandwidth allocation $\mu_x^i(k)$ is given, the queue length process

$$Q_x^i(k) = \left[Q_x^i(k-1) - L_x^i \mu_x^i(k) + \lambda_x^i(k)\right]^+ \tag{4.18}$$

given in (4.1) provides us a discrete time *GI/GI/*1 queue, either directly or through introducing supplementary variable(s) (for example, referring to Cox and Miller [106]) depending on the traffic model. For each of the possible traffic models, we need to formulate expressions for computing the transition matrix for the queue length process and implement the matrix as a separate module. When the transition probability matrix ***P*** becomes available, the stationary queue length distribution

$$\pi = (\pi_0, \pi_1, \pi_2, \ldots) = (\pi_x^i(0), \pi_x^i(1), \pi_x^i(2), \ldots) \tag{4.19}$$

can be determined by solving the matrix equation $\pi = \pi P$ and the normalization condition $\sum_n \pi_n = 1$.

In general, we cannot expect any explicit formula for the distribution. Computational methods should be employed to obtain numerical numbers. For this purpose, we use two numerical algorithms, one for computing the rate matrix, [108], and the other for computing the boundary probability vectors using the state reduction method or the GTH algorithm introduced by Grassmann, Taksa and Hayman [107].

The first algorithm is for computing the rate matrix **R** of the *GI/M/*1 type of matrix, partitioned according to blocks or levels, given by

$$\mathbb{P}_V = \begin{array}{c|cccccccccc} & 0 & 1 & 2 & \cdots & \text{n-1} & \text{n} & \text{n+1} & \text{n+2} & \cdots & \\ 0 & \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & & & & & & & & \\ 1 & \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & & & & & & & \\ \vdots & \vdots & \vdots & \ddots & \ddots & & & & & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & & & & & \\ n-1 & \mathbf{P}_{n-1,0} & \mathbf{P}_{n-1,1} & \mathbf{P}_{n-1,2} & \cdots & \mathbf{P}_{n-1,n-1} & \mathbf{P}_{n-1,n} & & & & \\ n & \mathbf{P}_{n,0} & \mathbf{C}_n & \mathbf{C}_{n-1} & \cdots & \mathbf{C}_2 & \mathbf{C}_1 & \mathbf{C}_0 & & & \\ n+1 & \mathbf{P}_{n+1,0} & \mathbf{C}_{n+1} & \mathbf{C}_n & \cdots & \mathbf{C}_3 & \mathbf{C}_2 & \mathbf{C}_1 & \mathbf{C}_0 & & \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \ddots & \ddots & \end{array} \tag{4.20}$$

where n is a finite number and $\mathbf{P}_{i,j}$ and $\mathbf{C}_i$ are matrices of finite sizes.

Let $\boldsymbol{\pi} = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots)$ be the stationary probability vector of $\mathbb{P}_V$ partitioned according to the levels. Then,

$$\pi_{n+k} = \pi_n \mathbf{R}^k, \quad \text{for } k = 0,1,2,\ldots, \tag{4.21}$$

where **R** is the minimal non-negative solution of the matrix equation

$$\mathbf{R} = \mathbf{C_0} + \mathbf{C}_1\mathbf{R} + \mathbf{C}_2\mathbf{R}^2 + \cdots \tag{4.22}$$

Therefore, for computing the stationary probability vector $\boldsymbol{\pi}$, we need to compute **R** and the boundary probability vectors $\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots, \boldsymbol{\pi}_n$.

For computing the matrix **R**, let

$$\mathbf{C}_1^{(0)} = \mathbf{C}_0 + \mathbf{C}_1, \quad \mathbf{C}_i^{(0)} = \mathbf{C}_i, \quad i = 2,3,\ldots \tag{4.23}$$

Compute for k = 1, 2, . . . ,

$$\mathbf{C}_i^{(k)} = \mathbf{C}_i + \widehat{\mathbf{C}}_1^{(k-1)} \widehat{\mathbf{C}}_{i+1}^{(k-1)}, \quad i = 1,2,\ldots, \tag{4.24}$$

where $\widehat{\mathbf{C}}_1^{(k-1)} = \sum_{n=0}^{\infty}\left(\widehat{\mathbf{C}}_1^{(k-1)}\right)^n$ with $\left(\mathbf{C}_1^{(k-1)}\right)^0 = \mathbf{I}$. Then,

$$\mathbf{R} = \lim_{k\to\infty}\left(\mathbf{C_0}\widehat{\mathbf{C}}_1^{(k)}\right) \tag{4.25}$$

For computing the boundary probability vectors $\pi_0, \pi_1, \pi_2, ...\pi_n$, we first compute the so-called censored matrix

$$\mathbb{P}^{(n)} = \begin{array}{c} \\ 0 \\ 1 \\ \vdots \\ \vdots \\ n-1 \\ n \end{array} \begin{array}{c} \begin{array}{cccccc} \mathit{0} & \mathit{1} & \mathit{2} & \ldots & \mathit{n\text{-}1} & \mathit{n} \end{array} \\ \left(\begin{array}{cccccc} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & & & & \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & & & \\ \vdots & \vdots & \ddots & \ddots & & \\ \vdots & \vdots & \vdots & \ddots & \ddots & \\ \mathbf{P}_{n-1,0} & \mathbf{P}_{n-1,1} & \mathbf{P}_{n-1,2} & \cdots & \mathbf{P}_{n-1,n-1} & \mathbf{P}_{n-1,n} \\ \mathbf{P}_{n+1} & \mathbf{P}_{n} & \mathbf{P}_{n-1} & \cdots & \mathbf{P}_{2} & \mathbf{P}_{1} \end{array}\right) \end{array} \tag{4.26}$$

where

$$\mathbf{P}_i = \mathbf{C}_i + \sum_{k=1}^{\infty}\mathbf{R}^k\mathbf{C}_{k+i}, \quad i = 1, 2, ..., n \tag{4.27}$$

and

$$\mathbf{P}_{n+1} = \mathbf{P}_{n,0} + \sum_{k=1}^{\infty}\left(\mathbf{R}^k\mathbf{P}_{n+k,0}\right) \tag{4.28}$$

$\mathbb{P}^{(n)}$ is a stochastic matrix. For this matrix, we apply the GTH algorithm to the matrix for obtaining the boundary vectors $\pi_0^{'}, \pi_1^{'}, ..., \pi_n^{'}$, which differ from the boundary probability vectors by a constant c. Use

$$\boldsymbol{\pi}_{n+k}^{'} = \boldsymbol{\pi}_{n+k}^{'}\mathbf{R}, \quad \text{for } k = 1, 2, ..., \tag{4.29}$$

to compute all other non-boundary vectors, which differ from the non-boundary probability vectors by the same constant c. Finally, the constant c is computed according to

$$c = (\boldsymbol{\pi}_0^{'} + \boldsymbol{\pi}_1^{'} + \boldsymbol{\pi}_2^{'} + \cdots)\boldsymbol{e} \tag{4.30}$$

and

$$\boldsymbol{\pi}_i = \boldsymbol{\pi}_i^{'} / c \tag{4.31}$$

where $\boldsymbol{e}$ is a column vector of ones.

Mathematically, the GTH algorithm is equivalent to the Gaussian elimination, which can be used to compute the stationary distribution vector for any transition matrix of a finite matrix. Since the challenge in handling the size of the matrix $\mathbb{P}$ if the GTH algorithm is directly applied to it, we divide the computations into two components. The GTH algorithm is only applied to computing the boundary probability vectors. Basically, the algorithm works by eliminating states, one state at a time from the set of states in the model. In other words, from the initial transition probability matrix, a new transition probability matrix of the size reduced by one is obtained, which is called the stochastic complement of the initial matrix for a state. At each subsequent step, a new stochastic complement is calculated from that found in the preceding step. Finally, an upper triangular matrix is obtained and the system is solved.

To use the GTH algorithm, we assume that the size for the matrix $\mathbf{P}$ is $N + 1$ or the state space is $\{0, 1, \ldots, N\}$, and let $\boldsymbol{\pi} = (\pi_0, \pi_1, \pi_2, \ldots)$ be a stationary distribution of $\mathbf{P}$, i.e., $\boldsymbol{\pi}=\boldsymbol{\pi}\mathbf{P}$ and $\sum_i \pi_i = 1$. The GTH algorithm for obtaining $\boldsymbol{\pi}$ is as follows:

1. For $n = N, N - 1, \ldots, 1$, do the following:

   Let $S = \sum_{j=0}^{n-1} p_{n,j}$

   Let $p_{i,n} = p_{i,n}/S$, $\quad i < n$

   Let $p_{i,j} = p_{i,j} + p_{i,n} p_{n,j}$, $\quad i, j < n$

2. Let $TOT = 1$ and $\pi_0 = 1$.

3. For $j = 1, 2, \ldots, N$, do the following:

   Let $\pi_j = p_{0,j} + \sum_{k=1}^{j-1} \pi_k p_{k,j}$

   Let $TOT = TOT + \pi_j$

4. Let $\pi_j = \pi_j / TOT$, $\quad j = 0, 1, 2, \ldots N$.

The initial implementation of this step is using the lower bound $\overset{\sim}{\mu}_x^i(k)$ provided in Step 0, and according to (4.1) to define an upper bound queue length process by

$$\tilde{Q}_x^i(k) = \left[\tilde{Q}_x^i(k-1) - L_x^i \tilde{\mu}_x^i(k) + \lambda_x^i(k)\right]^+ \tag{4.32}$$

For the following consecutive recursions, the approximate queue length process in (4.32) is defined using the updated bandwidth allocation $\overset{\sim}{\mu}_x^i(k)$ in Step 4.

For this queuing model, compute the transition probability matrix $\tilde{P} = (\tilde{p}_{i,j})$ and compute the steady-state probability distribution $\tilde{\pi}_x^i(n)$ for the queue length.

Step 2: The implementation of computing the steady-state probability distribution b_n of the extra available bandwidth in a frame is simpler. Within the same type x of connection, all terminals have the same token generation rate r_x^i, which is a constant. By assuming that Q_x^i are i.i.d. random variables, b_n can be evaluated easily as follows. Let

$$B(k) = B - \sum_{x,i} \mathbf{1}_{\{Q_x^i(k-1) \geq L\}} \bullet \mathbf{1}_{\{\Delta_x = 1\}} \tag{4.33}$$

where Δ_x is a random variable representing the number of bandwidth slots guaranteed to a connection of type x, which is independent of all of the random variables and has the distribution:

$$\begin{aligned} \delta_1 &= P\{\Delta_x = 1\} = \frac{r_i^x}{L} \\ \delta_0 &= P\{\Delta_x = 0\} = 1 - \frac{r_i^x}{L} \end{aligned} \tag{4.34}$$

and $\mathbf{1}_A$ is the indicator function such that

$$\mathbf{1}_A = \begin{cases} 1, & \text{if A occurs} \\ 0, & \text{if A does not occur} \end{cases} \tag{4.35}$$

$B(k)$ represents the bandwidth left in a frame, measured in packets, after the guaranteed bandwidth assignment to all connections.

For the implementation, in terms of the approximate queue length process defined in Step 1, define an approximate resource process as (4.3).

$$\tilde{B}(k) = B - \sum_{x,i} \mathbf{1}_{\{\tilde{Q}_x^i(k-1) \geq L\}} \bullet \mathbf{1}_{\{\Delta_x = 1\}} \tag{4.36}$$

Using the steady-state distribution $\tilde{\pi}_x^i(n)$ of the approximate queue length compute the steady-state probability distribution

$$\tilde{b}_n = \lim_{k \to \infty} P(\tilde{B}(k) = n), \quad n = 0,1,\ldots, \tag{4.37}$$

of the approximate extra bandwidth.

Step 3: Since without the knowledge about the fairness function, we cannot determine the token balance process $E_x^i(k)$ and vise versa, we need to provide an initial variable of θ_x^i or E_x^i in the implementation of the recursive method. Intuitively, the fairness in the TBFQ scheduling reduces the difference in queue length between wireless terminals in the same type of connection. This property can be similarly achieved by allocating extra resources to the longest queue. In this way, the initial fairness function can be defined as follows.

Initial fairness function for random allocation: For a fixed wireless terminal i in a fixed type of connection x, the fairness function is defined as

$$\theta_x^i(k) = \begin{cases} B(k) \text{ with probability } 1/m_k, & \text{if } Q_x^i(k-1) = \max_{j,y} Q_y^j(k-1), \\ 0, & \text{if } Q_x^i(k-1) < \max_{j,y} Q_y^j(k-1), \end{cases} \tag{4.38}$$

where m_k is the number of terminals with the longest queue. We use this initial function in the first recursion.

Updated fairness function for random allocation: Starting from the second recursion, the fairness function is determined by using the token balance process approximated in Step 4:

$$\theta_x^i(k) = \begin{cases} B(k) \text{ with probability } 1/m_k, & \text{if } E_x^i(k-1) = \max_{j,y} E_y^j(k-1), \\ 0, & \text{if } E_x^i(k-1) < \max_{j,y} E_y^j(k-1), \end{cases} \tag{4.39}$$

To implement this step, we define the approximate token borrowing process according to (4.8) using the results in previous steps and Simplification 1 or Simplification 2.

Initial fairness function for random allocation:

$$\tilde{\theta}_x^i(k) = \begin{cases} B(k) \text{ with probability } 1/m_k, & \text{if } \tilde{Q}_x^i(k-1) = \max_{j,y} \tilde{Q}_y^j(k-1), \\ 0, & \text{if } \tilde{Q}_x^i(k-1) < \max_{j,y} \tilde{Q}_y^j(k-1), \end{cases} \tag{4.40}$$

Updated fairness function for random allocation:

$$\tilde{\theta}_x^i(k) = \begin{cases} B(k) \text{ with probability } 1/m_k, & \text{if } \tilde{E}_x^i(k-1) = \max_{j,y} \tilde{E}_y^j(k-1), \\ 0, & \text{if } \tilde{E}_x^i(k-1) < \max_{j,y} \tilde{E}_y^j(k-1), \end{cases} \tag{4.41}$$

Compute the steady-state probability distribution $\tilde{f}_n$ of the approximate token borrowing according to

$$\widetilde{f}_n = \lim_{k\to\infty} P(\widetilde{\theta_x^i}(k) = n), \quad n = 0,1,... \tag{4.42}$$

Step 4: The implementation of this step can be done easily according to

$$\tilde{\mu}_x^i(k) = \mathbf{1}_{\{\Delta_x=1\}} + \tilde{\theta}_x^i(k). \tag{4.43}$$

Use this updated bandwidth allocation in Step 1 and repeat the recursion until the computed steady-state probability distribution satisfies the required accuracy.

Step 5: In this step, we can treat the token balance process as a *GI/GI/*1 queue with arrival rate r_x^i and service process $\mu_x^i(k)$, respectively, and use the computational algorithms introduced above to compute the stationary distribution $e_x^i(n)$.

The implementation is realized by defining an approximate token balance process according to (4.9):

$$\tilde{E}_x^i(k) = \left[\tilde{E}_x^i(k-1) + r_x^i - L_x^i \min\left(\tilde{\mu}_x^i(k), \left\lfloor \frac{\tilde{Q}_x^i(k-1)}{L_x^i} \right\rfloor \right) \right]^+ \qquad (4.44)$$

measured in bytes.

Then compute the transition probability matrix and the steady-state probability distribution $\tilde{e}_x^i(n)$ of the approximate token balance for each connection.

Repeat Step 1 to Step 5 until convergence.

4.7 Performance analysis

Implementation of the numerical analysis can be found in the Appendix A, where the analysis was conducted using on-off voice sources.

In practice, a talk spurt interval typically ranges from 0.4 to 1.2 second in length and a silence interval from 0.6 to 1.8 second in length. Since we assume that

$$P(\text{number of on-frames} = n) = (1 - p_1)\, p_1^{n-1}\,,\ n = 1, 2, \cdots,$$

and

$$P(\text{number of off-frames} = n) = (1 - p_0)\, p_0^{n-1}\,,\ n = 1, 2, \cdots,$$

the average number of on- and off-frames are $1/(1 - p_1)$ and $1/(1 - p_0)$, respectively. Let the frame duration be 4ms ([85]), the average length of on- and off-duration are $0.004/(1 - p_1)$ and $0.004/(1 - p_0)$ seconds, respectively. Therefore,

$$0.4 \le 0.004/(1 - p_1) \le 1.2 \quad \text{and} \quad 0.6 \le 0.004/(1 - p_0) \le 1.8,$$

or

$$0.99 \le p_1 \le 0.9967 \quad \text{and} \quad 0.9933 \le p_0 \le .9978.$$

In normal uncompressed speech encoding, we can assume a constant packet generate rate during the on-period at:

$$\beta = 64 \text{ kbps} = 8 \text{ kbytes/s} = 8000 \times 1/250 \text{ bytes/frame} = 32 \text{ bytes/frame},$$

where the frame duration is 4 ms or 250 frames/s.

The average packet generate rate for the on-off process is

$$\frac{\frac{1}{1-p_1}}{\frac{1}{1-p_1}+\frac{1}{1-p_0}} \bullet 32 \text{ bytes/frame} = \frac{1-p_0}{2-p_0-p_1} \bullet 32 \text{ bytes/frame}$$

The token generate rate will be higher than the average packet generate rate to assure the system stability. Therefore, let $r = r_x^i$ satisfies

$$r > \frac{1-p_0}{2-p_0-p_1} \bullet 32 \text{ bytes/frame}$$

The system is designed such that

$$n_x r \le 80\% \text{ channel data rate},$$

where the channel data rate is 1.48 Mbps, or B = 14 packets/frame. Hence,

$$n_x r \le 0.8 \frac{1.48 \times 10^6}{8 \times 250} \text{ packets/frame} = 592 \text{ bytes/frame}.$$

This leads to

$$n_x r \le 0.8 \frac{1.48 \times 10^6}{8 \times 250} \text{ packets/frame} = 592 \text{ bytes/frame}.$$

Example 1 The channel data rate is B = 14 packets/frame. Assume that the average length of the on-period is 1 second and the average length of the off-period is 2/3 = 0.6667 seconds. The constant packet generate rate during the on-period is 64 kbps or β = 32 bytes/frame if the frame falls in an on-period. The basic bandwidth assignment is assumed to be the same as the packet generate rate during the on-period, which is 64 kbps for a connection, or r = 32 bytes/frame. If we require that the total token generate rate is no more than 80% of the data channel rate, then $n_x \le 18$. We assume that the size of the data buffer is unlimited, no debt limit d_x^i or creditable threshold t_x^i is set, and no limit is imposed on burst credit b_x^i.

To implement the above model, we use the following values for the system parameters: B = 14 packets/frame (but we do not need to input this value since it has be incorporated into expressions); β = 32 bytes/frame; r = 32 bytes/frame; p_1=.996; p_0=.994.

The convergence of the mean queue length is shown in the following table. Within 5 iterations, the error has reached to an order of 10E-5.

Table 4.1: Convergence of the computation of mean queue length, $\bar{Q}$ in bytes.

Iterat.	1	2	3	4
$\bar{Q}$	223.399	71.049	71.900	71.904
ε		152.349	0.850	0.004
Iterat.	5	6	7	8
$\bar{Q}$	71.904	71.904	71.904	71.904
ε	2.06E-05	9.76E-08	4.63E-10	2.30E-12
Iterat.	9	10	11	
$\bar{Q}$	71.904	71.904	71.904	
ε	9.94E-14	9.94E-14	9.94E-14	

For all values n = 1, 2, 3, 4, 5, 10, 15 and 20, the computed mean buffer occupancy (queue length) in bytes converges very fast in less than 10 recursions (a few recursions for some of the values) corrected to one byte. The mean buffer occupancy Q (measured in bytes) is provided in the following table.

Table 4.2: Average buffer occupancy, measured in bytes.

n	1	2	3	4	5	10	15	20
$\bar{Q}$	44.5	50.5	54.7	58.1	60.9	71.9	90.5	199.2

From the Little's formula: $\bar{\lambda}\bar{W} = \bar{Q}$, we can provide the average packet delay (measured in ms), where

$$\bar{\lambda} = 8000 \bullet \frac{1 - p_0}{2 - p_0 - p_1}$$

is the average packet generate rate for the on-off process, measured in bytes per second. In the case of Example 1, λ = 4800 bytes/sec.

Table 4.3: Mean packet delay, measured in ms.

n	1	2	3	4	5	10	15	20
$\bar{W}$	9.27	10.5	11.4	12.1	12.7	15.0	18.9	41.5

Three curves representing the cumulative probabilities of the buffer occupancy for n = 5, 10, 15, and 20 are plotted in the following figure.

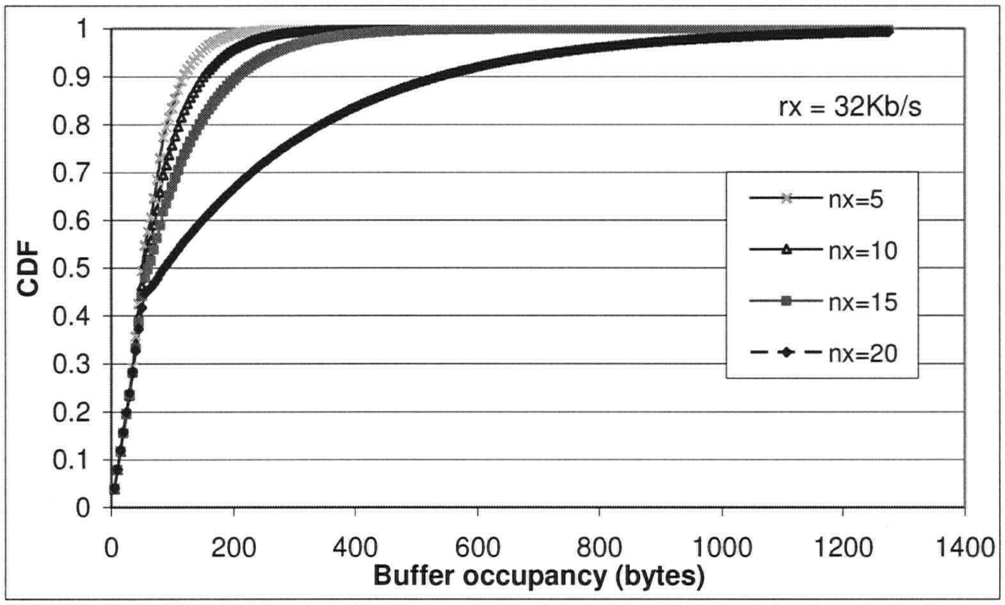

Figure 4.1: Cumulative probabilities of buffer occupancy for n = 5, 10, 15, and 20. r_x=32Kb/s.

In our next example, we take r to be between the data generation rate during the on-period and the average data generation rate during both on- and off-periods.

Example 2 The channel data rate, the average length of the on-period and the average length of the off-period remain the same as in the previous example: 14 packets, 1 second and 2/3 = 0.6667 seconds, respectively. The constant packet generate rate during the on-period remains 64 kbps or β = 32 bytes/frame if the frame falls in an on-period. The basic bandwidth assignment is now assumed to be 50 kbps or r = 25 bytes/frame. In this case, $n_x \leq 23$.

As a summary, B=14 packets/frame (but we do not need to input this value since it has be incorporated into expressions); β=32 bytes/frame; r=25 bytes/frame; p_1=.996; p_0=.994. Similar to Example 1, we provide the following computational results.

Table 4.4: Average buffer occupancy, measured in bytes.

n	1	2	3	4	5	10	15	20
$\bar{Q}$	44.3	52.2	58.6	64.4	69.8	94.5	130.2	280.0

Table 4.5: Mean packet delay, measured in ms.

n	1	2	3	4	5	10	15	20
$\bar{W}$	9.23	10.9	12.2	13.4	14.5	19.7	27.1	58.3

Three curves representing the cumulative probabilities of the buffer occupancy for n = 5, 10, 15, and 20 are plotted in the following figure.

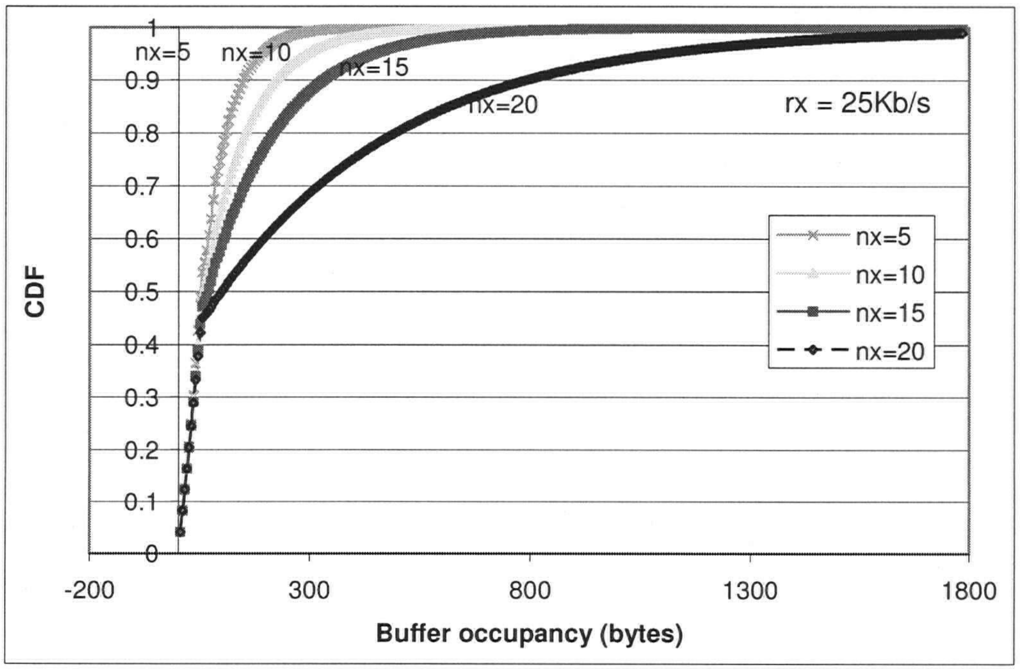

Figure 4.2: Cumulative probabilities of buffer occupancy for n = 5, 10, 15, and 20. r_x=25Kb/s.

It is important to compare the theoretical analysis to the simulation analysis as it validates the accuracy of the theoretical model. Using the same parameters as those in Example 1, event-driven simulations were conducted and the results are shown in Figure 4.3.

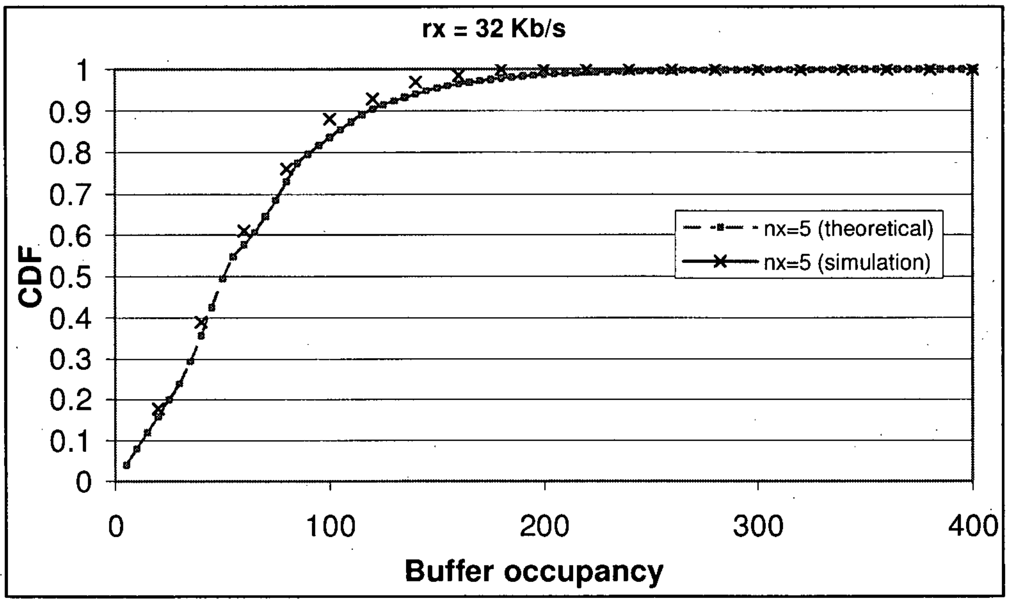

Figure 4.3 Cumulative distribution comparison between theoretical analysis and simulation (r_x=32Kb/s, n_x=5).

From the figure, we can see that the simulation results are very close to the theoretical calculations. In this case, the discrepancy is caused by the distribution of excess credits. As the number of voice connections increases, the simulation results continue to match the theoretical calculations, which lend confidence to the correctness of our simulation model. In the next two chapters, we shall apply the simulation model to more realistic wireless systems which would otherwise be very challenging to analyze mathematically.

Chapter 5 TBFQ IN TDMA TDD-BASED HETEROGENEOUS WIRELESS NETWORKS

In this chapter, we shall demonstrate how TBFQ can be utilized in a generic TDMA-based wireless system. Through simulation, the performance of TBFQ in both downlink and uplink is shown. Comparison with other algorithms is important as it demonstrates how TBFQ performs relative to others. Assumptions of the simulation such as the network and traffic models, architectures, and parameters are explained in detail.

The simulations were conducted using the commercial simulation software called OPNET Modeler [110]. All simulations were run for long periods (5000 sec in simulation time) to assure accurate and convergence of results; the 95% confidence interval is, at worst, within 5% of the values shown.

5.1 Desired properties of wireless scheduling algorithms

To compare various scheduling algorithms, we introduce some of the desired properties of a scheduling algorithm.

1) Efficient Bandwidth Utilization: One of the fundamental reasons for a scheduler is to exploit statistical multiplexing while providing QoS guarantees. This is very important in wireless networks. Unlike the broadband wired network, the wireless bandwidth is scarce. The scheduler should use statistical multiplexing to the fullest possible and provide maximum number of traffic streams with QoS guarantees.
2) Low or bounded Delay: Multimedia applications can be delay sensitive. Low latency and delay are desirable. Furthermore, the scheduler should have a limit on the delay of packet transmission. A delay bound can be achieved through the combination of connection admission control, and a traffic-shaping device such as a leaky bucket.
3) Fairness/Isolation: An algorithm should reimburse system resources that are not used by idle sessions to the backlogged sessions in a fair manner. In other words, the amount of available bandwidth distributed to backlogged sessions should be proportional to their contracted bandwidth. At the same time, an algorithm should have the ability to isolate well-behaved traffic from 'malicious' traffic, so that if a malicious traffic tries to draw more resources than it originally intended, the QoS of other well-behaved traffic should not be degraded.

4) Low Complexity: The complexity in calculation may affect the performance of an algorithm in practice. Algorithms should be kept simple yet efficient. The amount of sorting and searching should be limited. Ideally, O(1) complexity is desirable.

5) Heterogeneous Traffic Support: The algorithm should be able to multiplex diverse classes of traffic and should also be able to provide service differentiation within the same class.

6) Graceful service degradation: QoS performance can degrade for many reasons; bursty channel errors, location-dependent channel capacity and errors, and unanticipated traffic behavior (bursty traffic) to name a few. Although sometimes it is unavoidable to interrupt QoS abruptly, a scheduler should allow QoS performance to degrade gracefully as much as possible. As graceful degradation is subjective, we attempt to quantify the degree of graceful degradation as the following:

$$DGD_{\Phi} = \frac{\left|\Phi'[t_1,t_2] - \Phi_0[t_1,t_2]\right|}{\Phi_0[t_1,t_2]} \tag{5.1}$$

for all time intervals $[t_1,t_2]$. Φ is a QoS measure, for example, throughput or violation probability. The degrèe of graceful degradation in a QoS measure is the relative difference between the QoS measure under normal or ideal condition (Φ_0) and the QoS measure under non-ideal condition (Φ') during the period $[t_1,t_2]$ when abnormality occurred.

7) Connection Admission Control (CAC): In both [86] and [87], the authors pointed out that different service disciplines must be accompanied by different connection admission control algorithms. The CAC needs to have the knowledge of the underlying scheduling algorithms in order to make wise decision as to whether to admit more sessions. Admitting too many sessions may cause the QoS of existing sessions to degrade, while admitting too few may lose multiplexing gain and bandwidth utilization. Therefore, a service discipline should facilitate the use of a CAC.

8) Interoperability: Roaming of services across different wireless technologies will pose a challenge when heterogeneous wireless access systems are available. Normally, each wireless access system has its specific wireless scheduling algorithm implemented. However, it is desirable for a multimedia service algorithm to interoperate with other wireless systems for the purpose of seamless user-perceived QoS. Algorithms should be designed with the consideration that it would be possible to map QoS requirements to different systems. This would be necessary to provide seamless QoS for *native* terminals to roam to *foreign* systems, or vice versa, for *foreign* terminals to *native* system.

5.2 QoS-based framework and MAC model

The amalgamation of heterogeneous access networks is a challenge in next generation networks beyond the 3rd-generation (B3G). A mobile terminal that crosses from one network to another must be equipped with transceivers compatible with the systems it traverses; each system must also conform to its specific physical and MAC protcols. Carrying system-specific hardware for all systems is unavoidable. However, QoS provisioning can be supported uniformly across various systems without burdening the mobile terminals with multiple scheduling algorithms. In order to achieve this vision, we propose an architecture that is necessary for seamless QoS across heterogeneous networks.

Let us consider as an example, a mobile terminal traversing from a WLAN (802.11e based) network to an UMTS network and QoS is not to be interrupted. From the standpoint of seamless QoS, we could consider two approaches:

In the first approach, we assume scheduling algorithm is to be used for both access systems. The algorithm is administered by an agent in the core network fabric as one of the network service component it provides. The agent (also known as a QoS broker) may have the same algorithm implemented for both access systems. In the second approach, different scheduling algorithms are used for different systems. In this approach, the QoS broker could provide mapping functions between scheduling algorithms for various access systems. Note, however, that QoS mapping between two systems is not always possible.

In this research, we consider only the first approach where the same scheduling algorithm is used across different networks. Also it is necessary to request other QoS service components from the QoS broker that contains the necessary service components such as scheduling and CAC (see Figure 5.1). Once granted from the QoS broker, the requested QoS service components may be invoked in a distribute fashion, for example the scheduler may be invoked in all the base-stations. We assume dedicated high-speed connections are established between the services and the mobile access nodes. The de-coupling of scheduling algorithm from physical and data-link specifics not only allows the access of the same scheduling algorithm by different systems, it also permits ease of future enhancement of services. This framework coupled with the ability of the scheduling algorithm to work with various MAC is vital for seamless QoS provisioning.

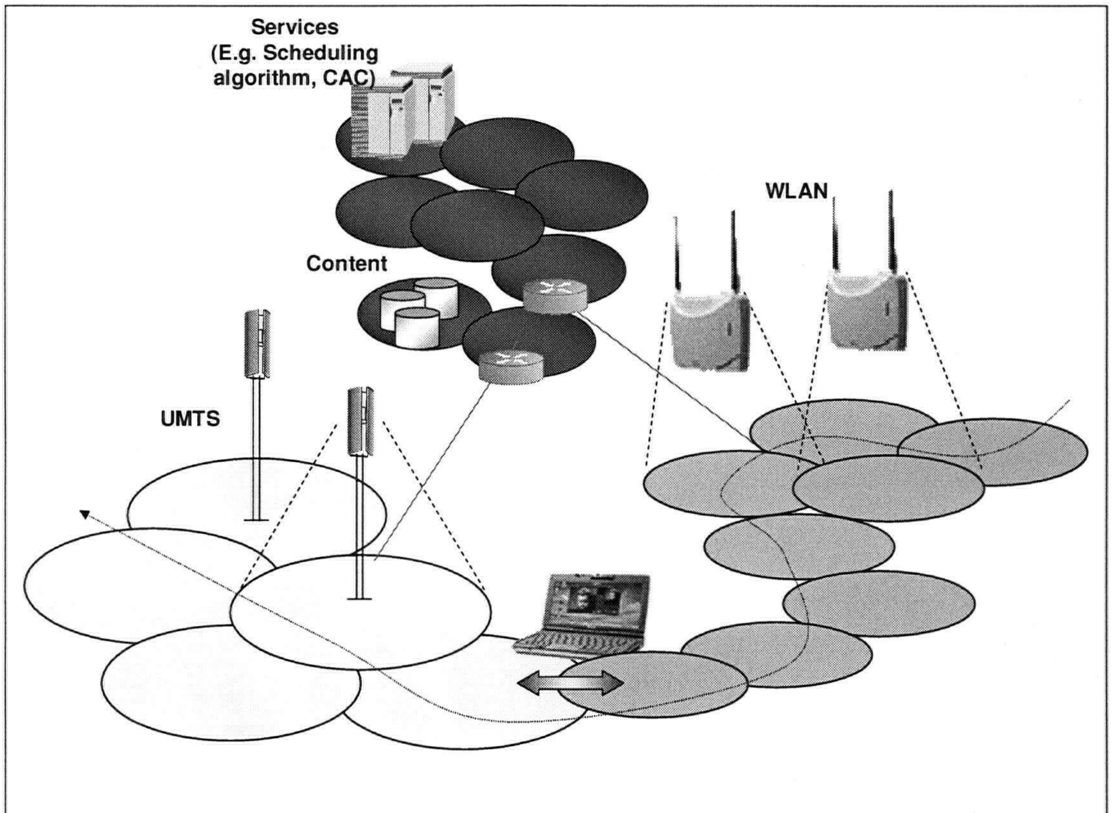

Figure 5.1 Network architecture of multi-access multi-service networks.

From Chapter 1, we gathered an understanding of the effect of channel access and scheduling algorithm on QoS, it is desirable to decouple the effect of channel access as much as possible as it gives clarity in understanding the effect due to the scheduler. Hence, we choose to use contention-free access and emphasize on the bandwidth reservation.

We focus on the packet transfer in both the uplink and downlink wireless channel. We assume that the quality of the wireless link is managed by physical layer, and the link is framed for time division multiple access/time division duplex (TDMA/TDD). The TDMA/TDD protocol has a number of attractive features, including the possibility of "on-demand" allocation of bandwidth. The TDMA/TDD frame structure is shown in Figure 5.2. The TDMA/TDD frame parameters are shown in Table 5.1. The fixed length TDMA frame is time-duplexed into an uplink and downlink channel, each further divided into control and data transmission periods. Slots assigned for control purposes are divided into control mini-slots each holding a control packet. The BS has absolute control of the numbers of data/control slots in each frame and the wireless terminal (WT) assigned to receive or send information during the data slots.

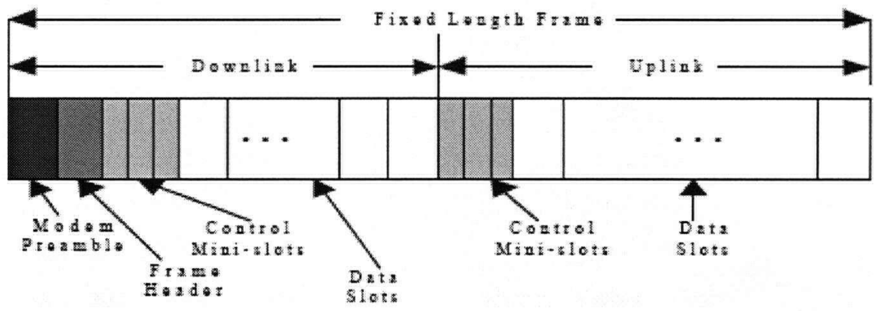

Figure 5.2 Format of the TDMA/TDD MAC frame.

Table 5.1 TDMA/TDD MAC FRAME STRUCTURE PARAMETERS

Channel Data Rate	1.48 Mbps
Frame Duration	4 ms
Packet size	53 bytes
Control slot size	20 bytes
Preamble size	16 bytes
Frame header	16 bytes
Number of slots per frame	14

The modem preamble is used for radio physical layer functions. The BS uses downlink control packets to announce slot allocations and assignments from and to the WT. Each downlink data slot holds one data packet with link layer and physical layer overheads for error correction and detection, etc. We assume that the packets are received with sufficient signal-to-noise-plus-interference ratio that, in combination with FEC coding, give a wireless data channel that meets the bit-error-rate QoS required by the respective services. In other words, the channel is error-free. Hence packet losses in our analysis are caused only by buffer overflow or by real-time (voice, video) packets dropped by the system due to violation of their delivery time constraints.

WT-originated packet transfer is achieved by a two-phase procedure. In the first phase, the access to the uplink of this TDMA system uses a slotted ALOHA based reservation protocol. The WT sends, in control mini-slots, a *packet_channel_request*, containing the number of time slots required for each uplink frame for its connection. The BS responds by a *packet_channel_assignment* message, with the assigned uplink slots for the WT. Initially (immediately after connection establishment), the effective bandwidth [84] is assigned to the WT. And periodically (e.g., one frame time) the BS will check for each *packet_channel_request* message, and schedule the necessary time slots to each WT if there are still enough time slots for the connection for the next uplink frame. In the second phase, each WT can communicate through in-band channel (using compressed control information) to the BS for dynamic allocation of more (or less) time slots as long as there is buffered data in the WT. Sometimes a connection requires no slot at all. In that case, the scheduler assigns no slot to it in the subsequent frames. An (1 bit) energy burst with a unique position for each admitted WT in each uplink control period is introduced for in-band signaling, such that when that WT has data in its buffer again, it uses an energy burst to signal the BS in the next uplink control slot period [85]. So, whenever an energy burst occurs in a specific position, the BS knows which previously idle connection has data to send. In the next downlink period the BS allocates the contracted bandwidth immediately. This allocation may be modified in subsequent frames depending on the connection's actual requirement using the above signaling mechanism.

5.3 Adaptation of TBFQ in TDMA

Referring to Figure 5.3 for the structure of TBFQ used for uplink and downlink scheduling, a WT within a service group x can be represented as WT^{i}_{x}, i=0,1,..., n_x ; x:{a=voice, v=video} where n_x is the total number of WTs in a service group x. Each WT^{i}_{x} has a LB associated with it and characterized as LB^{i}_{x}, = (r^{i}_{x}, P^{i}_{x}, D^{i}_{x}); i=0,1,...n_x ; x:{a=voice, v=video} where r is the token generation rate (bytes/s), P is the token buffer size (bytes), and D is the data buffer size (bytes). For uplink traffic, the policing function is implemented in the WTs, and for downlink traffic the policing function is implemented in the BS.

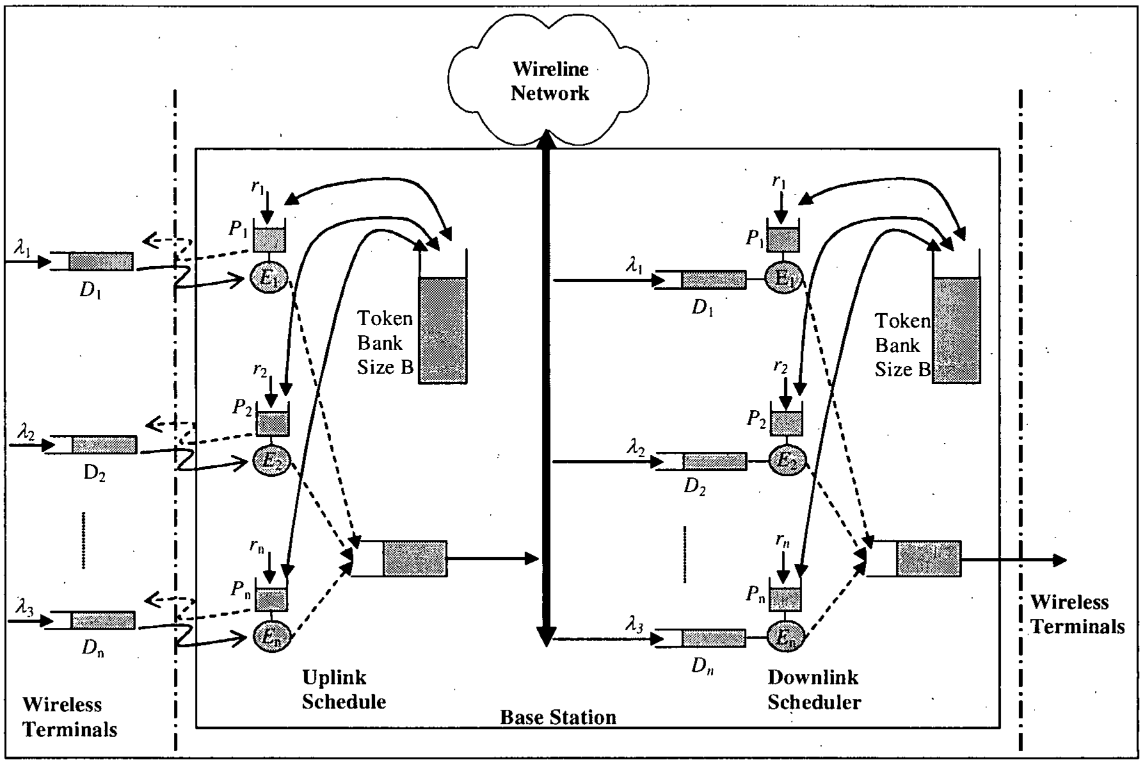

Figure 5.3 Structure of TBFQ for scheduling both uplink and downlink traffic at the Base Station.

The slot allocation is best described in the form of pseudo-code and is shown in Figure 5.4. The granting of slots is separated in two phases. First, the connections (or WTs) that have filled token pools will be granted a slot. If there are more connections than slots, the remaining connections will have to wait till the next frames. In phase two, connections may borrow tokens from the bank according to the algorithm described in Section 3.1.

```
First stage: grant slot according to Token Pool
// This stage scans all the terminals and grants one slot to
// the terminal whose request and token pool is not zero.
// $Terminal_i$ : the $i^{th}$ terminal, there are $N$ terminals in total.
// $Request_i$ : Request from $Terminal_i$
// $TokenPool_i$ : the token pool size of $Terminal_i$
// $Grant_i$ : slots that granted to $Terminal_i$
// $SatisfiedTeminals$ : total number of satisfied terminals.
FOR ($i = 1..N$)
{
IF (($Request_i > 0$) && ($TokenPool_i > 0$)) THEN
        Assign one slot to $Terminal_i$
        $TokenPool_i$—
        UpdateCouters($i$, $1$)
}

Second stage: grant according to borrowing tokens from Bank
// This stage lets terminals to borrow from token bank until
// either all $Slots\_per\_frame$ are consumed or all terminals are satisfied.
// $TotalGrants$: total number of slots granted in a frame.
// $Slots\_per\_frame$: number of slots per frame
// $Credit_i$: number of slots borrowed by $Terminal_i$
WHILE (($TotalGrants < Slots\_per\_frame$) && ($SatisfiedTeminals < N$))
{
        $i$ = HighestBorrowPriority ()
        $Borrow_i$ = BorrowBudget ($i$)
        IF ($Request_i < Borrow_i$) THEN
            $Borrow_i = Request_i$
        Assign $Borrow_i$ number of slots to $Terminal_i$
        $Credit_i$ += $Borrow_i$
        UpdateCounters($i$,$Borrow_i$)
}

Function UpdateCounters ($i$, $slots\_granted$)
//Update the counters of $Terminal_i$ after slots granted
{
        $TotalGrants$ += $slots\_granted$
        $Grant_i$ += $slots\_granted$
        $Request_i$ -= $slots\_granted$
        IF ($Request_i$==0) THEN
                $SatisfiedTeminals$++
}

Function HighestBorowPrioirty()
// Find the terminal that has the highest priority to borrow from the token bank.
{
        $Max\_borrow\_allowed\_this\_frame = Slots\_per\_frame - TotalGrants;$
        Find the unsatisfied $Terminal_j$ whose contribution to the bank is the maximum and it is elligible to
        borrow;
        RETURN $j$
}

Function BorrowBudget ($i$)
// Find the number of slots $Terminal_i$ is allowed to borrow.
{
        $Borrow\_allowed_i = amount\_contributed\_to\_token\_bank_i - Debt\_limit_i;$
        RETURN $min(Borrow\_allowed_i, Max\_borrow\_allowed\_this\_frame, Burst\_Credit)$
}
```

Figure 5.4 Pseudo-code for TDMA slot allocation algorithm using TBFQ.

5.4 Call admission control algorithm

The evaluation of performance of any broadband system that carries multiplexed streams of different traffic classes is meaningless without the use of a CAC algorithm [88]. The objective here is clearly very simple: given a call arriving, requiring a virtual connection with specified QoS (such as bandwidth, loss, probability, delays), should it be admitted? However, the implementation of the CAC can be quite complex. In a real network control, messages would have to be sent along the end-to-end path of a connection that would have to provide this connection to ascertain whether QoS objectives could be met without adversely affecting other calls already in progress. For homogeneous on-off sources, Guerin et al. [89] offered the Equivalent Bandwidth CAC algorithm formula. In a multi-access multi-service network scenario, the CAC problem becomes very complex and is a topic of research in itself. To reduce the complexity of our CAC algorithm, we make the following assumption: the forecast of handoff and mobility model of wireless terminals is not considered. The requesting connections would come either from the same cell or handoff from another system, and once admitted they do not move beyond the coverage of the cell. We developed an algorithm similar to the one proposed in [90]. Each connection i must provide both the desired bandwidth ($B_{i,d}$) and the minimum bandwidth ($B_{i,m}$) to the CAC. The desired bandwidth is between the average and maximum rate of the connection which is depended on the type of traffic. The minimum bandwidth is the minimum required by the connection for maintaining acceptable quality. The CAC first checks the total resources, B, which is comprised of B_H (resources reserved for handoff connections) and B_N (resources for new connections). For new connections, the CAC attempts to use B_N to allocate $B_{i,d}$ to the connection if possible, and if that is not possible it will try to allocate $B_{i,m}$. If the requested bandwidth is larger than $B_{i,m}$ the bandwidth compensation algorithm is invoked. Our bandwidth compensation algorithm is designed to work closely with TBFQ. The algorithm attempts to redistribute connections with bandwidth greater than $B_{i,m}$. If the compensation algorithm fails to find additional bandwidth, the connection is rejected. For handoff terminals requesting access, B_H is used instead. However if handoff and mobility models were considered, sharing of B_H and B_N may be provided. The algorithm is shown in Figure 5.5.

```
Connection admission control algorithm
// There are $n_N$ existing connections, and connection $(n+1)_N$ is requesting admission.
// $n_N$ connections are sharing bandwidth $B_N$ within a cell.
// There are $n_H$ existing handoff connections, and connection $(n+1)_H$ is requesting admission.
// $n_H$ connections are sharing bandwidth $B_H$ within a cell.
// This algorithm calculates whether there is enough resources for connection $(n+1)$.
{
CASE SWITCH connection (n+1) :
    CASE NEW_Connection :
            IF Available($B_N$) >= $B_{n+1,d}$ THEN
                Grant($B_{n+1,d}$) to connection $(n+1)_N$
            ELSE IF Available($B_N$) >= $B_{n+1,m}$ THEN
                Grant($B_{n+1,m}$) to connection $(n+1)_N$
            ELSE
                Compensate($B_{n+1,m}$ ,NEW_Connection, $B_N$)
            END;
    CASE NEW_Handoff :
            IF Available($B_H$) >= $B_{n+1,d}$ THEN
                Grant($B_{n+1,d}$) to connection $(n+1)_N$
            ELSE IF Available($B_H$) >= $B_{n+1,m}$ THEN
                Grant($B_{n+1,m}$) to connection $(n+1)_N$
            ELSE
                Compensate($B_{n+1,m}$, NEW_Handoff, $B_H$)
            END;
    CASE FUTURE_TYPES :        // for future extensions to incorporate mobility
    OTHERWISE :
            Reject();
}
Bandwidth compensation calculation function
Compensate ($b_i$, FLOW_TYPE, $B_T$)
// $b_i$ is the bandwidth to compensate for
// FLOW_TYPE is the type of connections
// $B_T$ is the total bandwidth available for FLOW_TYPE
{   $B_{max\_avail} = B_T - \sum_{i \in FLOW\_TYPE} B_{i,m}$
    $B_{avail} = B_T - \sum_{i \in FLOW\_TYPE} B_i$
    IF $B_{max\_avail} < b_i$
        Reject()
    ELSE IF $B_{avail} >= b_i$
        Grant($b_i$)
    ELSE
    {
        $B_{needed} = b_i - B_{avail}$
        $\forall i \in k$, $B_i > B_{i,m}$, subtract $\left(\frac{B_i}{\sum_{i \in k} B_i}\right) B_{needed}$
        Grant($B_{avail} + B_{needed}$) to connection $(n+1)_{FLOW\_TYPE}$
    }
}
```

Figure 5.5 CAC algorithm used in conjunction with TBFQ algorithm.

5.5 Traffic model parameters

A voice source generates a signal that follows a pattern of talk-spurts and silent gaps. Voice sources may be represented by an on-off model with negative-exponentially distributed ON and OFF periods [93][94]. In practice, each talk spurt interval typically averaging 0.4-1.2 sec is followed by a silence interval averaging 0.6-1.8 sec. Within each talk spurt intervals, 64 Kbps PCM-coded digital voice is assumed although in practice, voice encoding is often used.

For video traffic, we chose to use real data streams from videoconference employing the ITU-T H.263 standard. The parameters of the H.263 source are shown in Table I.

Table 5.2 PARAMETERS OF H.263 VIDEO SOURCE

Encoding format	352 x 288
Maximum Frame Size	66416 bits
Minimum Frame Size	1615 bits
Frame Rate	15 Hz
Mean Bit Rate	90.42 Kbits/sec
Total Number of Frames	500
Encode Frames	I: 1 P: 499 B:0

For data traffic, World Wide Web traffic model is used. It has an ON-OFF two-state model with the ON period consisting of a sequence of document page transmission request from an individual user [8]. The length of the ON period has a Weibull distribution

$$p(x) = \frac{k}{\theta}(\frac{x}{\theta})^{k-1} e^{-(x/\theta)^k} \tag{5.2}$$

with k between 0.91 and 0.77, and θ between $e^{4.4}$ and $e^{4.6}$. The length of the OFF period has a Pareto distribution

$$p(x) = \frac{\alpha k^{\alpha}}{x^{\alpha+1}} \tag{5.3}$$

with α between 0.9 and 0.58, and k = 60 sec. Within the ON period, the page inter-arrival time also has the Weibull distribution with k = 0.5 and $\theta = e^{1.5}$. The size of the document page has the Pareto distribution with $\alpha < 1$.

5.6 Downlink performance

We compare the performance of TBFQ with FIFO (first-in-first-out), PGPS, EDF, and RR schemes. For PGPS and EDF, we allow the transmission of timing information to be transmitted over the wireless medium to the BS without creating overhead to the channel capacity. Even though this is unrealistic in practice, for simulation purposes these schemes can be used as performance benchmarks.

The token bucket parameters used for voice and video traffic are specified in Table 5.3. These parameters serve as an initial starting point for our simulation.

Table 5.3 Scheduler parameters for voice and video traffic

Token Bucket Parameters for Voice Traffic		
TBFQ	'Burst Credit'	530 bytes
	'Debt Limit'	-530 bytes
	'Creditable Threshold'	0
	Token Generation Rate	50kb/s
	Token Pool Size	53 bytes
P-GPS, EDF, RR, FIFO	Token Generation Rate	50kb/s
	Token Pool Size	530 bytes
Token Bucket Parameters for Video Traffic		
TBFQ	'Burst Credit'	530 bytes
	'Debt Limit'	-530 bytes
	'Creditable Threshold'	0
	Token Generation Rate	250kb/s
	Token Pool Size	53 bytes
P-GPS, EDF, RR, FIFO	Token Generation Rate	250kb/s
	Token Pool Size	530 bytes

For voice traffic, token generation rate (50Kb/s) is selected based on the criteria that the provision is between the average traffic generation rate and the peak rate. To ensure fair comparison, burst credit size of 10 packets for TBFQ and token pool size of the same for other algorithms were used to ensure the same burst potential are achievable for all the algorithms.

To ensure a fair comparison, same leaky bucket parameters are used for PGPS, EDF, RR, and FIFO. Leaky bucket was used as the flow policing mechanism. In implementing the leaky bucket with these schedulers, we define priorities for traffic classes similar to [85], where voice and video traffic have the highest priority, and data has the lowest priority. When the scheduler is servicing "conforming" requests, which is defined as requests that belong to connections whose token pool is nonempty, it follows the priority table. Within each priority class, the scheduler serves the request of each connection as long as slots are available and the connection's token pool is not empty. The order of the packets served in each class is determined by the algorithm of the schedulers. Every time a slot is allocated to a connection, a token is removed from that connection's token pool. Within the same priority class, the scheduler gradually allocates one slot at a time to the connection that has the most tokens left in its token pool. When all the token pools are emptied, the scheduler serves "nonconforming" requests. It starts allocating slots for connections, starting from the highest priority (conversational) down to the lowest priority (background best effort) until all the requests are served.

In Figure 5.6 and Figure 5.7, the voice packet mean delay and jitter are shown. In all three schemes (TBFQ, PGPS, and FIFO), no packet discards were exercised. At low loading scenario, all schedulers have very similar delay performance. This is because the token generation is at a rate (50Kb/s) that is sufficient to accommodate the average voice packet generation rate, which is approximately at 39Kb/s. As loading increases, the TBFQ performance improvement becomes apparent. It has the lowest mean packet delay value starting from 31 voice users, and both EDF and PGPS schemes have higher mean delay at high loading condition. For the FIFO and RR schemes, the high average delay shown is expected. The delay performance between TBFQ, PGPS, and EDF are quite similar in low and medium loading conditions, although EDF has the lowest delay. The lower delay in TBFQ than in EDF and PGPS at high loading condition is because the systems is starting to saturates, and link utilization performance is very sensitive at these loading conditions.

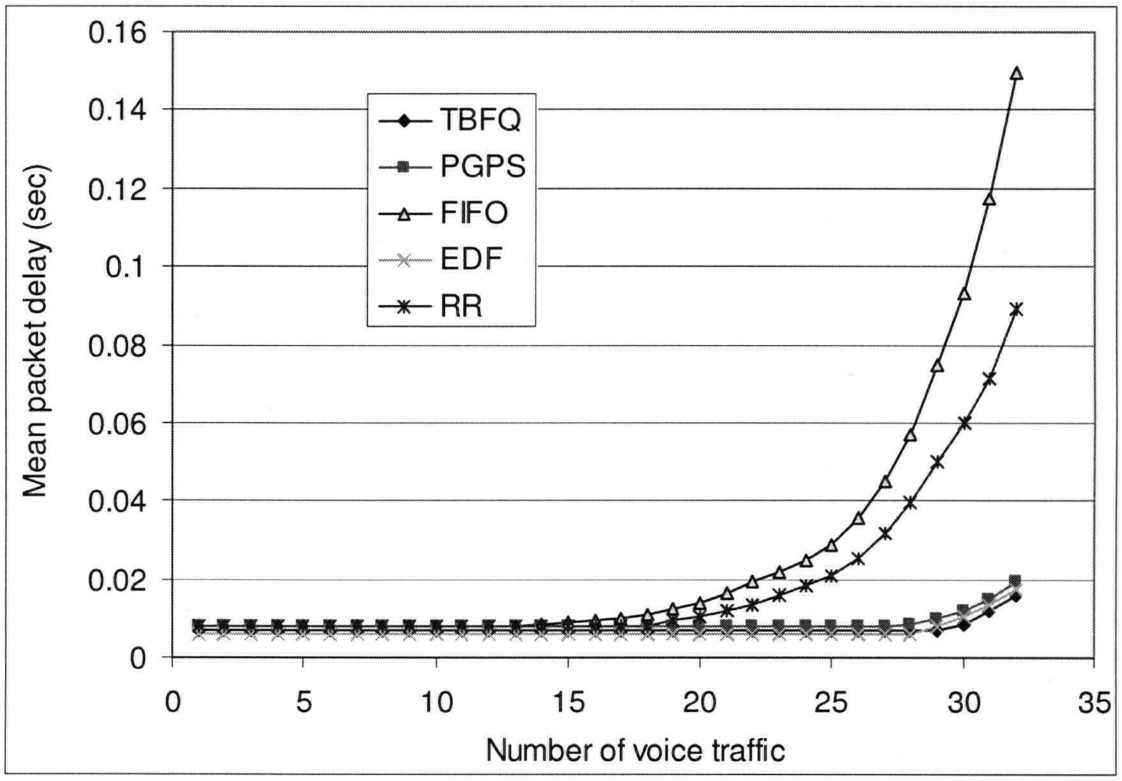

Figure 5.6 Mean packet delay of voice traffic source. No packet discard is used.

If we allow voice packets, that exceed the 40ms delay threshold, to be dropped before they are transmitted to the BS, then we can obtain the probability of violating packets (see Figure 5.8). Again, at low loading conditions, all the schedulers have very similar performance. FIFO and RR start to show increasing violation probability at around 15 voice calls. As the loading increases, the better performance in TBFQ over PGPS and EDF becomes apparent; this is due to the constraints of the leaky bucket flow policing mechanism used for PGPS and EDF.

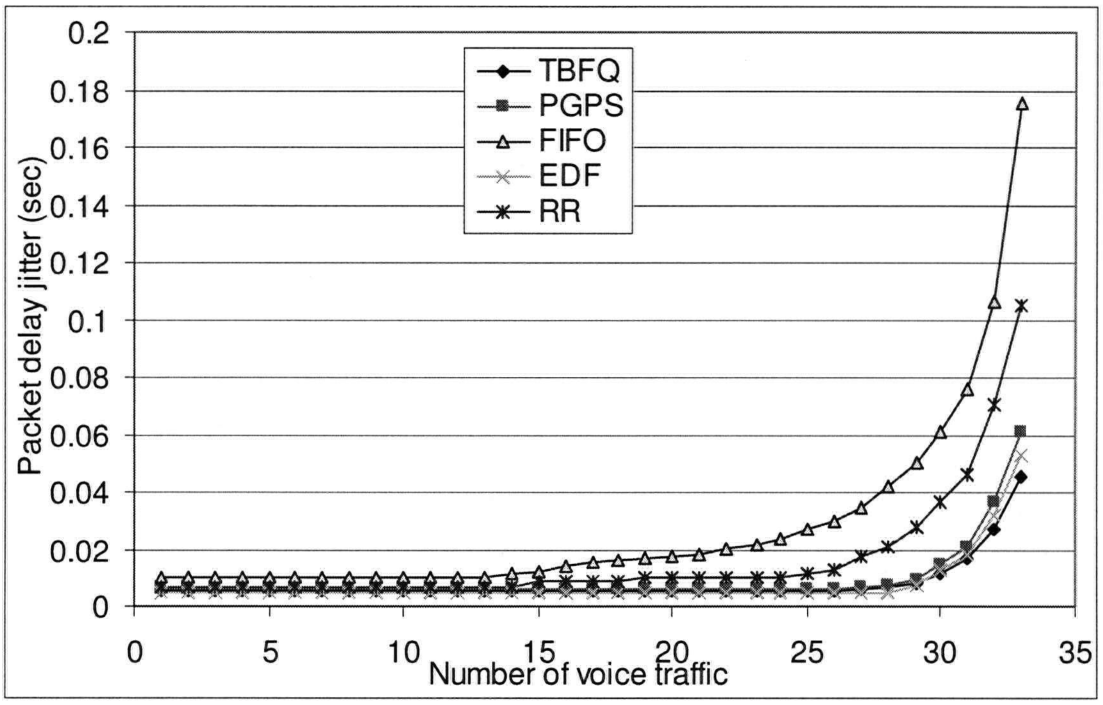

Figure 5.7 Delay jitter of voice traffic source. No packet discard is used.

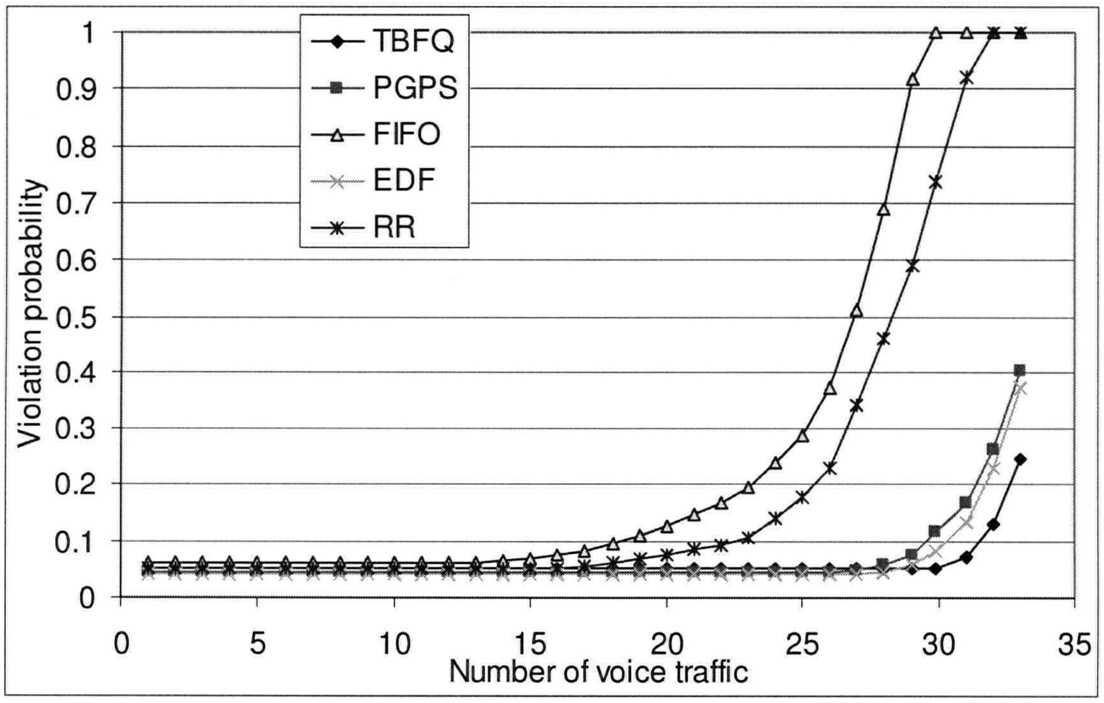

Figure 5.8 Violation Probability of voice packets that has exceeded the 20ms delay limit. Without CAC.

For the real video traffic used in the simulation, the length of the video stream is limited to 500 frames of data. In other words, only 33 sec of video (at 15 frames/sec) data is available. In order to simulate a longer connection time, the video sequence is recycled from the end to the beginning. Since only one video source is used, for multiple video sources the start time and the starting frame number are randomized during the simulations. If we define the burstiness of a video source to be the ratio of the peak frame size to the average frame size, the source that was used had a burstiness of 12.8. The peak frame size is caused by the insertion of the I-frame in the H.263 sequence. To an end user, the delay of video frames is more meaningful than delay of packets. So for video streams, we measure the video frame delay and jitters rather than the packet delay and jitters even though packet delays contribute to the video frame delay. In Figure 5.9 and Figure 5.10, the video frame delay and jitters performance are shown. Both TBFQ and PGPS meet the QoS requirement of video traffic; for TBFQ, a frame rate of 100 frames/sec can be achieved when there are 10 video traffic sources. For PGPS, approximately 60 frames/sec can be achieved where at 10 concurrent video traffic sources. However, we can clearly see that FIFO is not suitable for handling video traffic which has very low frame rate of around 16 frames/sec even at fewer concurrent sources.

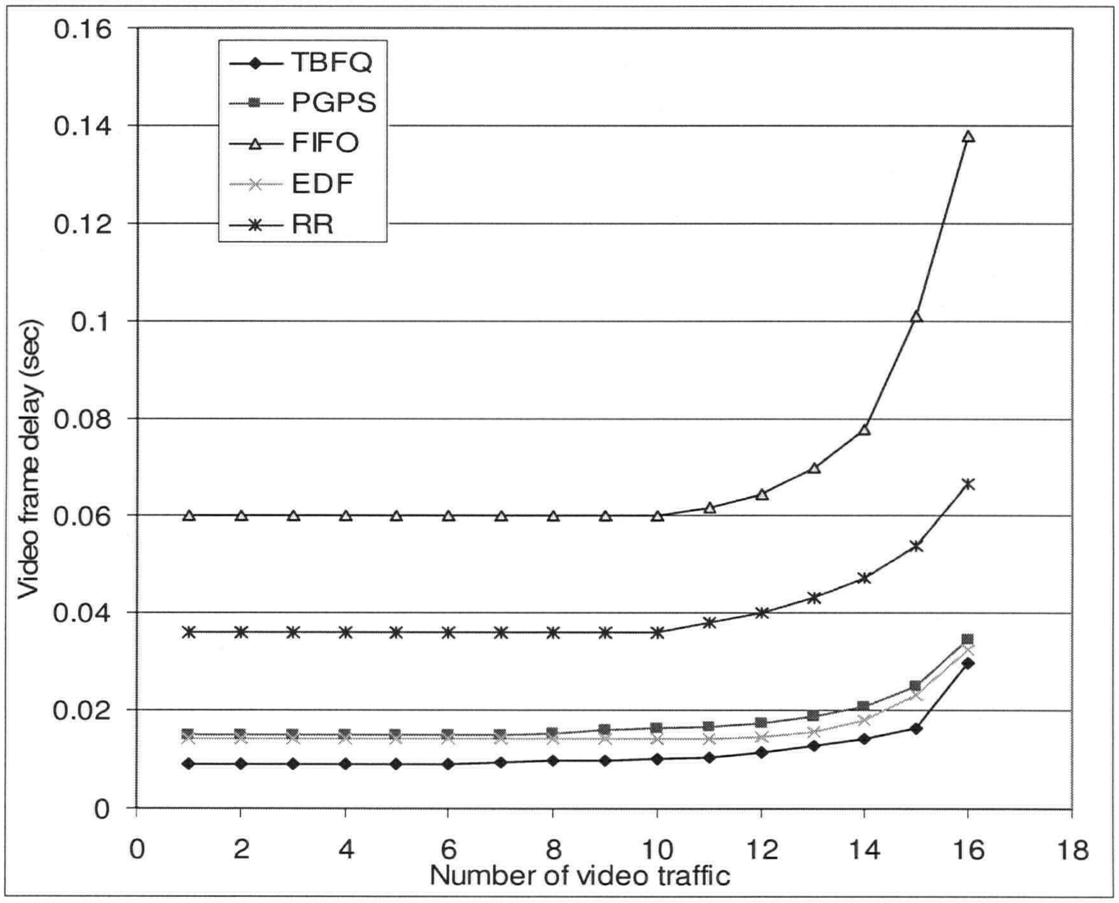

Figure 5.9 Video traffic frame mean delay

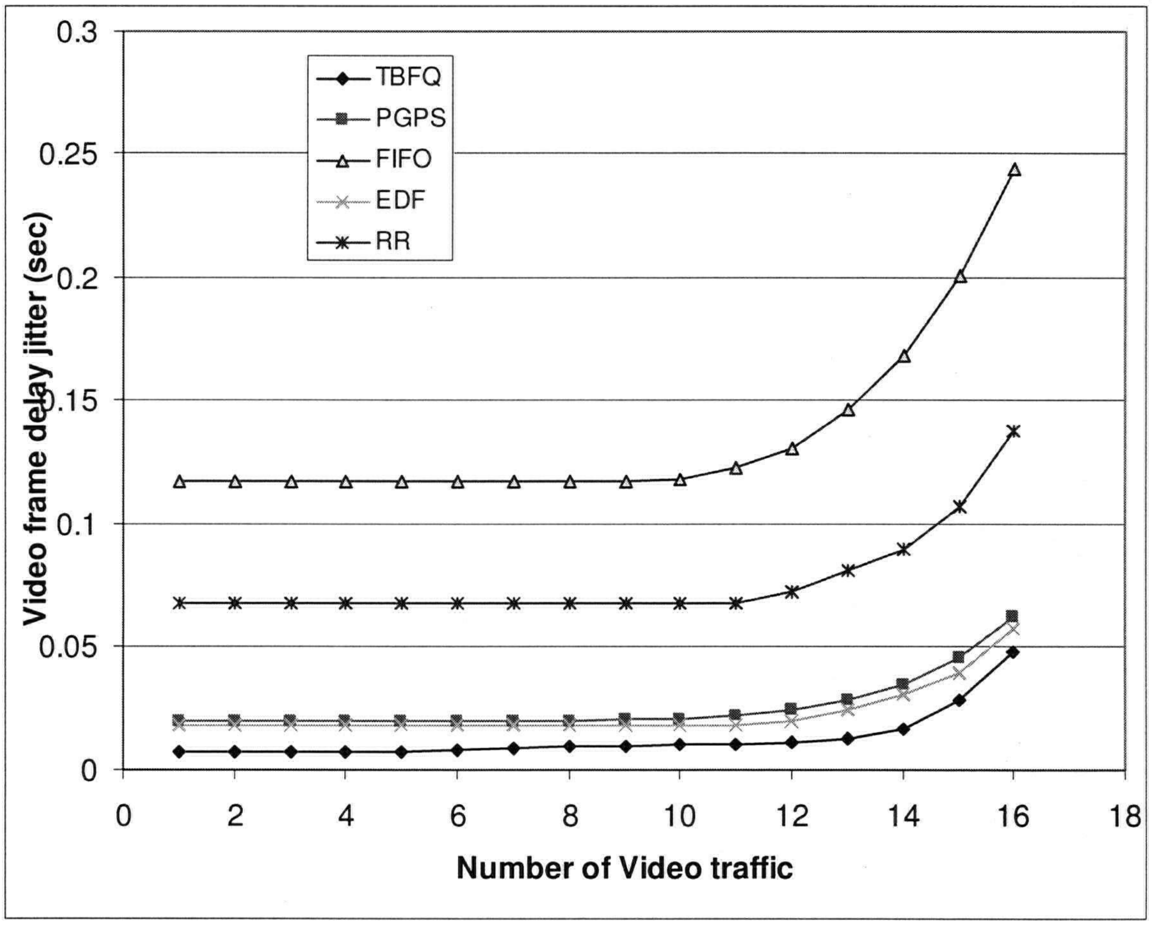

Figure 5.10 Video frame delay jitter.

For the WWW traffic, the burstiness is in the order of hundreds; and for this type of traffic, throughput is a more appropriate QoS measure than packet delay and jitters. The throughput is defined as the rate (in Kbits/sec) of data packets arriving at a WT. Our result for the throughput performance is shown in Figure 5.11. For TBFQ, the throughput is around 58.7Kb/s; and 58.3Kb/s for PGPS. However, FIFO has wider fluctuation in throughput performance. This is cause by the burstiness of WWW traffic. FIFO has very little capability to alleviate large bursts of data. The lack of capability becomes worst when it is constrained by leaky bucket. PGPS and TBFQ, on the other hand, are much more resilient to bursty traffic. TBFQ has slightly higher throughput, because of the ability to burst out traffic from the potential it accumulates in its priority index. Also, the leaky bucket flow policing limits the performance of PGPS.

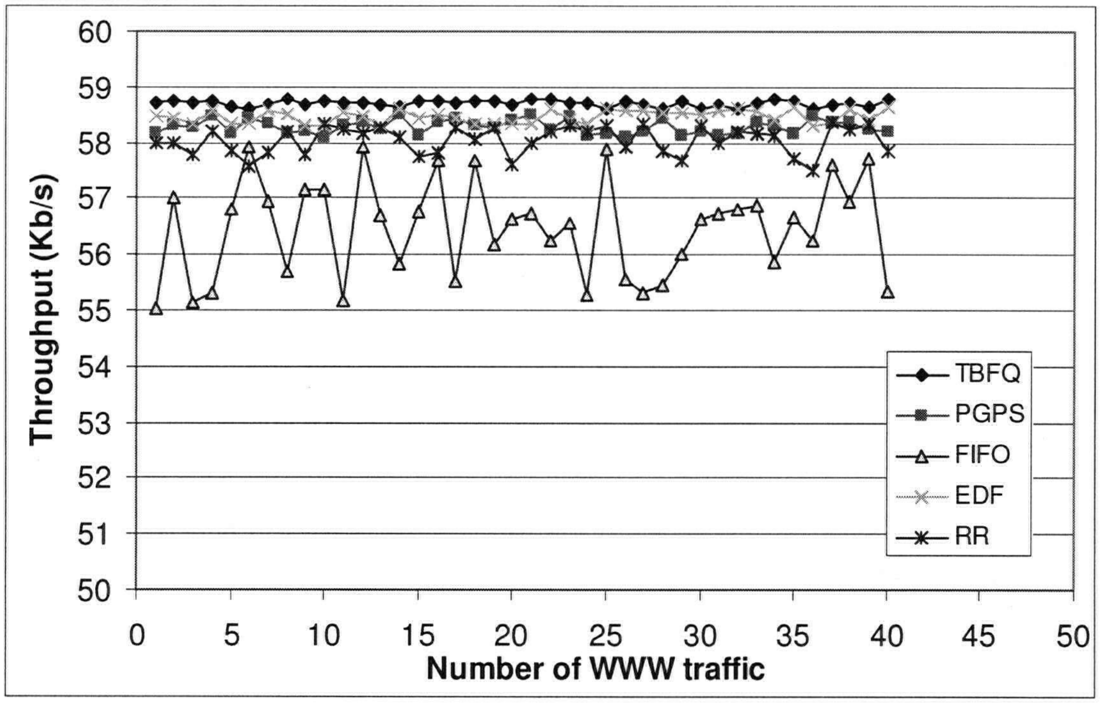

Figure 5.11 Throughput measurement of WWW traffic source.

The choice of token rate is a critical one. Figure 5.12 shows how the token rate affects the performance. We take 10 video traffic and study the violation probability (packet delay >100 msec) as the token generation rate varies. The range of token rates was set so that it covers the average rate of the video traffic, as well as several times that average. We found that EDF and PGPS would eventually perform better than TBFQ when the token rate was increased to a high enough level. However, we also note that TBFQ's performance improves quicker than the others when the token rate is increased. The advantage of this becomes clear when the operator has to determine (at the BS) the token rate (between the average and the maximum rate) to be used for a bursty traffic. Determining the token rate of a bursty variable-bit-rate traffic is not a trivial task. TBFQ has a wider tolerance of token rates for acceptable performance, and therefore for non-bursty sources such as the voice connections it makes little difference whether we allocate the peak rate 64Kb/s or its average of 0.6×64Kb/s.

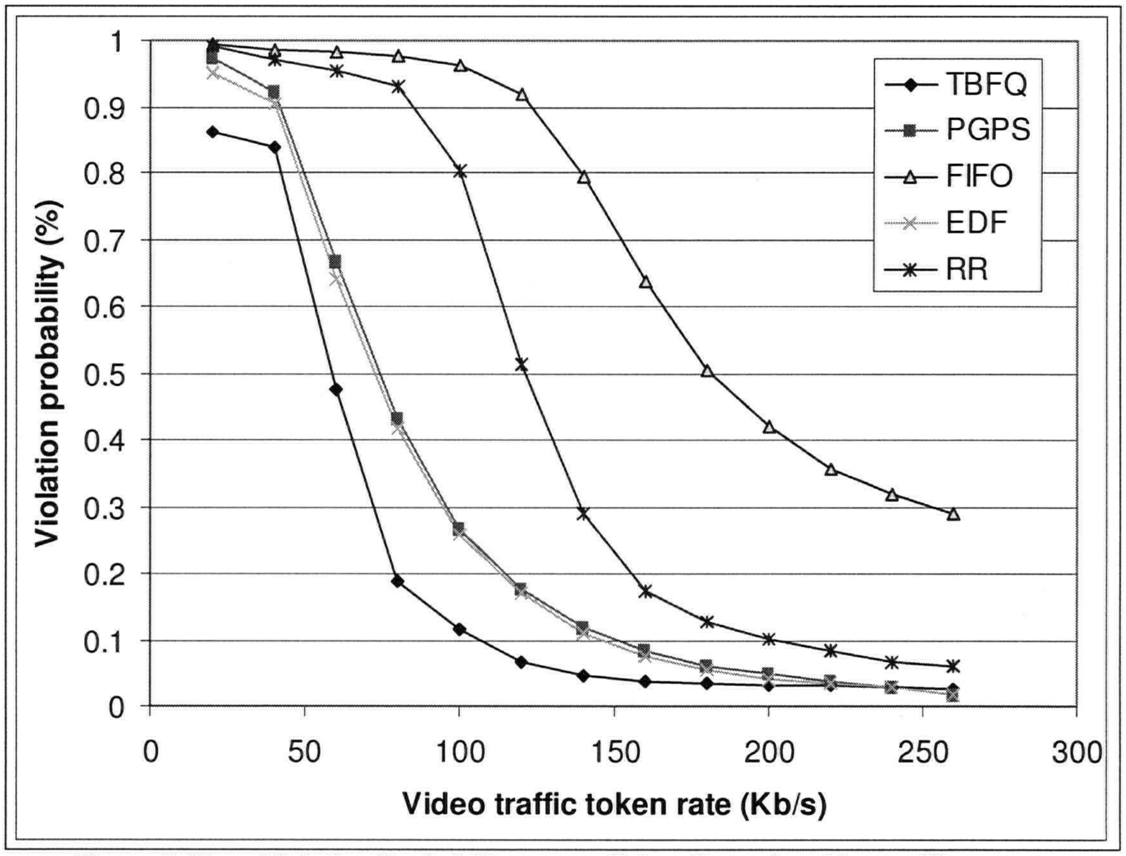

Figure 5.12 Violation Probability versus Token Rates for video traffic source.

We can also look at this from another perspective. In bursty traffic streams there will be periods in time when the traffic will exceed the assigned token rate. This happens very often in bursty real-time applications because the peak to average rate can be significant (12.8 in this case). By using TBFQ, the BS is able to gracefully accept the temporary traffic profile violation and maintain acceptable QoS. This is due to the 'soft' QoS provisioning capability of TBFQ [78]. In the notorious error-prone wireless environment, *soft* capability is necessary.

5.7 Uplink performance under error conditions

We adopted a two-state Markov model (from [92]) to emulate the process of packet transmission errors. The channel varies between a "good" state and a "bad" state, *s0* and *s1,* respectively, for each packet transmission. During *s0*, packets are transmitted error free, and errors occur during *s1*. The probability of remaining in a good state is 0.328. The probability of transiting from good to bad state is 0.0000235 and 0.46945 for bad to good state.

In addition to delay and throughput performance for voice and video traffic alone, we also mixed the voice and video traffic, and investigate the performance in link utilization, isolation, fairness, and the selection of required bandwidth.

In Figure 5.13, the cumulative distribution of delay for voice traffic is shown. For voice, we allow system loading of 0.86. For RR, 50% of delay is below 40ms. This shows the inability of RR in coping with error conditions. In FIFO, we allow the BS to have the knowledge of the timing information of packets in each WT even though in practice that is not possible. The head-of-queue (HOQ) packet of each queue is compared to determine which comes first and is scheduled according to FIFO. This order is fixed, and if a connection goes into *bad* state it will have to wait until it comes out of bad state and the other HOQ packets are served first. The second packet in the same queue faces the same challenge. RR is slightly better in that there is a better chance that when an HOQ packet comes out of the bad state, it can be served earlier. If a queue has just become good, and RR is pointing at the previous queue (in the queue sequence), then it will be served almost immediately. TBFQ has very similar delay performance from EDF and PGPS. This is mainly because the traffic is constant-bit-rate and system load is moderate. The minor difference comes from treatment of error conditions. The effect is amplified when we look at video frame delays of real-time video traffic.

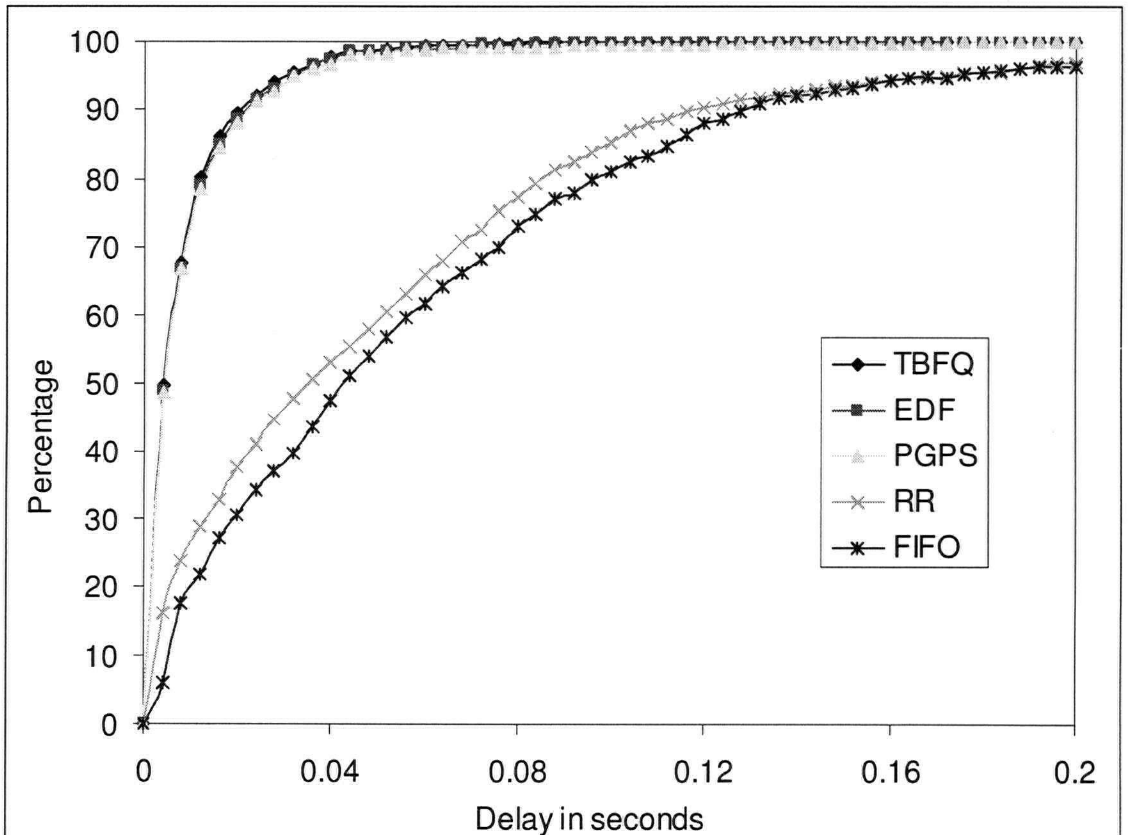

Figure 5.13 Packet delay CDF for voice traffic source only (load=0.86)

In video frame delay (Figure 5.14), TBFQ has a clear performance advantage. If we consider a large video frame that has arrived at the input queue (in WT), the flow policing mechanism in PGPS and EDF restricts it through the leaky bucket mechanism. In practice, it would not be feasible for the BS to know the timing information of packets from each WT. However, for the sake of benchmarking, we allow BS to have the knowledge of timing information in each WT. The BS keeps track of the timing and uses a virtual system to emulate all the queues from the WTs. If the destination is in a bad state, sorting is necessary to make room for packets whose destination is in good state. There is no mechanism in both EDF and PGPS to allow a burst to borrow bandwidth from the future; otherwise, it would not meet the instantaneous fairness that these schemes are designed for. However, in TBFQ, arriving burst first stay in their in-coming queue, they are accumulated in their queue until they are ready to be served for transmission. The out-going queue is a virtual queue located in the BS and is reserved only for packets ready to be transmitted immediately. If a connection has accumulated enough *E* before a large video frame arrives, it is very likely that it can borrow tokens in advance from the bank. If that is the case, the overall delay is lowered, which results in Figure 5.14. This is a tradeoff with instantaneous fairness. For example, during the period when the large video frame of connection A is being served, connection B may have arrived in its queue a large frame of data as well. Connection B would have to concede to connection A until its *E/r* exceeds that of connection A (assuming *debt_limit* is not reached). At lower system loading scenario (0.52 or less), performance difference is negligible among TBFQ, EDF and PGPS – over 90% of the delays are less than 23ms. At higher loading scenario (e.g. 0.94 loading), the system saturates and the performance of TBFQ, EDF and PGPS all lower accordingly – 48.7% of delays are below 150ms. The performance difference gap between TBFQ, EDF and PGPS starts to diminish. We believe that there is an optimum loading point where TBFQ can maximize its statistical multiplexing gain.

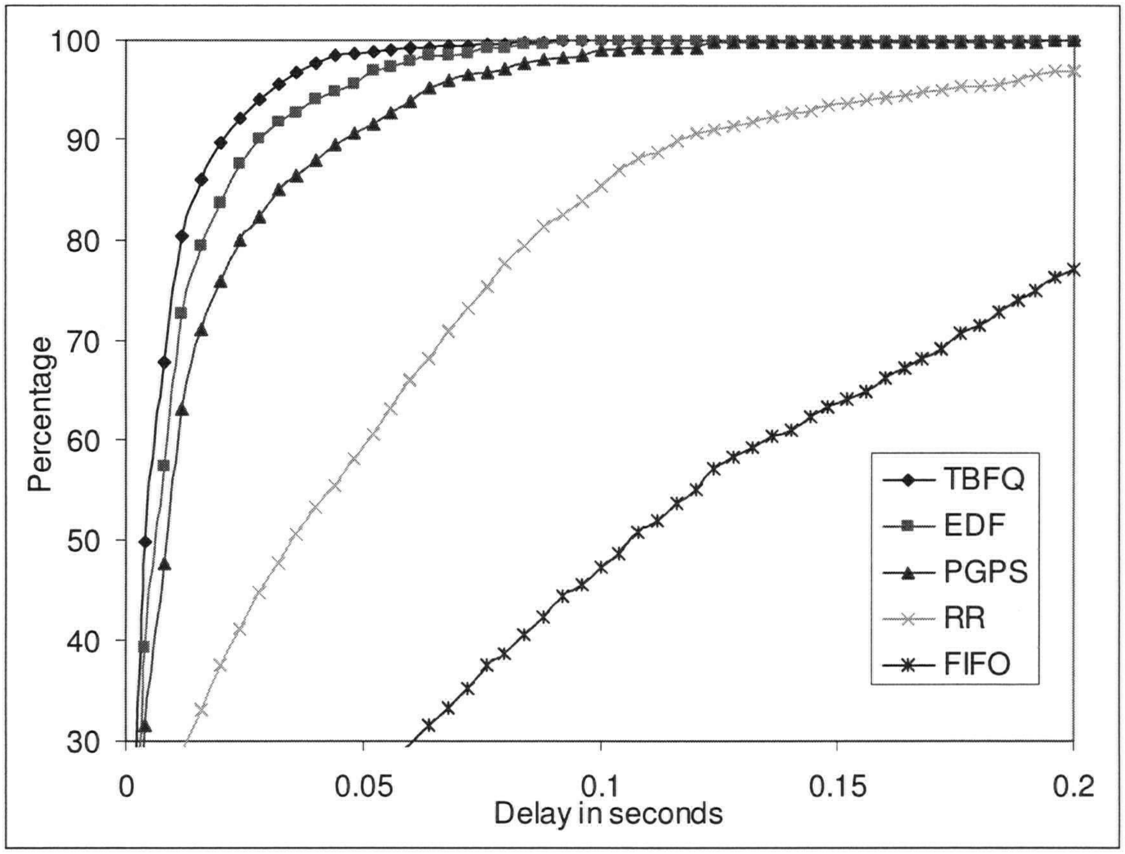

Figure 5.14 Frame delay CDF for video traffic source only (load=0.73)

Mean throughput performance was measured for data traffic (Figure 5.15). To provide the users with 64Kb/s data service, we set the token generation rate to be 64Kb/s for all connections. With 30 data connections admitted (81% loading), 51.1% of packets had 60Kbps or less for TBFQ , 45.3% had 57.5Kbps or less for EDF, 58% had 57.7Kbps or less for PGPS, 54.5% had 55Kbps or less for RR, and 41.2% had 45Kbps or less for FIFO. Maximum of 65Kbps was achieved by TBFQ, EDF and PGPS. This is because of the less than maximum loading so there is extra bandwidth for additional services.

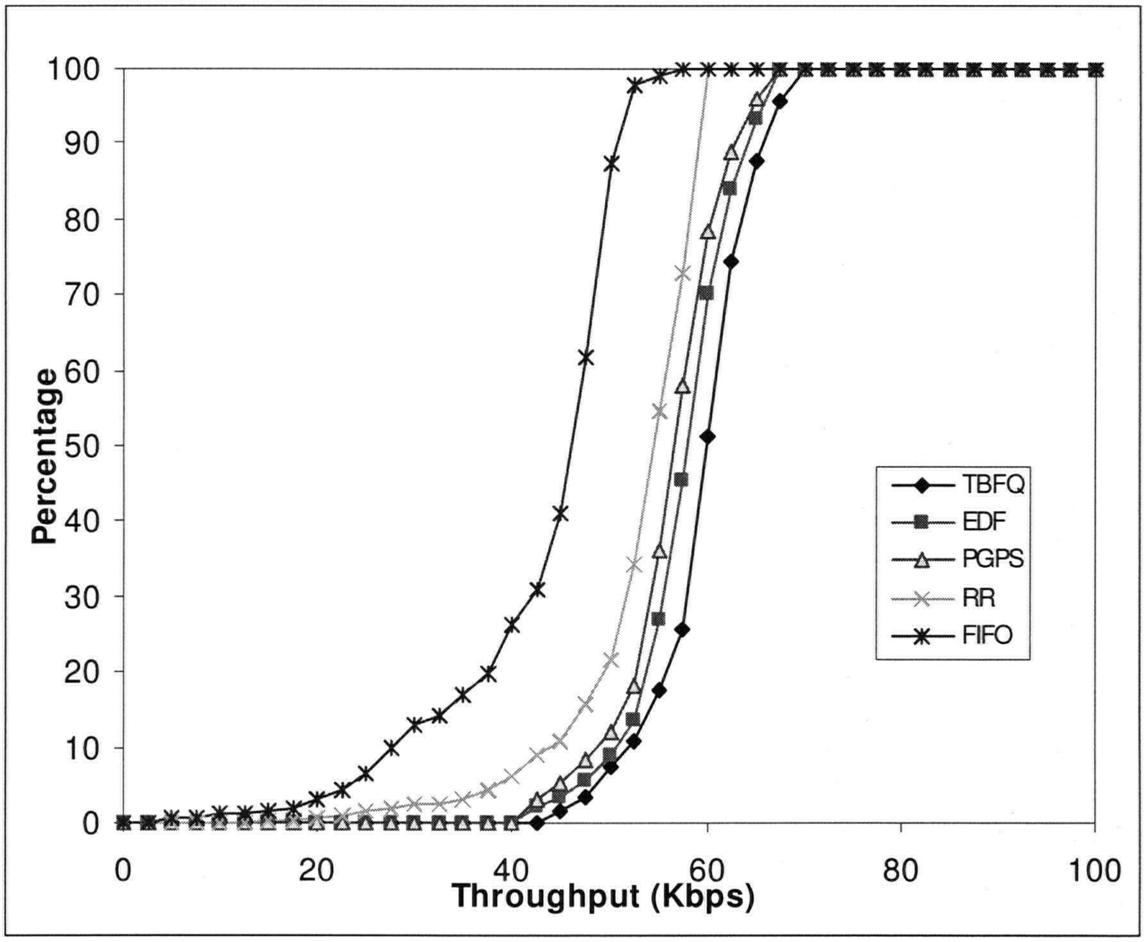

Figure 5.15 Throughput CDF for data traffic source only (0.81 loading)

In Figure 5.16, the effects of voice traffic load on voice packet mean delay is shown with the combination of various concurrent video connections (4, 7 and 10 video traffic). No packet discard was exercised. LB policing was used and the parameters are the same as those shown in Table 5.3. The reason for lower delay performance seen in TBFQ is similar to what we have seen in the downlink. The poorer than expected performance of PGPS and EDF is caused primarily by the constraints in the LB. Though the use of flow policing is arguably an implementation decision, we believe it is necessary as operator would like to be able to segregate users into various levels of QoS and price categories. Without some form of flow policing, paid users from the higher categories would eventually opt for the lowest price package.

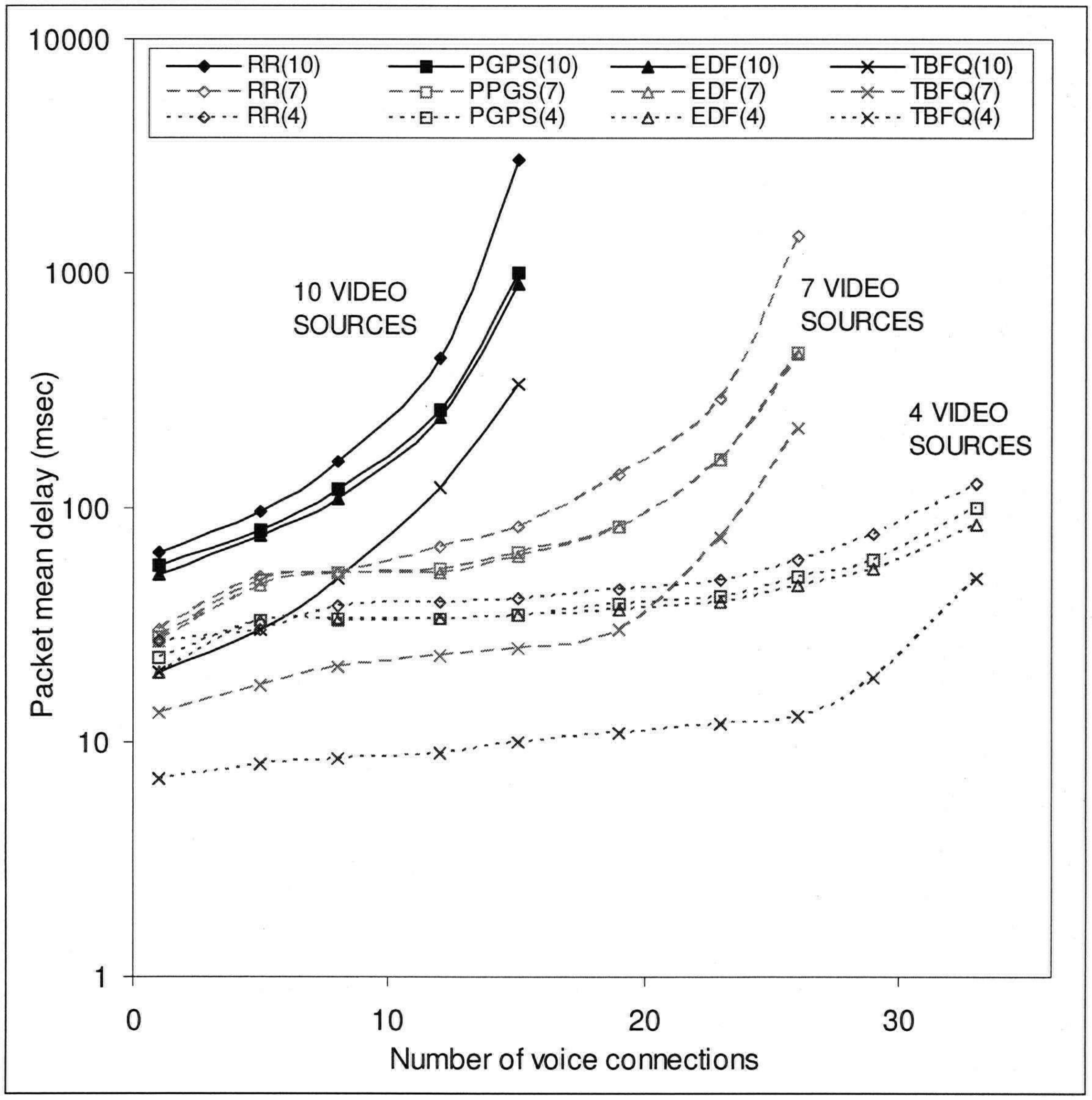

Figure 5.16 Voice Packet Mean Delay with video traffic source.

For the sake of argument, if flow policing were removed, PGPS and EDF would naturally have the best delay performance. However, their improvement over TBFQ is not significant (less than 5ms) in the normal loading condition. The complexity of both EDF and PGPS makes them impractical to deploy in any wireless network and not to mention the issues in providing seamless QoS. However, the work complexity of TBFQ is $O(1)$ as determined in ***Theorem* 3.1**.

Figure 5.17 shows the schedulable region where the packet violation threshold and delay tolerance for voice and video are (10%, 40ms) and (10%, 100ms), respectively. TBFQ supports the greatest number of video traffic for a given voice load, but EDF and PGPS provide performances that are nearly as good. The performances for EDF and PGPS would improve and exceed that of TBFQ if LB flow policing were removed.

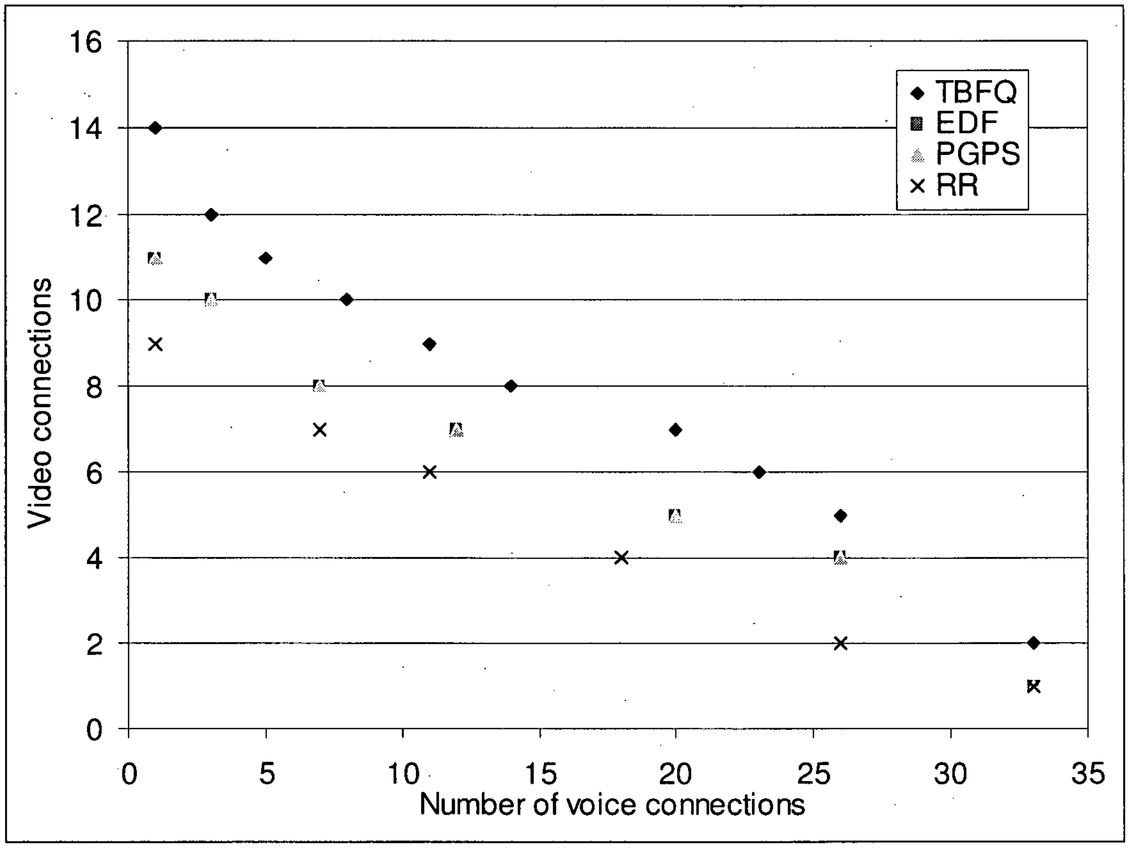

Figure 5.17 Admissible region of voice and video traffic source.

5.8 Isolation performance

The isolation performance of TBFQ is demonstrated in Figure 5.18 where only a time segment of the simulation is shown in the figure. We modified the traffic parameters so that their rate profiles are increased. We load the system to 94% with a malicious video connection (connection 1) and 5 well-behaved video connections (connections 2 to 6). The video connection 1 has an average rate of 408Kb/s and peak rate of 1024Kb/s, and each of the remaining video connections is modified to have an average rate of 204Kb/s and peak rate of 512Kb/s. Token rate of 512Kb/s is assigned to each connection, so connection 1 is the 'malicious' source. By assigning the token rate close to the peak rate (450Kb/s) for connections 2 to 6, the packet delay performance is expected to be quite good. However, connection 1 is purposely under-provisioned and the result shows poor delay performance as expected. The excess traffic from connection 1 (malicious) does not affect the delay performance of other well-behaved connections.

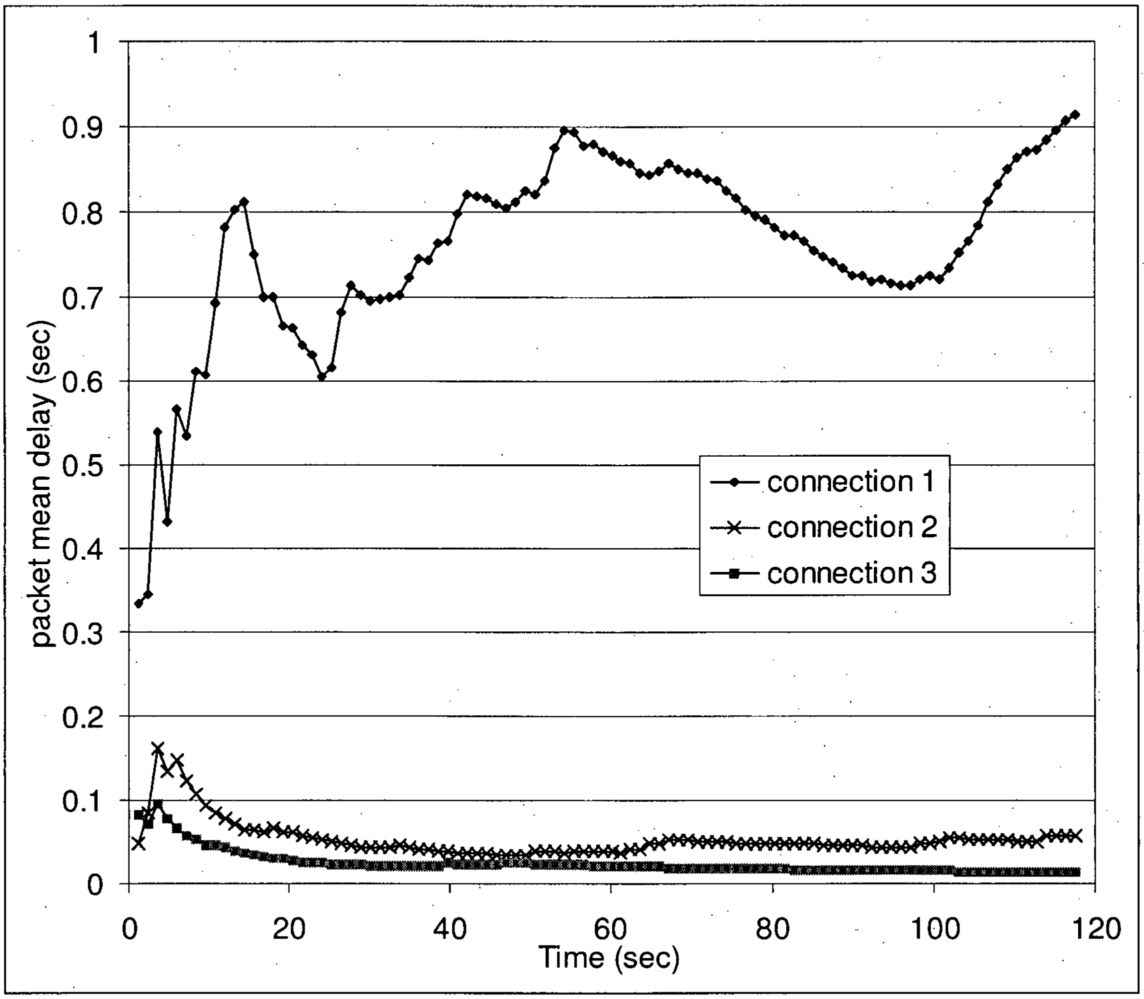

Figure 5.18 Isolation performance between well-behaved and 'malicious' video traffic source.

This can easily be explained. As connection 1 generates a large burst, it continues to borrow tokens from the bank. During that time, other connections may not be able to borrow from the bank but they will at least be served by their token rate which satisfies their peak rate requirement. When connection 1 reaches the *debt limit*, no token is allowed to be borrowed; however, it will continue to be served at its own token rate. The remaining connections will enjoy at least their minimum reserved rate plus a share of excess bandwidth, which is now share by only 5 connections instead of 6. Hence, more services can be provided.

5.9 Location-dependent error conditions

The CIF technique proposed in [75] is a well-known technique used for location-dependent error conditions. To compare the performance of TBFQ with that of CIF, we use the similar simulation parameters such as source properties and error patterns as in [75]. Within 200 seconds of simulation time, errors occur only during the first 45 sec leaving enough error-free time to demonstrate the long term fairness property. The error model used is the simple periodic error burst. Error pattern 1 represents a periodic error burst of 1.6 sec with 3.2 sec of intermediate error-free time. Error pattern 2, a less severe error pattern, represents a periodic error of 0.5 sec with 5.5 sec of intermediate error-free time. The source properties are shown in Table 5.4.

In Figure 5.19, we demonstrate services received according to their error pattern. For better resolution, only 100 seconds of simulation is shown. It is apparent that there is fluctuation of service in the first 45 seconds when error is present. During this time when there is error in the link, Data-1 and -3 receive less service, therefore they are at the bottom of the curves. After their link becomes error-free, their service converges very rapidly along with the data flows, which demonstrate the ability of the algorithm to converge rapidly to the guaranteed throughput. It also demonstrates long-term fairness according to the definition.

Table 5.4 Properties of the 6 flows in the simulations.

	Guaranteed rate	**Source model**	**Error**
Audio	64 Kbps	CBR	None
Video	125 Kbps	CBR	None
Data-1	200 Kbps	Greedy	Pattern 1
Data-2	200 Kbps	Greedy	Pattern 2
Data-3	200 Kbps	Greedy	Pattern 1
Data-4	200 Kbps	Greedy	None

Figure 5.20 gives a different view by showing the difference between services received by each data traffic stream under the conditions in Table 5.4 and services that would have been received by each data traffic stream under error-free condition. Obviously, there is not much change that can be seen for Data-4. And Data-1 and -3 received less service during the error period than Data-2.

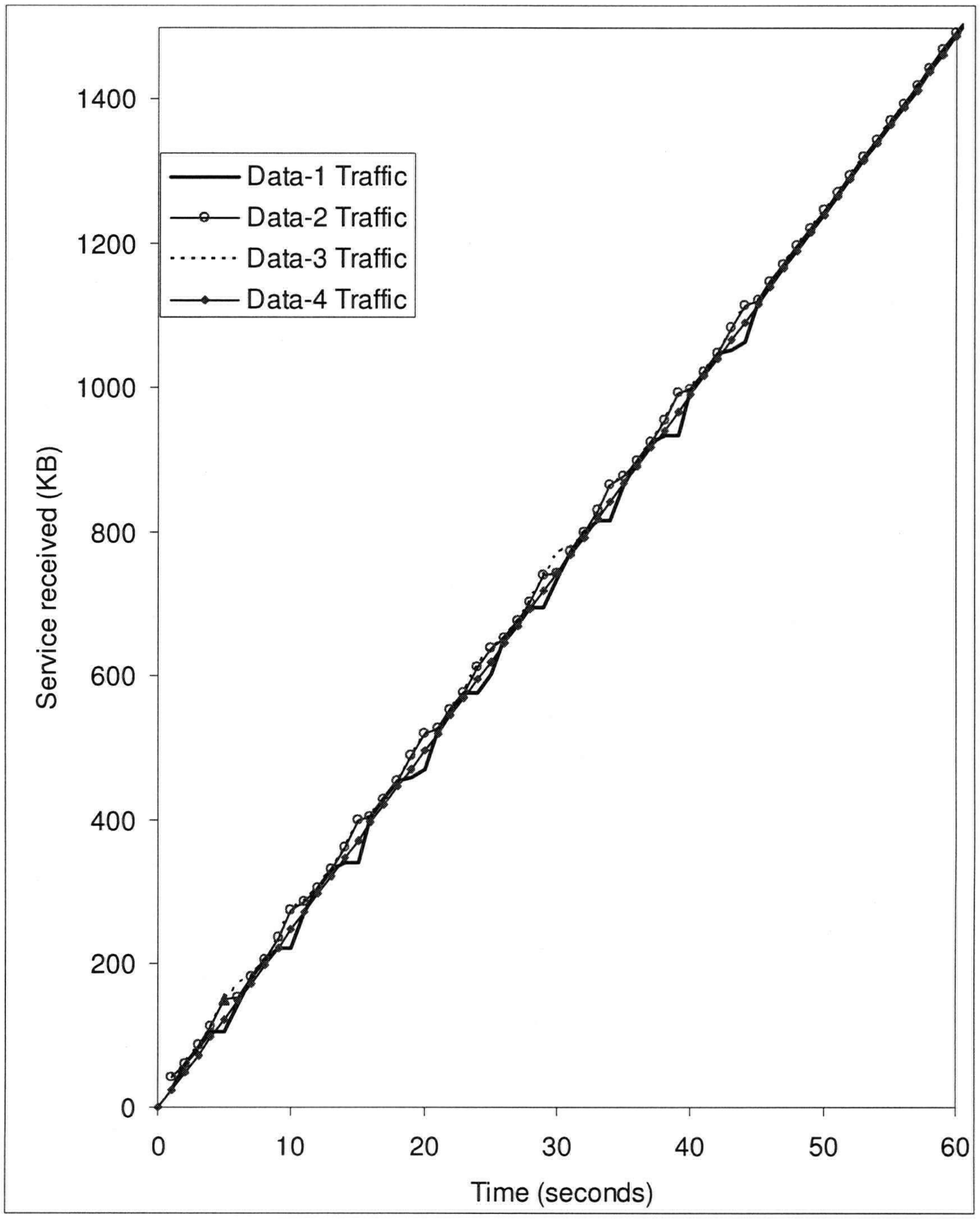

Figure 5.19 Serviced received by each data traffic source. Note that Data-1 and -3 are at the bottom among the curves.

We can also notice from Figure 5.20 that both Data-1 and -3 received very similar amount of service during the error period and that Data-4 is not affected. This shows the short-term fairness capability of the algorithm, as similar traffic under similar error link condition should receive the similar service treatment. This also shows good isolation between data flows.

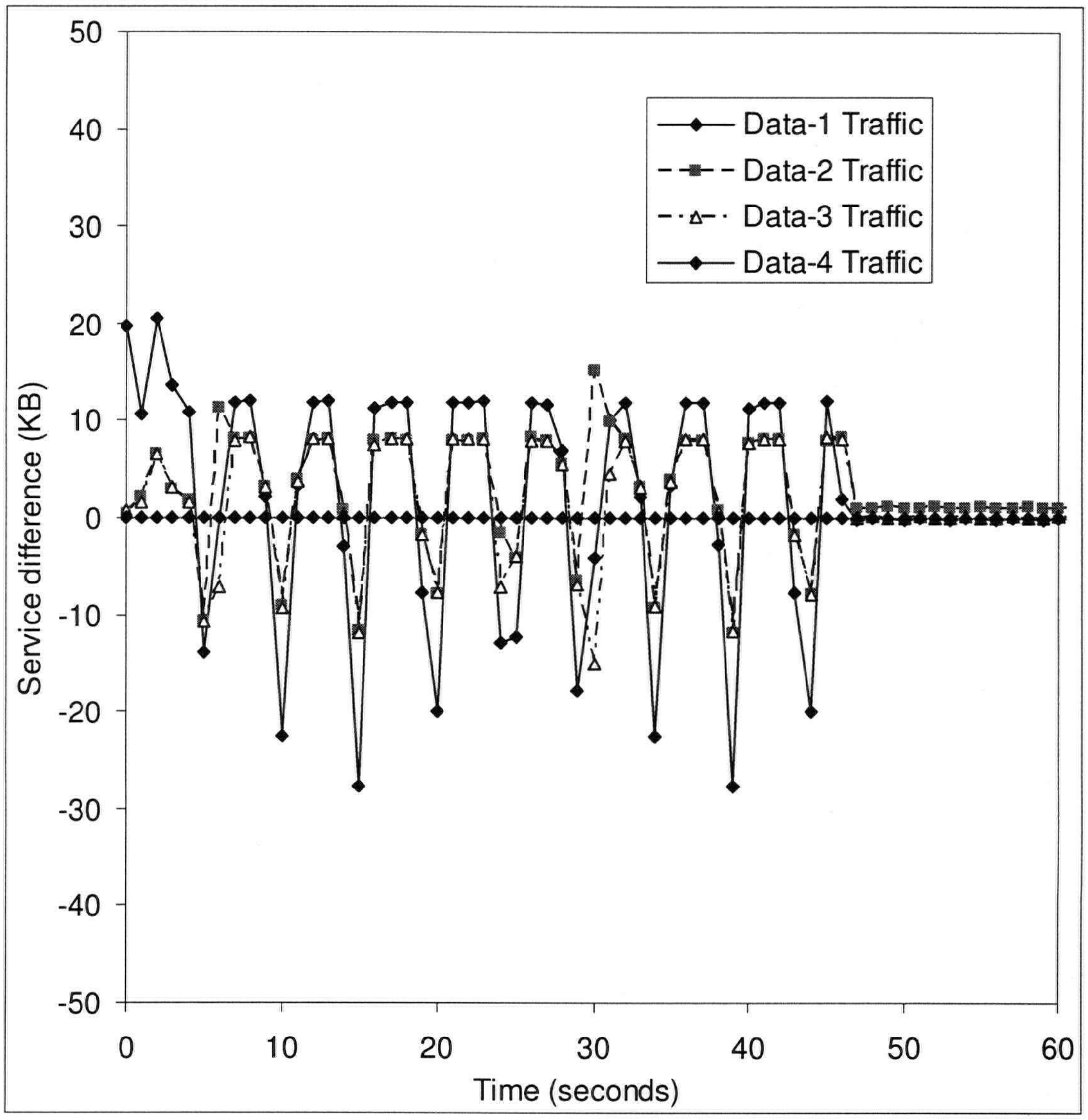

Figure 5.20 Difference between the actual services received by the data traffic source under error conditions in Table 5.4 and the expected amount of service under error-free condition.

The results we have obtained in both Figure 5.19 and Figure 5.20 are quite similar to that of the CIF algorithm found in [75] when $\alpha=0$. In CIF, $\alpha=0$ means that leading sessions receive no service as long as there exists a lagging error-free session in the system. However, it was stated that although it also has the ability of fast convergence to the guaranteed throughput, such aggressive compensation (i.e. $\alpha=0$) is not desirable because it affects the fairness. We have demonstrated that although TBFQ does not have the exact mechanism, it is able to maintain the same property without sacrificing fairness.

Chapter 6 TBFQ IN WCDMA HIGH SPEED DOWNLINK PACKET ACCESS (HSDPA)

6.1 Introduction

Standards released by the Third Generation Partnership Project (3GPP) prior to Release 5 have defined a WCDMA system that can support user data rates of up to 2MB/s under ideal conditions [95]. However, user appetites for higher data rates are quickly questioning the ability of a UMTS system to attract and sustain a subscriber population. Promises of higher data rates from technologies such as Wi-Fi are pressuring wireless providers to offer competing services with equivalent bandwidth capabilities. Release 5 [95] and Release 6 [96] of the UMTS standard now include a specification for a new high speed downlink packet access system that will be capable of providing data rates of up to 14.4Mb/s under ideal conditions. The techniques employed to achieve the higher data rates while maintaining compatibility with currently available equipment include Adaptive Modulation and Coding (AMC), Hybrid Automatic Repeat Request (HARQ), Fair Scheduling and Fast Cell Site Selection (FCSS).

The idea of HSDPA is to increase the possible downlink data rate by increasing the spectral efficiency. The focus on the downlink data rate is originated in services that demand high data rate such as Internet access and file downloads. The first phase of HSDPA has been specified in 3GPP release 5. Phase one introduces new basic functions and is aimed to achieve peak data rates of 14.4 Mbps. Newly introduced are the High Speed Downlink Shared Channels (HS-DSCH), the adaptive modulation QPSK and 16QAM and the High Speed Medium Access protocol (MAC-hs) in the Node-B. The second phase of HSDPA is currently being specified in 3GPP release 6 [96] and is aimed to achieve data rates of up to 28.8 Mbps. The third phase of HSDPA which still is a long way down the road will concentrate on the air interface. It will introduce a new Air Interface with Orthogonal Frequency Division Multiplexing (OFDM) and higher modulation schemes. Phase three of HSDPA aims at data rates of up to 50 Mbps. In order to support HSDPA, new physical channels, logical channels as well as changes to protocols have been added to the UMTS Specification.

HSDPA achieves its high speeds through similar techniques that amplify EDGE performance past GPRS, including higher-order modulation, variable coding and incremental redundancy, as well as through the addition of powerful new techniques such as fast scheduling. HSDPA takes WCDMA to its fullest potential for providing broadband services, and is the highest-throughput cellular-data capability defined. The higher spectral efficiency and higher speeds not only enable new classes of applications, but also support a greater number of users accessing the network.

HSDPA achieves its performance gains from the following radio features:

- High speed channels shared both in the code and time domains
- Short transmission time interval (TTI)
- Fast scheduling
- Higher-order modulation
- Fast link adaptation
- Fast hybrid automatic-repeat-request (ARQ)

To support HSDPA, new physical channels have been added to the UMTS specification:

Physical Channel Changes

- High Speed Physical Downlink Shared Channel (HS-PDSCH) is the transport mechanism for the new HSDPA logical channels. This channel will be both time and code shared between users attached to a Node-B.
- High Speed Dedicated Physical Control Channel (HS-DPCCH) is an uplink channel that carries packet acknowledgment signaling for each transport block and a Channel Quality Indicator (CQI) used by the Node-B to perform AMC.

Logical Channel Additions:

- HS-DSCH – High Speed Downlink Shared Channel - provides the logical transport mechanism for data transfer.
- HS-SCCH – High Speed Shared Control Channel – provides timing and coding information to the User Equipment (UE). This allows the UE to listen to the HS-DSCH at the correct time and using the correct codes to allow successful decoding of received data.

The HS-DPCCH is an uplink control channel. It carries signaling and channel quality information from the User Equipment (UE) to the Node-B. This information is used by the Node-B to perform the adaptive modulation, and coding of the above described HS-PDSCH. The transmitted signaling information also contains acknowledgements or non-acknowledgements for each received user data block.

The HS-DSCH provides the logical transfer mechanism for the data that is transported on the physical channel HS-PDSCH. The HS-SCCH is a downlink signaling channel providing information to the UE. The information provided is around timing and coding and amongst others such as the channel code set, the modulation scheme, the transport block size and the UE identity. This data enables the user equipment to “listen” to the HS-DPCH in an optimized way, at the right time and with the correct codec in order to decode the received data. It enables a connection without wasting precious radio resources.

Adaptive Modulation and Coding (AMC) is one of the major changes in HSDPA. In UMTS release99, modulation techniques were applied to provide a reliable connection under changing environmental conditions. With decreasing signal to noise ratio more errors are transmitted with the signal. The higher the coding rate applied, the better the chances of an UE to decode the original data. But on the other hand, the higher the coding rate, the more bits are sent to transmit the information which means that more bandwidth is used.

HSDPA uses both the modulation used in WCDMA, namely Quadrature Phase Shift Keying (QPSK) and under good radio conditions, an advanced modulation scheme, 16 Quadrature Amplitude Modulation (16 QAM). The benefit of 16 QAM is that four bits of data are transmitted in each radio symbol as opposed to two with QPSK. 16 QAM increases data throughput, while QPSK is available under adverse conditions.

Depending on the condition of the radio channel, different levels of forward error correction (channel coding) can also be employed. For example, a three quarter coding rate means that three quarters of the bits transmitted are user bits and one quarter are error correcting bits. The process of selecting and quickly updating the optimum modulation and coding rate is referred to as fast link adaptation. This is done in close coordination with fast scheduling. Fast scheduling exploits the short TTI by assigning channels to the users with the best instantaneous channel conditions, rather than in a round-robin fashion. Since channel conditions vary somewhat randomly across users, most users can be serviced using optimum radio conditions, and can hence obtain optimum data throughput. The system also makes sure that each user receives a minimum level of throughput.

Table 6.1 shows the different throughput rates achieved based on the modulation, the coding rate, and the number of HS-DSCH codes in use. Note that the peak rate of 14.4 Mbps occurs with a coding rate of 4/4, 16 QAM and all 15 codes in use.

Table 6.1 HSDPA Throughput Rates

Modulation	Coding Rate	Throughput with 5 codes	Throughput with 10 codes	Throughput with 15 codes
QPSK	1/4	600 kbps	1.2 Mbps	1.8 Mbps
	2/4	1.2 Mbps	2.4 Mbps	3.6 Mbps
	3/4	1.8 Mbps	3.6 Mbps	5.4 Mbps
16 QAM	2/4	2.4 Mbps	4.8 Mbps	7.2 Mbps
	3/4	3.6 Mbps	7.2 Mbps	10.7 Mbps
	4/4	4.8 Mbps	9.6 Mbps	14.4 Mbps

In addition to the code multiplexing of traditional W-CDMA channels, where user data is transmitted via dedicated channels, HSDPA also introduces time multiplexing. This means that several users share the same channel and at times where one user is not using an available resource it is becoming available to others. The reasoning behind this approach is that user traffic is becoming more of a bursty nature, so that a large number of users can use the same time-multiplexed channel and efficiently use the available radio network resources. Figure 6.1 illustrates different users obtaining different radio resources.

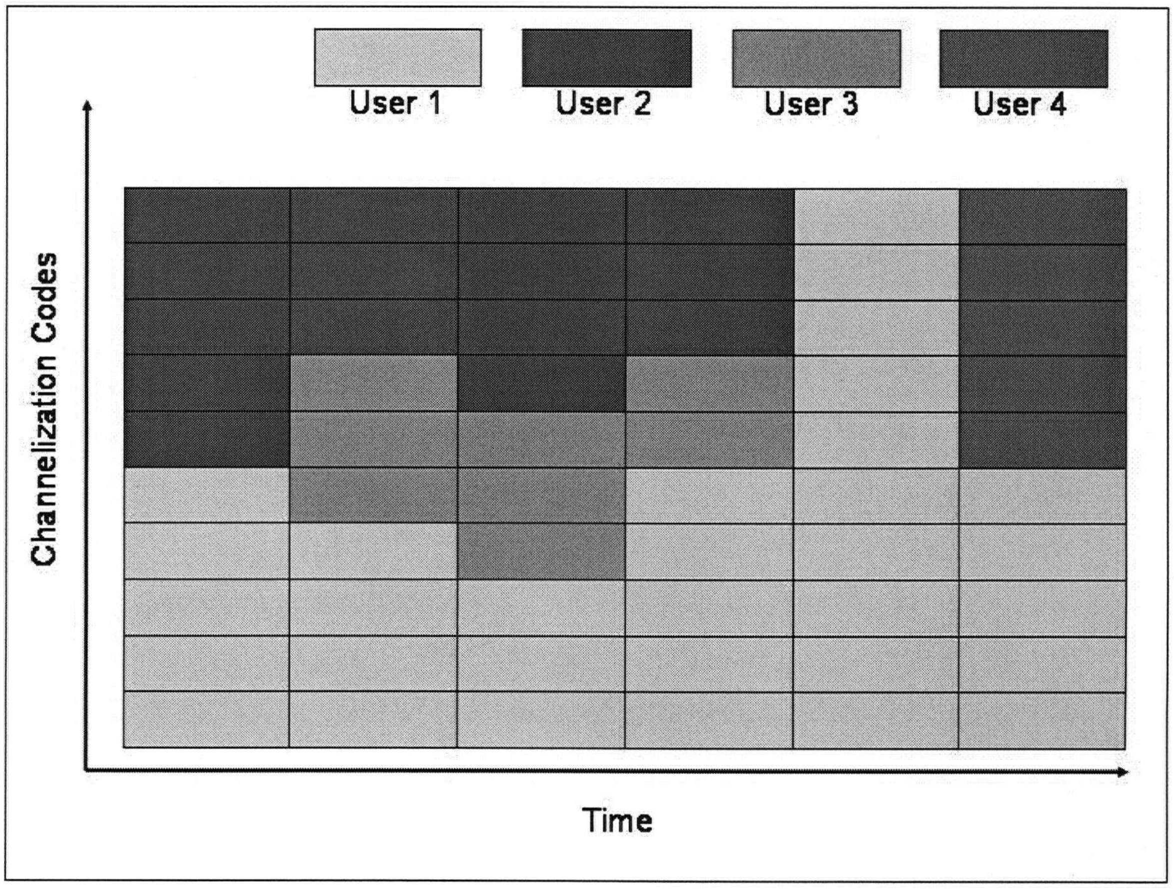

Figure 6.1 Time and code multiplexing of HSDPA downlink shared channels

These features function as follows. First, HSDPA uses high speed data channels called High Speed Downlink Shared Channels (HS-DSCH). Up to 15 of these can operate in the 5 MHz WCDMA radio channel. Each uses a fixed spreading factor of 16. User transmissions are assigned to one or more of these channels for a short transmission time interval of 2 msec, significantly less than the interval of 10 to 20 msec used in WCDMA. The network can then readjust which users are assigned which HS-DSCH every 2 msec. The result is that resources are assigned in both time (the TTI interval) and code domains (the HS-DSCH channels). The short TTI allows for multiple retransmissions in the event of receive errors. Furthermore, it allows reasonably sized payloads and minimal wasted bandwidth in the event a user cannot fill the allocated channel capacity.

To further optimize the HS-DSCH channel, limited sharing between UE is provided by the use of channel coding. By allocating a limited number of spreading codes to the HS-DSCH, any user that cannot fill the channel capacity can share the remaining capacity rather than waste the unused portion. Two UEs will be assigned to a HS-DSCH TTI but each will be given a different spreading code for data recovery.

Hybrid Automatic Repeat Request (HARQ)

ARQ is a system where a receiving station will send an acknowledge (ACK) message to the sending station when a data packet has been successfully received. The success of the reception is usually by comparison to a checksum (CRC) sent with the data. In the event that the checksum calculated by the receiving station does not match the checksum included with the data packet, the receiving station will send a negative acknowledgment (NAK) to the sender and discard the erroneous data packet. This will cause the sending unit to retransmit the erroneous data packet. In HSDPA, this scheme has been modified to minimize retransmission by the two methods below.

In Chase Combining (CC), when the UE detects that a data packet has been received in error, it will send a NAQ to the sending Node-B. Rather than discarding the erroneous packet, it will be stored. In the event the retransmitted packet is also received in error, the previous packet and the current packet will be combined in an attempt to recover from the data errors. Each time a packet is resent, the same coding scheme is used. Eventually the packet will either be received without error, the UE will recover from the errors or the maximum number of resends will be reached and error recovery will be left to higher protocol levels.

Incremental Redundancy (IR) is similar to Chase combining however retransmitted data is coded using additional redundant information to improve the chances that the packet will be received either without errors or with enough errors removed to allow combing with previous packets to allow error correction.

Fast Cell Site Selection (FCSS)

While a UE travels through a network, it is possible for more than one Node-B to be in communications with the UE. A UE will construct a list called an Active Set where it keeps all node-B stations that could be used for communications. Fast cell site selection allows a UE to scan its active set and select the cell with the best current transmission characteristics to transmit the data packets. The advantage of this system is that higher data rates can be achieved by always using the best supporting node-B for data transmission.

6.2 New Node-B Requirements

For non-HSDPA support, node-B stations are connected to Radio Network Controller (RNC) that provide scheduling, coding parameters and retransmission services to UE devices. To support HSDPA, these parameters will need to be determined based on the instantaneous channel conditions as reported by the UEs. The delays associated with the forwarding of channel data to the RNC for processing, coupled with the burden of an RNC having to service multiple node-B stations, require the node-B to handle the work rather than the RNC.

The new node-B functions will not replace those existing in the RNC. Radio Link Control (RLC) and Medium Access Control (MAC) functions will still need to exist in the RNC to provide packet delivery order services as well as non-HSDPA services. Handoffs must be supported by the RNC to assure no loss of data as well as FCSS. The physical connections between network components will not change and is illustrated in Figure 6.2.

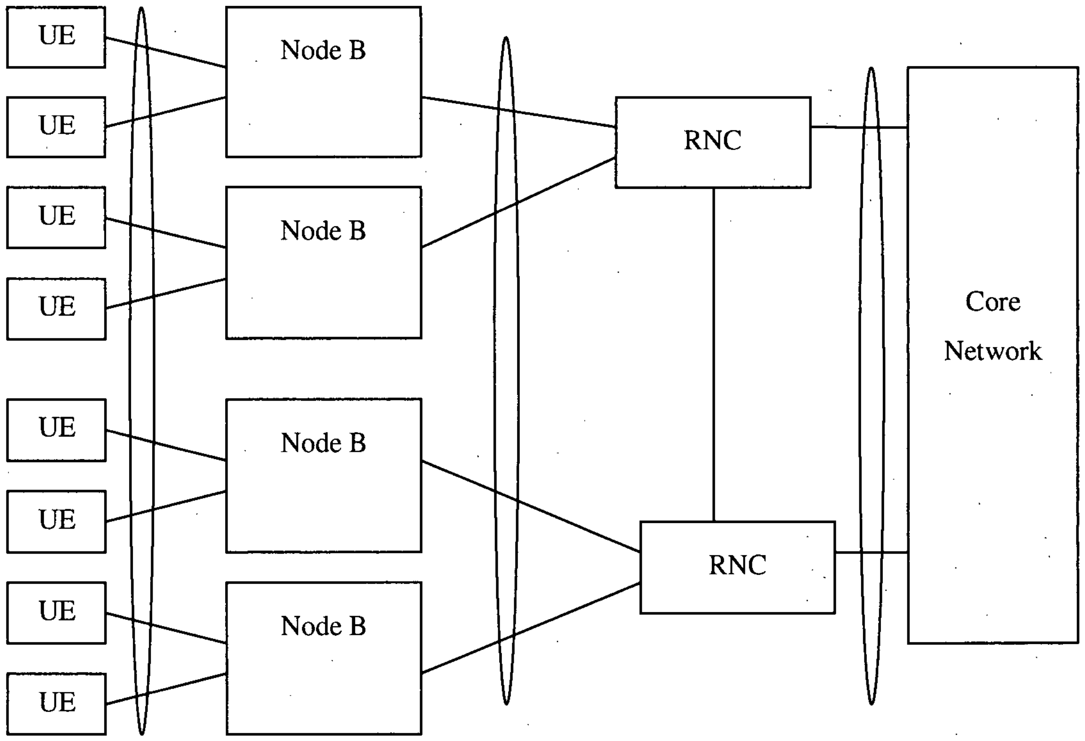

Figure 6.2 Network Element Layout

6.3 Protocol Architecture Changes

Support of the new HSDPA capabilities will require updates to the current protocol architecture. These changes will be implemented as additional protocol capabilities in order to preserve backward compatibility. The new protocol architecture for the HS-DSCH is illustrated in Figure 6.3 [98].

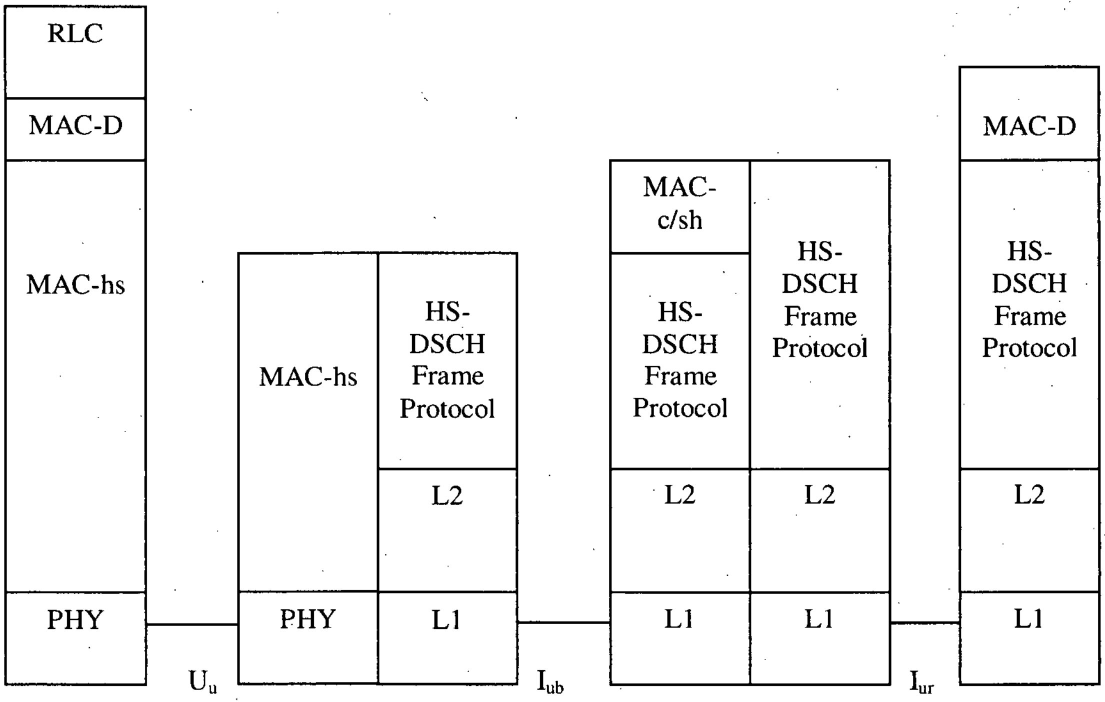

Figure 6.3 HSDPA Protocol Architecture

The main changes were introduced for the MAC protocol [97]. The MAC decides on which channel the Protocol Data Units (PDU's) will be transmitted. Traditional MAC protocol resides in the Radio Network Controller, whereas for HSDPA, the High Speed Physical Downlink Channel the High Speed MAC (MAC-hs) was introduced. The MAC-hs resides in the node-B. It takes care of the transport block scheduling, channel allocation and the transport format selection. Further tasks of the MAC-hs include: Adaptive Modulation and Coding (AMC), Fast packet scheduling mechanism, and Hybrid Automatic Repeat Request (HARQ).

6.4 MAC Layer Architecture

The MAC layer for UMTS is comprised of several entities, each supporting MAC functions for their associated transport channels:

- MAC-b supports broadcast channels
- MAC-c/sh supports the common and shared channels including the paging channel, forward and random access channels, common packet channel, downlink shared channel.
- MAC-d supports dedicated transport channels
- MAC-hs supports the high speed downlink shared channel

Only the MAC entity added to support HSDPA will be discussed here. Readers interested in the details of the other MAC entities are referred to [97]. The MAC-hs performs distinct functions depending on whether it is located on a UE or Node-B. Figure 6.4 [98] illustrates the composition of the MAC-hs on the UE.

The HARQ entity is responsible for generating ACK or NAK responses to received packets. Re-order Queue Distribution routes packets to the correct reordering queue based on Queue ID. This supports multiple services requesting data over the HSDPA feature. The reordering entity is responsible for collecting consecutive protocol data units (PDU) based on the Transmission Sequence Number (TSN). MAC-hs PDU will not be forwarded to the disassembly function until all missing PDU units with lower TSN assignments have been received. The disassembly entity is responsible for removing the MAC-hs header and all padding bits. The MAC-d PDUs extracted from the payload are forwarded to the MAC-d for processing.

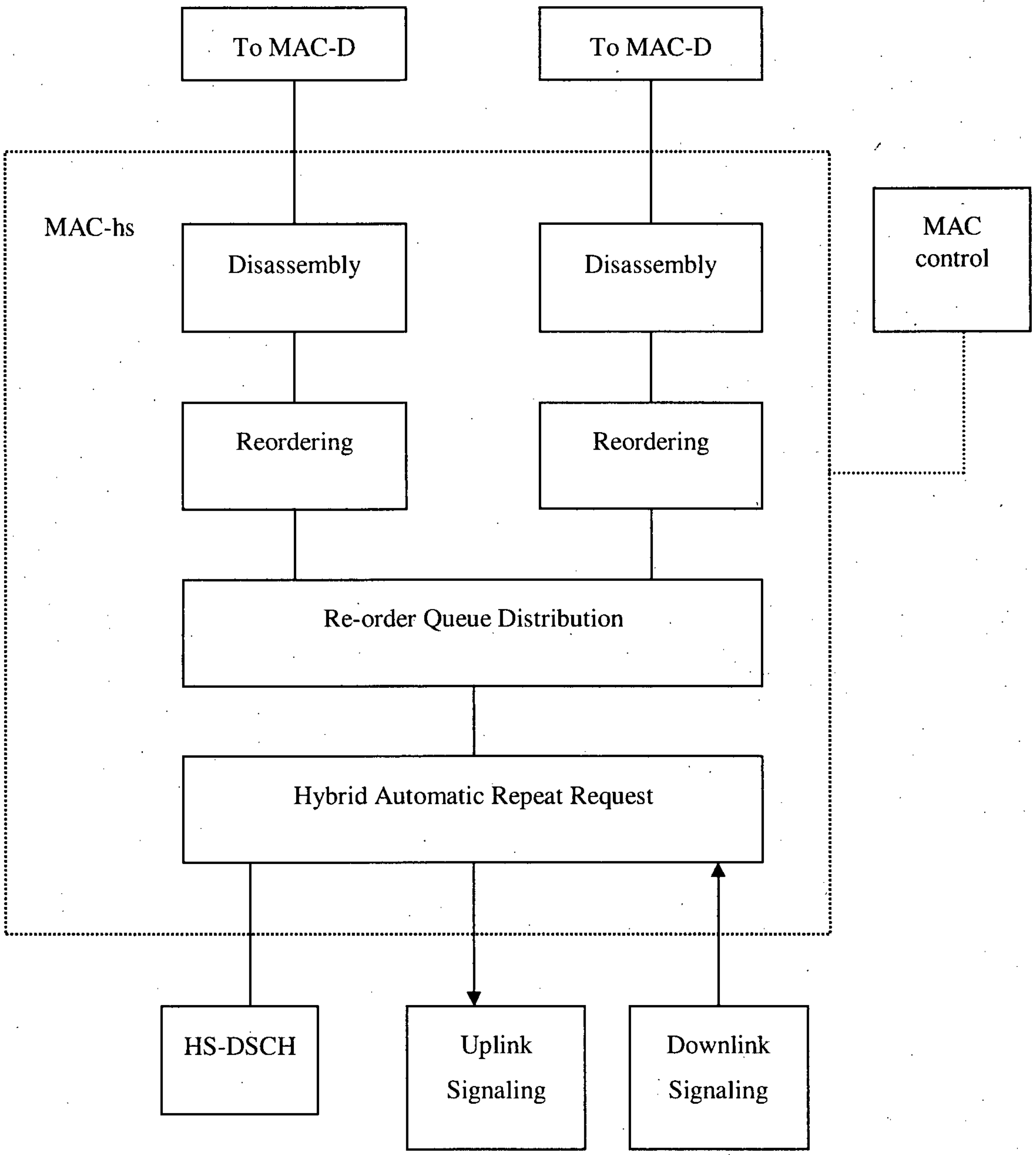

Figure 6.4 MAC-hs on the UE

Figure 6.5 [98] illustrates MAC-hs on the Node-B. The MAC control provides a MAC PDU flow control function between the MAC-hs and MAC-c/sh. The intent is to allow a coordinated approach to the use of the HS-DSCH and other available data channels. Channel selection will depend on loading of the HS-DSCH and will be dynamically determined. Scheduling and priority handling entity manages data flow by assigning a Queue ID and TSN for each new MAC-hs PDU being serviced. It will also re-queue a PDU if retransmission is requested by the HARQ. The HARQ entity is responsible for requesting retransmission from the correct priority queue in the event a NAK has been sent by the UE for any sent

MAC-hs PDU. TFRC selects the appropriate transport format and resource for the data transmitted on the HS-DSCH.

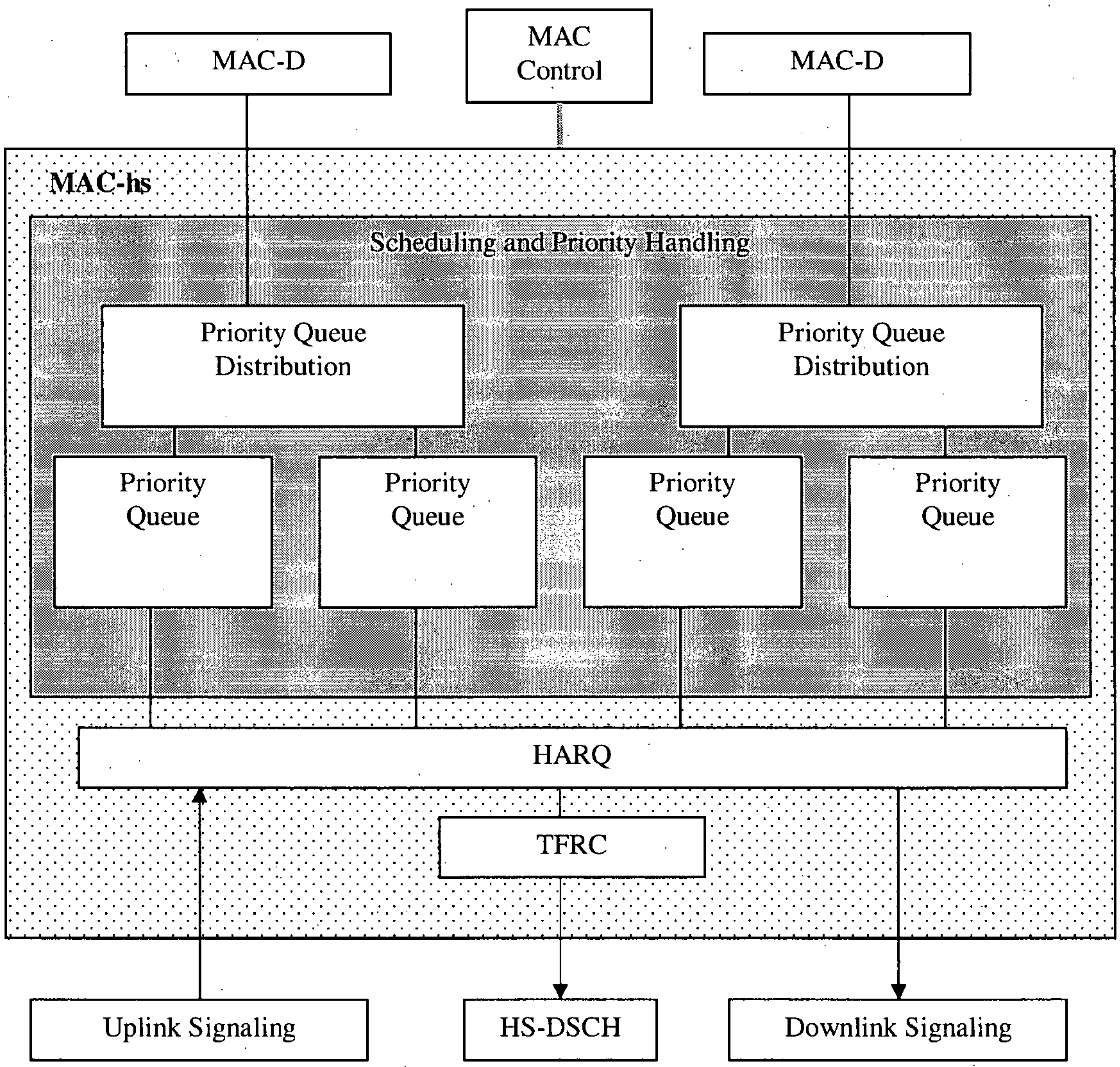

Figure 6.5 MAC-hs on the Node-B

In HSDPA, the UE is actively feeding back information about the channel conditions which is used by the Node-B to determine Modulation and coding scheme. For each Transmit Time Interval (TTI) the UE feedback is taken into account the best possible modulation and coding is chosen and the highest possible transmission rate is obtained. The Fast scheduling mechanism handles the logical channel resources and determines which particular user should be served within a 2ms time interval. This mechanism also takes into account the information sent by the individual UE's. The knowledge of the instantaneous quality of a channel makes it possible to avoid sending data packets during channel fades and instead schedule a UE's that are in better conditions. The challenge for the packet scheduling function is to optimize the cell capacity and at the same time fulfill QoS requirements. The MAC-hs PDU structure is illustrated in Figure 6.6 [98].

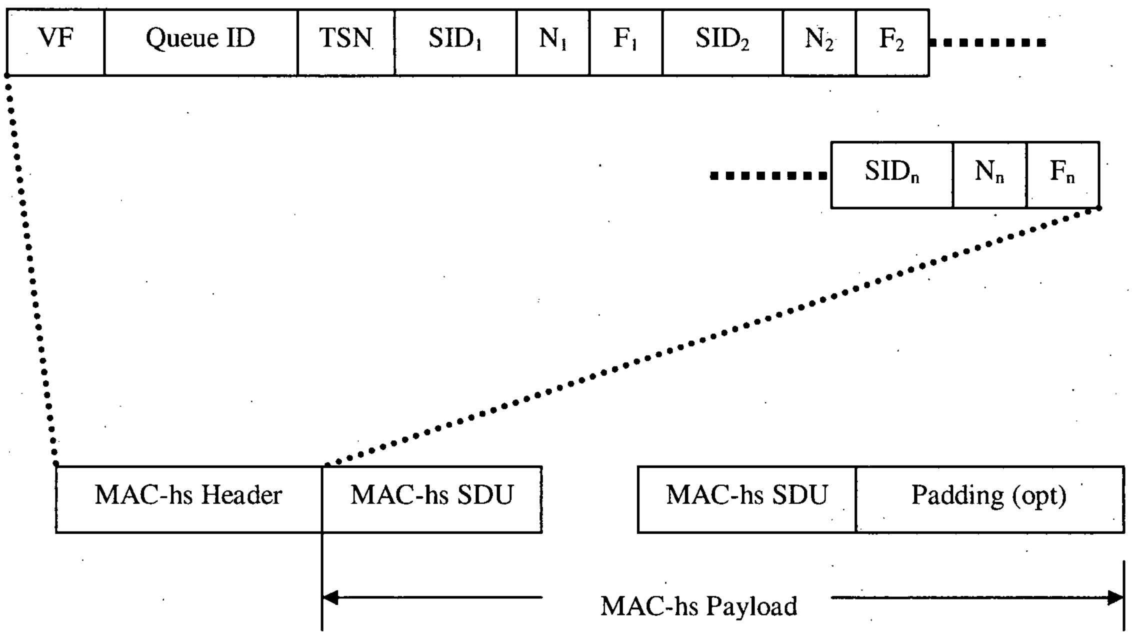

Figure 6.6 MAC-hs Protocol Data Unit

- VF – Version Flag – a one bit field that at this time is reserved for future use.
- Queue ID – Provides a three bit identification of the reordering queue in the UE and is needed to support multiple simultaneous data applications.
- TSN – Transmission Sequence Number – a six bit field that is used by the UE to reassemble data streams into correct sequence order prior to forwarding to higher protocol layers.
- SID – Size Index Identifier – a three bit field that indicates the size of a set of consecutive MAC-d PDUs
- Nn – Indicates the number of consecutive MAC-d PDUs of equal size that have been transmitted. This field is seven bits long.
- Fn – Flag field is a one bit field that indicates if more SID fields are contained in the current MAC-hs header. If the field is set to a '1', it indicates the end of the MAC-hs header with the following field starting the MAC-hs SDUs.
- MAC-hs SDU is identical to a MAC-d PDU (encapsulation of the MAC-d PDU into the MAC-hs PDU).

6.5 TBFQ Scheduling in HSDPA

Since the HS-DSCH is shared between UEs, a method of scheduling is required to provide service to all UEs. This becomes very complicated since the first thought would be to service users that were capable of receiving data at the highest data rate. This unfortunately would not be fair to users on the fringe of cells as it would lead to starvation of users on the fringes of cells. Several fast scheduling algorithms have being analyzed for HSDPA [99]. They are Proportional Fair (PF), Maximum C/I (MaxCI), and Round Robin (RR).

MaxCI serves the user with the most favorable channel condition in every TTI. This is the most efficient way to use the channel; the total throughput will be maximized. The drawback is that the system is unfair. Users with good channel conditions are served all the time at the expense of users with less favorable channels.

The PF algorithm takes both throughput and fairness into account [100]. It serves the user with the largest relative channel quality:

$$\begin{gathered} P_i = \frac{R_i(t)}{\lambda_i(t)} \\ i = 1..N \end{gathered} \tag{6.1}$$

where $R_i(t)$ is the instantaneous data rate experienced by user i, and $\lambda_i(t)$ is the averaged user throughput. To accelerate the convergence of the stochastic approximation [101], the averaging process is calculated as,

$$\lambda_i(n) = \left(1 - \frac{1}{T_c}\right) \cdot \lambda_i(n-1) + \frac{1}{T_c} \cdot R_i(n) \tag{6.2}$$

where $\lambda_i(n)$ represents the stochastic approximation to the user throughput, $R_i(n)$ the user data rate in the current TTI, and T_c is the time constant (in TTIs) of the low pass filter. $\lambda_i(n)$ is updated every TTI with $R_i(n)$ equal to zero if the user is not served, and with a data rate equal to the transmitted bits divided by the TTI duration if the user is scheduled and successfully receives the coded word. In [100], the averaging window length is selected to be 1.6 seconds ($\approx 10\lambda$ at 3km/h) to ensure that the process is long enough to average out the fast fading variation without masking shadow fading effects.

In addition to comparing TBFQ with the aforementioned algorithms, we introduce two additional modifications of the TBFQ, namely, TBFQ+maxCI, and TBFQ+PF, to take advantage of the CQI received from the UEs. The two modified TBFQ algorithms are augmentation of maxCI and PF to the

TBFQ algorithm described in Chapter 3. In TBFQ+maxCI, packets are admitted into the output queue according to the TBFQ algorithm. After packets are admitted, maxCI scheduling is performed to determine the order of packets to be broadcasted. For TBFQ+PF, it would be PF instead of maxCI. Figure 6.7 shows the model of the two algorithms.

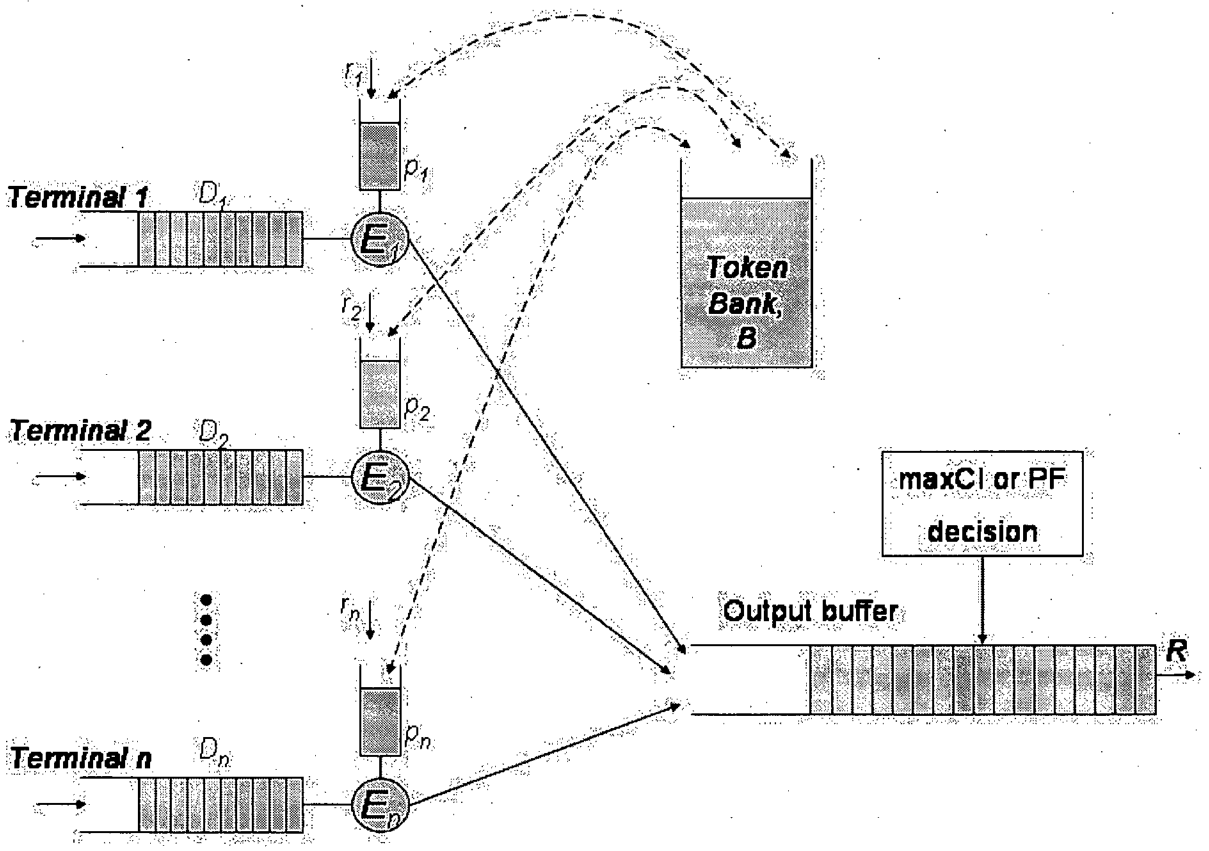

Figure 6.7 Model of TBFQ+maxCI or TBFQ+PF

6.6 Simulation configuration

The cell layout of the simulation is shown in Figure 6.8. There are two rings of cells and total of 19 cells. The center cell is the one that is analyzed, the others cells are served as interference sources to the center cell.

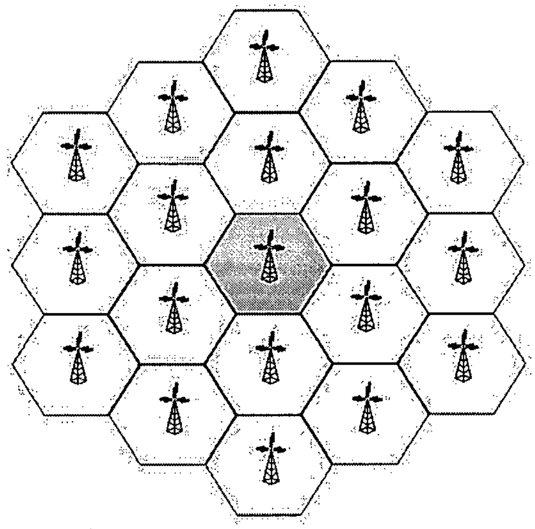

Figure 6.8 Cell layout of the simulation experiments

For the propagation loss, we use the *absolute mean path loss* (in dB) found in [102]:

$$\overline{L}(d)[dB] = L_B(d_0) - 10 \cdot n \cdot \log_{10}(d / d_0) \tag{6.3}$$

where, n is the path loss exponent; typical range of n is $3.5 \leq n \leq 5$, d is the distance between transmit and receive antenna, d_0 is the reference distance or free space propagation corner distance; typically range between 1 and 3 meters, and $L_B(d_0)$ is the free space path loss in dB from the transmitter to the reference distance and is defined as,

$$L_B(d_0)[dB] = +27.56 - 20\log f[MHz] - 20\log d_0 \tag{6.4}$$

The above model only takes into account the distance between the transmitter and the receiver. However, the path loss at a given distance from the transmitter is not constant, due to environmental obstructions. That is, all hosts at distance d from the transmitter do not see the same signal attenuation. Between the transmitter and a receiver A there may be a hill, whereas between the transmitter and another receiver B, the terrain may be flat. The shadowing caused by such obstructions is modeled by the log-normal model. As per the log-normal model:

$$P_L(d)[dB] = \overline{L}(d)[dB] + X_\sigma \tag{6.5}$$

where $\overline{L}(d)$ is the mean path loss at distance d as in (6.3), and X_σ is a Gaussian random variable (normal) with zero mean, and standard deviation σ.

In additional to the propagation loss model, we include a simple mobility model. When the simulation is initialized, there is N number of UEs uniformly distributed in the center cell. Each UE is moving at the speed of 3Km/h and will remain in the center cell. As an UE reaches the edge of the center cell, it changes direction by 180°.

Since data applications are the kind of services that HSDPA was designed to excel in, we will use the following data services (

Table 6.2) for the simulation: MMS (multimedia messaging services), EMAIL, and WWW (web browsing).

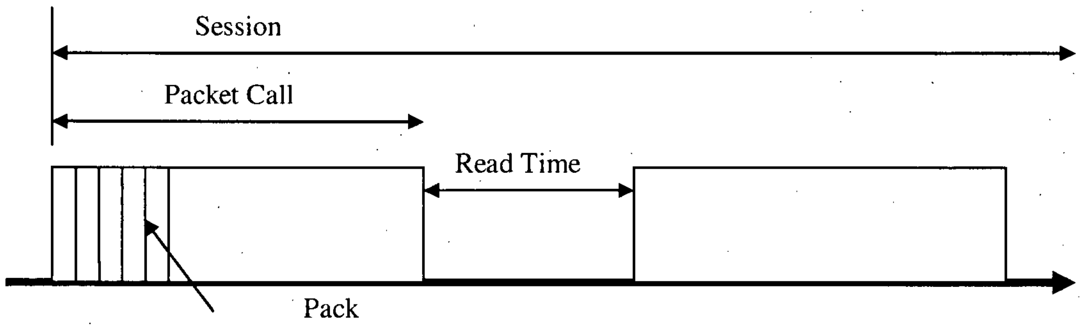

Figure 6.9 Data service traffic model parameter definition

Table 6.2 Data services traffic model and QoS requirements

Application Name	**MMS**	**EMAIL**	**WWW**
Packet calls/session (Distribution)	1	Mean=3 (Geometric)	Mean=3 (Geometric)
Packet Call Size (Kbytes) (Distribution)	Mean=3, Max=20 (Pareto 1.2)	Mean=10, Max=200 (Pareto 1.2)	Mean=50, Max=1000 (Pareto 1.2)
Packet Call Read Time (sec) (Distribution)	Mean=45 (Geometric)	Mean=20 (Geometric)	Mean=7 (Geometric)
QoS Target Delivery Delay (sec)	5	10	5

Simulation parameters are shown in Table 6.3 and Table 6.4 [103]. The center cell is allocated 80% of the maximum power for the HS-DSCH, with interference coming from all the other 18 surrounding cells where maximum power is transmitted. Five AMC schemes are used with the required

SIR and the transmission rate per code (Table 6.4). The maximum data rate per code is 0.237 if QPSK ½ is used, and the corresponding SIR required is at least $-20dB$.

Table 6.3 Simulation parameters for HSDPA

System parameters	
Number of cells	19
Cell radius	500 m
TTI period	2 ms
Time slot per frame	3
Spreading factor	16
Number of HS-DSCH codes	10
Power allocated to DSCH	80%
Path loss factor, n	3.8
Standard deviation of shadowing (σ)	8 dB
Maximum cell transmission power	20 W
HARQ	None

Table 6.4 Selected MCS and parameters [103]

Modulation and coding schemes for SF=16		
MCS	Peak data rate per code	Minimum required SIR
QPSK ½	0.237 Mbps	-20 dB
QPSK ¾	0.356 Mbps	-16 dB
16QAM ½	0.477 Mbps	-9 dB
16QAM ¾	0.716 Mbps	-4 dB
64QAM ¾	1.076 Mbps	6 dB

The TBFQ scheduler parameters are shown in Table 6.5. The token rate is set to 256 Kbps which can be considered as a data user rate plan supported by the service provider.

Table 6.5 TBFQ Parameters in HSDPA simulations

Token Rate (bits/sec)	256000
Burst Credit (bytes)	500
Debt Limit (bytes)	-10000
Leaky Bucket size (bytes)	1500

6.7 Performance analysis

This section presents the analysis on the HSDPA performance with various fast scheduling algorithms. Figure 6.10 shows the cumulative distribution function (CDF) of UE throughput within the cell. There are 64 UEs in the cell, and each is considered a Pedestrian A with a mobile speed of 3Km/h [102].

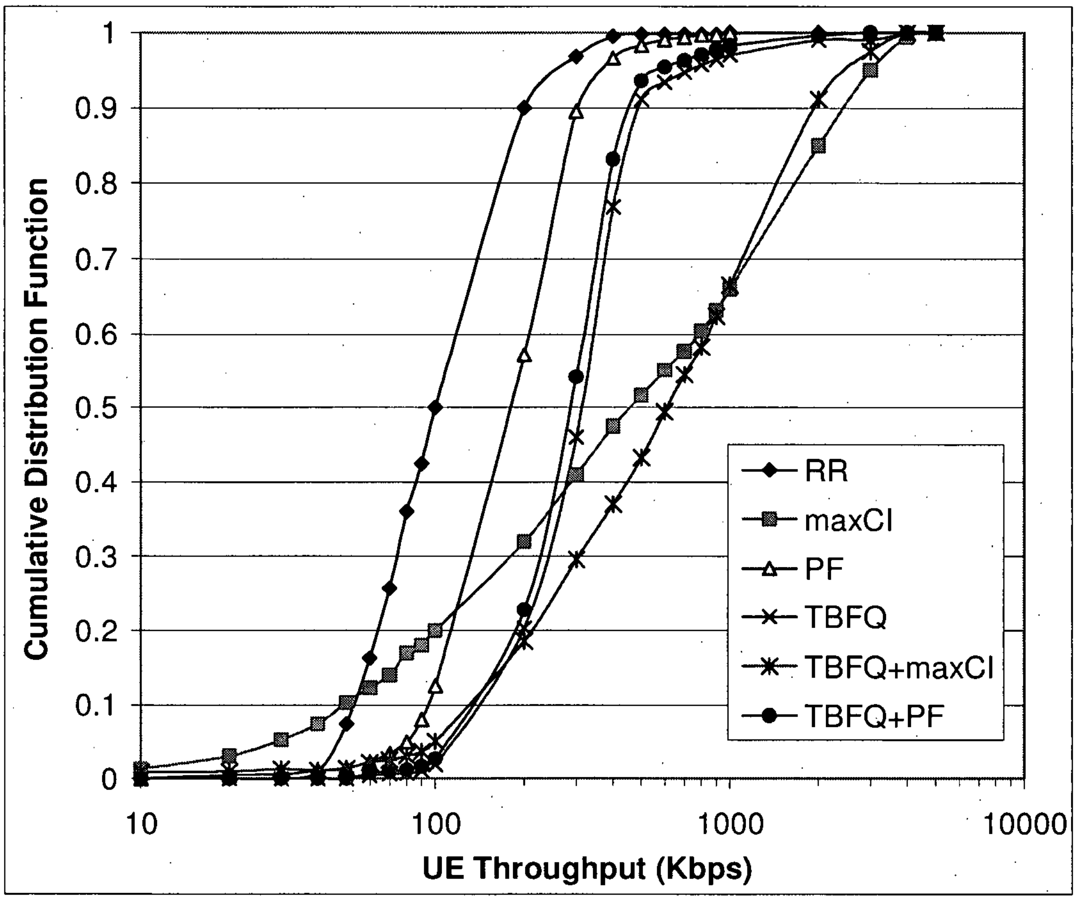

Figure 6.10 Cumulative Distribution Function of UE throughput in the centre cell.

Fairness can be determined by the slope of the cdf of user throughput. A fair throughput strategy is one that would try to give every user the same throughput regardless of their channel condition, and it has steeper distribution. Whereas a fair resource strategy would have less steep cdf because it allows higher throughputs for users with better average channel condition, thereby maximize the utilization of the same amount of resources. Both RR and PF are fair throughput strategies and outperform maxCI from 0% to 11% percentile, and 0% to 23% percentile, respectively. However, maxCI outperforms RR and PF for the rest of the cdf curve. This is because both RR and PF strive to maintain fair throughput among the users. The improvement of 11% and 23% percentile, for RR and PF respectively, are at the expense of the 89% and 77% percentile of the users. In other words, maxCI achieves much higher throughput (over 10% percentile of users above 2.5Mbps) at the expense of the lower percentile users. TBFQ is also a fair throughput strategy as demonstrated by the steepness of the distribution. It can be noted that TBFQ has overall better performance than RR and PF with slightly steeper curve which implies a fairer throughput treatment than RR and PF.

TBFQ outperforms maxCI from 0% to 36% percentile, but maxCI has better performance for the rest of the cdf curve. In maxCI, 50% of the users achieve an average throughput of 500 Kbps or less, and only about 6% of the users under TBFQ achieve 500Kbps or less. An interesting observation can be made for TBFQ+maxCI. It is an improvement over the pure TBFQ. The slope is less steep compare to pure TBFQ, because it is no longer a fair throughput strategy but a mix between a fair throughput and fair resource. Even though maxCI has better throughput above the 65% percentile, TBFQ+maxCI has the best overall performance.

TBFQ+PF has very similar performance as the pure TBFQ, although TBFQ is slightly better. 10% percentile of users achieve 90Kbps or less under the TBFQ algorithm; versus 17% in TBFQ+PF. Majority of the user achieve 500Kbps in TBFQ (91%) and TBFQ+PF (94%).

Figure 6.11 shows the cumulative distribution of packet delay experienced by all the UEs. It is quite clear that TBFQ+maxCI has the best overall performance with 50% of the users have 600 msec or less delays, and 84% percentiles are under the 2 second delay mark. With the exception of RR and PF, 90% percentiles are under the 5 sec delay mark which meets the QoS target in Table 6.2. We can also note that TBFQ+PF does not improve delay performance over TBFQ. From 84% to 100% percentile, TBFQ and TBFQ+PF have very similar delay performance.

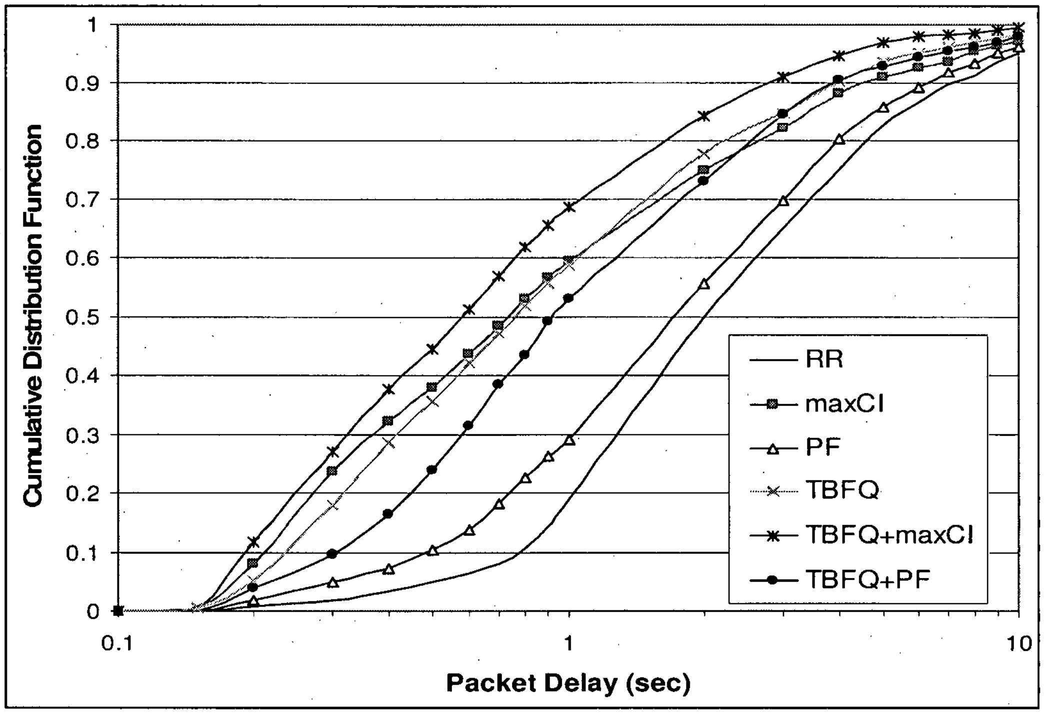

Figure 6.11 Cumulative Distribution Function of packet delay.

Chapter 7 CONCLUSIONS AND FUTURE RESEAERCH RECOMENDATIONS

7.1 Conclusions

Next generation wireless access networks will be integrated and interoperable in a seamless fashion. Preparation for such convergence has already started to take place, and projects have been initiated around the world to determine what that future will look like – for example, the WINNER project in European Union, the 4G Mobile Forum sponsored by China, and WiMAX (802.16). One of the common visions for next generation networks is a user-centric QoS paradigm where users will always be connected to the best wireless access network.

Motivated by this vision, the TBFQ wireless scheduling algorithm was proposed and developed (Chapter 3). Characteristics and performance of the algorithm were shown. The TBFQ is a work-conserving algorithm that has a complexity of O(1). Due to its simplicity in design, it is suitable for fast scheduling requirement in next generation wireless environment, and it is also effective in its ability to serve multimedia services which will prevail in next generation wireless access networks. One of the reasons that TBFQ is suitable for provisioning QoS in the wireless access networks is its *soft* scheduling capability. Traffic profile often deviates from the anticipated, simply because of the nature of wireless environment. We have demonstrated that TBFQ gracefully tolerates such traffic profile deviation and still provides minimum level of service.

In terms of fairness, we have demonstrated that TBFQ is a fair throughput strategy. It maintains the minimum throughput for each connection and distributes excess resources fairly according to usage history. The usage is normalized to each connection's reserved rate, and unlike many algorithms, it maintains a longer usage history to determine fairness. This achieves longer term fairness as oppose to instantaneous fairness – a tradeoff between being a *soft* provisioning algorithm compared to a *hard* one.

Ideally, it would be very useful to develop a closed-form equation or model that would allow us to characterize the TBFQ algorithm and determine its performance. However, it is very challenging, if not impossible, to prove such form exists because of its long term dependence of the queues in TBFQ. Feasibility studies were carried out to determine suitable methods in analyzing the system. It was concluded the recursive approach (Chapter 4) is the best methodology among the recommended ones in determining numerically the distribution of queues occupancy for voice connections, and have found that convergence occurred very rapidly.

To demonstrate that TBFQ is indeed suitable for next generation wireless access technologies, further analysis were conducted through simulation studies in both TDMA and CDMA systems (Chapter 5 and Chapter 6). A novel scheduling framework was proposed in Chapter 5 to answer the challenges of seamless scheduling across different networks. Operating under a TDMA system in both uplink and downlink directions, the results show that TBFQ has higher utilization of wireless resources compared with other algorithms when the system is in an average or highly loaded condition. It is able to maintain QoS when handling more traffic mixes. The delay, jitter, and throughput performance meet the QoS requirements of multimedia applications. The algorithm also offers exceptional isolation capability in which normal users would not be affected by malicious users within the same system. Furthermore, good performance under erroneous channel condition is another trait that makes TBFQ a suitable wireless scheduling algorithm. For mobile terminals that have just came out of location-dependent error conditions, the convergence to the normalized service is very fast and comparable to that of the CIF technique, yet without sacrificing fairness. We have also proposed a suitable CAC mechanism for the TBFQ algorithm.

An enhancement of the TBFQ was proposed in Chapter 6. The modification of the algorithm is called TBFQ+maxCI; it is a hybrid between fair throughput strategy and fair resource strategy. Both algorithms – TBFQ and TBFQ+maxCI – allow service providers to offer wider spectrum of fairness strategy and guarantee to users. Both algorithms are capable of taking advantage of the advanced features that HSDPA system offers, such as AMC and HARQ. The TBFQ alone would use more system resources to make sure that users near the edge of a cell meet its throughput guarantees. This however does not make good use of the system resources. The maxCI maximizes system resources by favoring users with higher channel quality but it has serious fairness issues. The TBFQ+maxCI offers a good compromise which can be seen by the results shown in Chapter 6.

7.2 Future research recommendations

There are several interesting research streams that can be branched out from the research conducted in this Ph.D. thesis.

As the analysis and studies conducted in this thesis are focused on fixed wireless systems, mobility of UEs would introduce an interesting dimension to TBFQ. **Issues of mobility** would require a management framework for TBFQ parameters which would include parameter passing mechanism. The fairness performance during cell switching should be affected as well, especially in location-dependent error conditions. The mapping of TBFQ parameters to other networks with different QoS mechanism is also worth studying.

The **CAC algorithm** that was developed for TBFQ could be improved further by anticipating user application characteristics and mobility. The study of CAC is a complex and difficult problem. Most researchers are interested in its technical performance, however, service providers are interested in revenue considerations. It would be interesting to incorporate TBFQ with cost parameters.

Compared to simulation analysis, theoretical analysis leads us to a better understanding of the model, and algorithmic computations consume much less time to obtain exact numerical results. The methodology employed allows us to have a great flexibility in modeling the fairness and addressing QoS issues to achieve performance optimization. This research has proposed a new approach for fair scheduling of the scarce wireless resources to wireless connections. It has opened a door to many important and challenging new research topics/projects, such as:

- **Implementation of data traffic model and numerical analysis:** The implementation of the data traffic model is similar to that for the voice model. However, since the transition matrix for the queue length process for the data model is a matrix of *M/G/*1 type, we cannot use the same method to do the computation. Since the size of the matrix is huge, which might be lager than 10,000, traditional methods usually fail. It is recommended to do a study to determine an effective numerical procedure for computing the stationary probability vector.
- **Mixed types of connections** After the model validation, it is recommended that further study and analysis be done, which includes implementation of the proposed methodology and analysis of the wireless network with mixed types of wireless connections.
- **Optimization** One of the advantages of the proposed algorithmic analysis is the flexibility in implementing different fairness scheduling algorithms, all based on the same fairness principle and comparing the performance of these algorithms, and eventually leading to recommendations for optimized system performance.

BIBLIOGRAPHY

[1] E. Gustafsson and A. Jonsson, "Always Best Connected," IEEE Wireless Commun., Feb. 2003, pp.49-55.

[2] ITU-RWP8F. Preliminary draft new recommendation (PDNR): Vision framework and overall objectives of the future development of IMS—2000 and of systems beyond IMS—2000, ITU-R M. [IMS-VIS] (Rev.1)-E 4 June 2002. Available at: http://www.fcc.gov/wrc03/files/docs/meeting/iwg/iwg_1/wrc03_iwg_1_background_doc_32.pdf

[3] Mohr W. WWRF—the wireless world research forum. Electronics and Communication Engineering Journal 2002; 14(6): 283–291.

[4] Jens Zanders, Seong-Lyun Kim, "Radio resource management for wireless networks," Mobile Communications Series, Artech house publishers, 2001.

[5] Nishith D. Tripathi, Jeffrey H. Reed, and Hugh F. VanLandingham, "Radio resource management in cellular systems," Kluwer Academic Publishers, 2001.

[6] Chen-Nee Chuah, Roy D. Yates, and D. J. Goodman, " integrated dynamic radio resource management," Proc. 45th IEEE VTC, pp. 584-88, 1995.

[7] A. Chandra, V. Gummalla, and J. O. Limb, Wireless Medium Access Control Protocols, IEEE Communications Surveys, Second Quarter 2000 pp. 2-15

[8] Y. Cao, and V. O. K. Li, "Scheduling Algorithms in Broad-Band Wireless Networks," Proceedings of the IEEE, vol. 89, no. 1, Jan 2001.

[9] R-G Cheng, C-S Wu, M-H Lin, M-C Liu, J-H Lin, H-R Lan, and C-C Liu, BMW OVERALL ARCHITECTURE AND PROTOTYPING, IEEE

[10] T. Enderes, S. C. Khoo, C. A. Somerville, and K. Samaras, Impact of Statistical Multiplexing on Voice Quality in Cellular Networks, in: Mobile Networks and Applications 7, 153-161, 2002.

[11] D. Raychaudhuri, D. Reininger, and S. Biswas, UPC based bandwidth Allocation for VBR Video in Wireless ATM Link, Proceeding of IEEE ICC'97, pp.1073-1097, Montreal, June 1997.

[12] O. Kubbar, and H. T. Mouftah, ALOHA-Based Channel Access Scheme for Wireless ATM Networks, Proceeding of the Canadian Conference on Broadband Research (CCBR), Ottawa, pp. 20-30, June 1998.

[13] O. Kubbar, and H. T. Mouftah, Multiple Access Control Protocols for Wireless ATM: Problems definitions and Design objectives, IEEE Communications Magazine, Nov. 1997, pp.93-98.

[14] O. Kubbar, and H. T. Mouftah, Broadband Wireless networks: An Investigation into the Traffic Behavior, Control, and QoS Guarantees, ICC 2000, vol. 2, pp. 985-989.

[15] D. Raychaudhuri, and N. D. Wilson, ATM-Based transport architecture for multiservices wireless personal communication networks, IEEE JSAC, vol. 12, pp. 1401-1414, Oct 1994

[16] D. J. Goodman, R. A. Valenzuela, K. T. Gayliard, and B. Ramamurthi, Packet reservation multiple access for local wireless communications, IEEE Trans. Comm, vol 37, pp. 885-890, Aug 1989.

[17] X. Qiu and V. O. Li, Dynamic Reservation Multiple Access (DRMA): A new multiple access scheme for Personal Communications System (PCS), ACM/Baltzer Wireless Networks Journal, vol. 2, pp. 117-128, June 1996.

[18] J.-F. Frigon, H.C.B. Chan, and V.C.M. Leung, Data and voice integration in DR-TDMA for wireless ATM networks, Proc. IEEE ICC, Jun. 1999.

[19] J. Mikkonen, et al., The MAGIC WAND Functional overview, IEEE JSAC, vol. 16, no.6, Aug. 1998, pp. 95372.

[20] E. Dahlman, P. Beming, J. Knutsson, F. Ovesjö, M. Persson, and C. Roobol, "WCDMA—The radio interface for future mobile multimedia communications," IEEE Trans. Veh. Technol., vol. 47, pp. 1105–1118, Nov. 1998.

[21] A. E. Brand, and A. H. Aghvami, Multidimensional PRMA with Prioritized Bayesian Broadcast – A Strategy for Multiservice Traffic over UMSS, IEEE Trans. on Vehic. Tech., vol 47, no. 4, Nov 1998.

[22] D. Raychaudhuri, Performance Analysis of Random Access Packet-Switched Code Division Multiple Access Systems, IEEE Trans. Comm. Vol. 29, no. 6, June 1981, pp. 895-901.

[23] I. Akyildiz, J. Mcnair, and L. Carrasco, Medium Access Control Protocols for Multimedia Traffic in Wireless Networks, IEEE Network Magazine, vol. 13, no. 4, Jul/Aug 1999, pp. 39 –47.

[24] N. Passas et al., Quality-of-Service-oriented Medium Access Control for Wireless ATM Networks, IEEE Comm. Mag., Nov 1997.

[25] D. J. Goodman, S. X. Wie, "Efficiency of packet- reservation multiple access," IEEE Trans. Vehic. Tech., vol. 40, Feb. 1991, pp. 170-76.

[26] M. J. Karol, Z. Liu, K. Y. Eng: „Distributed-Queueing Request Update Multiple Access (DQRUMA) for Wireless Packet (ATM) Networks", Proceedings ICC'95, 1995.

[27] D. Petras, A. Hettich and A. Krämling. Air Interface of a Wireless ATM System. Proceedings NOC'97, Antwerp, Belgium, Jun. 1997.

[28] N. Pronios, I. Dravopoulos, et al: „Wireless ATM MAC Overall Description", public CEC Deliverable No. 3D1 of the Magic WAND ACTS project, 1996.

[29] D. Raychaudhuri et al., “WATMnet: A prototype wireless ATM system for multimedia personal communication,” IEEE JSAC, vol. 15, Jan. 1997, pp. 83-95.

[30] F Bauchot et. al., “MASCARA, a MAC protocol for wireless ATM,” Proc. ACTs Mobile Summit ’96, Granada, Spain, Nov. 1996, pp. 647-51.

[31] GPP TS 25.321 “MAC Protocol Specification”, v3.6, December 2000.

[32] T. Kwon, Y. Choi, C. Bisdikian, and M. Naghshineh, "Measurement-based call admission control for adaptive multimedia in wireless/mobile networks," Proc. of IEEE Wireless Communications and Networking Conference, volume 2, pages 540–544, May 1999.

[33] D. Levine, I. Akyildiz, and M. Naghshineh, "A resource estimation and call admission algorithm for wireless multimedia networks using the shadow cluster concept," IEEE/ACM Trans. on Networking, 5(1):1–12, Feb. 1997.

[34] M. Naghshineh and S. Schwartz, "Distributed call admission control in mobile/wireless networks," IEEE Journal on Selected Areas in Communication, 14(4):711–717, May 1996.

[35] J. M. Peha and A. Sutivong, "Admission control algorithms for cellular systems. Wireless Networks," 7(2):117–125, Apr2001.

[36] R. Ramjee, D. Towsley, and R. Nagarajan, "On optimal call admission control in cellular networks," Wireless Networks, 3(1):29–41, May 1997.

[37] D. Hong, S.S. Rappaport, “Traffic Model and Performance Analysis for Cellular Mobile Radio Telephone Systems with Prioritized and Nonproritized Handoff Procedures”, IEEE Trans. Veh. Technology, Vol. VT-35, No. 3, August 1986, pp.77-92.

[38] J. M. Capone and I. Stavrakakis, "Delivering diverse delay/dropping qos requirements in a tdma environment", In Proc. of ACM MobiCom’97, volume 1, pages 110–119, Sept. 1997.

[39] N. R. Figueira and J. Pasquale, "Providing quality of service for wireless links: wireless/wired networks", IEEE Personal Communications, 6(5):42–51, Oct. 1998.

[40] C. Oliveira, J. B. Kim, and T. Suda. An adaptive bandwidth reservation scheme for high-speed multimedia wireless networks. IEEE Journal on Selected Areas in Communication, 16(6):858–874, Aug. 1998.

[41] A. K. Talukdar, B. R. Badrinath, and A. Acharya. Mrsvp: a resource reservation protocol for an integrated services network with mobile hosts. Wireless Networks, 7(1):5–19, Jan2001.

[42] W. Zhuang, B. Bensaou, and K. C. Chua. Adaptive quality of service handoff priority scheme for mobile multimedia networks. IEEE Trans. on Vehicular Technology, 49(2):494–505, Mar 2000.

[43] Evans, J.S. and Everitt, D., “Effective bandwidth-based admission control for multiservice CDMA cellular networks,” IEEE Transactions on Vehicular Technology 1999, vol.48, no.1, pp.36-46.

[44] Dongmei Zhao, Xuemin Shen, and Mark, J.W, "Efficient call admission control for heterogeneous services in wireless mobile ATM networks. IEEE Communications Magazine , Volume: 38 Issue: 10 , Oct. 2000, pp. 72 -78

[45] Anding Zhu, Jin Ding, and Jiandong Hu, "Adaptive Call Admission Control for Multi-Class CDMA Cellular Systems," Fifth Asia-Pacific Conference on Communications and Fourth Optoelectronics and Communications Conference. APCC/OECC'99, pp. 533-6 vol.1.

[46] Ishikawa, Y and Umeda, N., "Capacity Design and Performance of Call Admission Control in Cellular CDMA Systems," IEEE JSAC, vol. 15, no. 8, Oct 1997, pp. 1627-1635.

[47] Zhao Liu and El Zarki, M., "SIR-based call admission control for DS-CDMA cellular systems," IEEE JSAC, vol.12, no.4 May 1994, pp.638-44.

[48] Michael Andersin, Zvi Rosberg, and Jens Zander, "Soft and Safe Admission Control in Cellular Networks," IEEE Transaction on Networking, vol.5, no 2, April 1997.

[49] Knutsson, J., Butovitsch, P., Persson, M., Yates, R.D, "Downlink admission control strategies for CDMA systems in a Manhattan environment," IEEE 48th Vehicular Technology Conference (VTC'98), p.1453-7 vol.2.

[50] Kuri, J. and Mermelstein, P., "Call admission on the uplink of a CDMA system based on total received power," IEEE International Conference on Communications 1999, ICC '99, pp. 1431 -1436 vol.3.

[51] Baldo, O., Lee Kok Thong, and Aghvami, A.H., "Performance of distributed call admission control for multimedia high speed wireless/mobile ATM networks," IEEE International Conference on Communications, 1999. ICC '99, pp. 1982 -1986 vol.3.

[52] John Peha and Arak Sutivong, "Admission Control Algorithms for Cellular Systems," ACM/Baltzar Wireless Network, March 2001, Vol. 7, No. 2, pp. 117-125.

[53] Guo, Y. and Aazhong, B., "Call admission control in multi-class traffic CDMA cellular system using multiuser antenna array receiver," IEEE 51st Vehicular Technology Conference Proceedings, 2000, VTC2000-spring, pp. 365 -369, vol.1.

[54] Yang Xiao, Chen, C.L.P., and Wang, Y., "Quality of service and call admission control for adaptive multimedia services in wireless/mobile networks," Proceedings of the IEEE National Aerospace and Electronics Conference, 2000, NAECON 2000, pp. 214 -220.

[55] 3GPP Specification TS23.107, Dec. 1999.

[56] S. Nanda, D. J. Goodman, U. Timor, "Performance of PRMA: A Packet Voice Protocol for Cellular Systems", IEEE Trans. on Vehicular Technology, Vol. 40, No. 3, Aug. 1991, pp. 584-598.

[57] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated service networks – the single node case," in Proc. IEEE INFOCOM, Florence, Italy, May 1992, pp. 915–924.

[58] S. Keshav, An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network, Addison-Wesley, Reading, MA, 1997.

[59] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in Proc. ACM SIGCOMM, Austin, TX, Sep. 1989, pp. 1–12.

[60] S. J. Golestani, "A self-clocked fair queueing scheme for broadband application," in Proc. IEEE INFOCOM, Toronto, Canada, Jun. 1994, pp. 636–646.

[61] J. C. R. Bennett and H. Zhang, "WF2Q: Worst-case fair weighted fair queueing," in Proc. IEEE INFOCOM, San Francisco, CA, Mar. 1996, pp. 120–128.

[62] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round-robin," IEEE/ACM Trans. Networking, vol. 4, no. 3, pp. 375–385, Jun. 1996.

[63] S. S. Kanhere, H. Sethu, and A. B. Parekh, "Fair and efficient packet scheduling using elastic round robin," IEEE Trans. Parall. Distr. Syst., vol. 13, no. 3, pp. 324–336, Mar. 2002.

[64] J. Nagle, "On packet switches with infinite storage," IEEE Transactions on Communications, vol. 35, no. 4, April 1987.

[65] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," IEEE Transactions on Networking, vol. 3, no. 4, pp. 365–386, August 1995.

[66] S. Floyd, "Notes on class-based-queueing and guaranteed service," Unpublished Notes: http://www.aciri.org/floyd/cbq.html, July 1995.

[67] L. Georgiadis, R. Guerin, and A. Parekh, "Optimal multiplexing on a single link: Delay and buffer requirement," In Proceedings of INFOCOM'94, pp. 524-532, April 1994.

[68] L. Georgiadis, R. Guerin, V. Peris, and Sivarajan, "Efficient network QoS provisioning based on per node traffic shaping," IEEE/ACM Transactions on Networking, 4(4):482-501, August 1996.

[69] C. L. Liu and J. W. Layland, "Scheduling algorithm for multiprogramming in a hard real time environment," Journal of the ACM, 20(1):46-61, January 1973.

[70] D. Ferrari, and D. C. Verma, "A scheme for real-time channel establishment in wide area networks." IEEE Journal on Selected Areas in Communication, 8(3):368-379, April 1990.

[71] H. Adiseshu, G. Parulkar, and G. Varghese, "A reliable and scalable striping protocol," Proceedings of ACM SIGCOMM, Palo Alto, CA, August 1996, pp. 131–141.

[72] D. Verma, H. Zhang, and D. Ferrari, "Delay jitter control for real-time communication in a packet switching network," Proceedings of TriComm, 1991.

[73] P. Bhagwat, A. Krishna, and S. Tripathi, "Enhancing throughput over wireless LAN's using channel state dependent packet scheduling," Proc. INFOCOM96, Mar. '96, pp. 1133-40.

[74] S. Lu, and V. Bharghavan, "Fair scheduling in wireless packet networks," IEEE/ACM Trans. Networking, vol. 7, no.4, pp. 473-489, 1999.

[75] T. S. E. Ng, I. Stoica, and H. Zhang, "Packet fair queueing algorithms for wireless networks with location-dependent errors," in Proc. INFOCOM98, Mar. 1998, pp. 1103-1111.

[76] P. Goyal, H. M. Vin and H. Cheng, "Start-Time Fair Queuing: A Scheduling Algorithm for Integrated Services Packet Switching Networks," IEEE Trans. Networking, vol. 5, no. 5, pp. 690-704, Oct. 1997.

[77] W. K. Wong, V.C.M. Leung, Scheduling for integrated services in next generation broadcast networks, Proc. IEEE WCNC, New Orlean, LA, Sep. 1999.

[78] W. K. Wong, H. Zhu, V. C. M. Leung, "Soft-QoS Provisioning using the Token Bank Fair queueing scheduling Algorithm," IEEE Wireless Communications, vol. 10, no. 3, Jun 2003 pp. 8-16.

[79] D. P. Bertsekas and R. Gallager, Data Networks, Prentice Hall, Upper Saddle River, NJ, 2nd edition, 1991.

[80] Y. Zhou and H. Sethu, "On the relationship between absolute and relative fairness bounds," IEEE Communication Letters, vol. 6, no. 1, pp. 37–39, January 2002.

[81] D. Stiliadis and A. Verma, "Latency-rate servers: A general model for analysis of traffic scheduling algorithms," IEEE Transactions on Networking, vol. 6, no. 3, pp. 611–624, October 1996.

[82] D. Stiliadis. "Traffic schedulinrr in packet switched networks: Analysis, design, and implementation," Ph.D. thesis, Computer. Engineering. Department, University of California, Santa Cmz, June 1996.

[83] D.R. Cox, "A use of complex probabilities in the theory of stochastic processes," Cambridge Philosophical Society, vol. 51, pp. 313–319, 1955.

[84] K. W. Ross, "Multiservice Loss Models for Broadband Telecommunication Networks", London: Spring-Verlag, 1995.

[85] W. K. Wong, Y. Qian, V. C. M. Leung, Scheduling for heterogeneous traffic in next generation wireless networks, IEEE Globecom Conference, v 1, 2000, pp.283-287.

[86] J. C. Chen, K. M. Sivalingam, P. Agrawal, and R. Acharya, On scheduling multimedia services in low-power MAC for wireless and mobile ATM networks, IEEE International Symposium on Personal, indoor and Mobile Radio Comm., v 1, 1998, pp 243-247.

[87] H. Zhang, Service disciplines for guaranteed performance service in packet-switching networks, Proc. IEEE, v 83, n 10, Oct. 1995, pp 1374-1396.

[88] O. Kubbar, and H. T. Mouftah, Broadband wireless networks: An investigation into the traffic behavior, control, and QoS guarantees, ICC 2000, v 2, pp. 985-989.

[89] R. Guerin, H. Ahmadi, and M. Nagashineh, Equivalent Capacity and its application to Bandwidth Allocation in High Speed Networks, IEEE J. Select. Areas Comm., v 9, n 7, Sept. 1991, pp. 968-981.

[90] M. Seth, A. O. Fapojuwo, Adaptive resource management for multimedia wireless networks, VTC2003F, Orlando, FL, Oct. 2003.

[91] W. K. Wong, H. Tang, S. Guo, and V, C. M. Leung, Scheduling algorithm in a point-to-multipoint broadband wireless access network, VTC2003 Fall, Orlando, FL, Oct. 2003.

[92] 3GPP TR 25.848 V4.0.0 Technical Report: 3rd Generation Partnership Project; Technical Specification: Group Radio Access Network; Physical layer aspects of UTRA High Speed Downlink Packet Access, Mar. 2001.

[93] M. Schwartz, "Information Transmission, Modulation, and Noise, 4th Ed. New York: McGraw-Hill, 1990.

[94] D. L. Isaacson, R. W. Madsen, Markov Chains: Theory and Applications, New York: Wiley & Sons, 1976.

[95] 3GPP TS 25.308 V5.4.0 (2003-03) High Speed Downlink Packet Access (HSPDA) Overall Description; Stage 2 (Release 5)

[96] 3GPP TR 25.899 V0.2.2 (2003-11) HSDPA Enhancements (Release 6)

[97] 3GPP TS 25.321 V5.6.0 (2003-09), Medium Access Control (MAC) Protocol Specification (Release 5)

[98] 3GPP TR 25.950 V4.0.0 (2001-03) UTRA High Speed Downlink Packet Access (Release 4)

[99] P. Ameigeiras, "Packet Scheduling And Quality of Service in HSDPA," Ph. D. Thesis, Aalborg University, Denmark October 2003.

[100] Jalali A. et al. Data Throughput of CDMA-HDR a High Efficiency-High Data Rate Personal ommunication Wireless System. Vehicular Technology Conference, 2000. VTC 2000 Spring. Volume 3. pp. 1854-1858.

[101] Wang I.-J. et al. Weighted Averaging and Stochastic Approximation. Decision and Control, 1996.,Proceedings of the 35th IEEE, Volume: 1. 11-13 Dic. Pp 1071-1076.

[102] UMTS 30.03 version 3.2.0. TR 101 112 v3.2.0 (1998-04). Selection procedures for the choice of radio transmission technologies of the UMTS.

[103] T. Kolding, F. Frederiksen, and P. Mogensen, "Performance Evaluation of Modulation and Coding Schemes Proposed for HSDPA in 3.5G UMTS Networks," Proc. Of the Symposium on Wireless Personal Multimedia Communications (WPMC01), September 2001, Aalborg, Denmark, Vol. 1, pp. 307-312.

[104] G.J.J.A.N. van Houtum, "New approaches for multi-dimensional queueing systems," Ph.D. thesis, Eindhoven University of Technology, 1995.

[105] G.J.J.A.N. van Houtum, W.H.M. Zijm, I.J.B.F. Adan and J. Wessels, "Bounds for performance characteristics: a systematic approach via cost structures," Stochastic Models, vol. 14, 205-224, 1998.

[106] D. R. Cox, and H. D. Miller, The Theory of Stochastic Processes, 2nd edition, John Wiley and Sons, 1965.

[107] W.K. Grassmann, M.I. Taksar and D.P. Heyman, "Regenerative analysis and steady state distributions forMarkov chains," Operations Research, vol 33, 1107–1116, 1985.

[108] Y.Q. Zhao, "Censoring Technique in Studying Block-Structured Markov Chains," in Advances in Algorithmic Methods for Stochastic Models, Latouche and Taylor Eds, Notable Publications Inc., 417-433, 2000.

[109] M.F. Neuts, Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach, The Johns Hopkins University Press, Baltimore, 1981.

[110] http://www.opnet.com/products/networkr&d/network_performance_eval.html

APPENDIX A : IMPLEMENTATION AND ANALYSIS USING ON-OFF VOICE TRAFFIC

A.1 Uniform On-Off Voice connections

We consider the FaSA with on-off voice connections and implement this model using the recommended recursive approach. For the purpose of implementation, we assume that the packets generated from all connections have the same size $L = L_x^i$ bytes (e.g. L=53 bytes). Since we are considering frame-based wireless systems, the parameters of the frame will set the same as those in [85] where a frame has 14 slots. Voice sources follow an alternating pattern of talk spurts and silence periods (on and off). Within each of the talk spurt intervals, it is standard to assume that the speech codec rate is either 32 kbps or 64 kbps. The alternating on-off pattern can be mathematically modeled as an on-off process or a two-state discrete time Markov chain, or both on and off periods are geometrically distributed. In practice, each on interval typically averaging 0.4-1.2 sec is followed by a silence interval averaging 0.6-1.8 sec. It assumes that the number of on-slots and the number of off-slots are both geometrically distributed random variables, with the average $1/(1 - p_1)$ and $1/(1 - p_0)$ respectively, and these two variables are independent. Equivalently,

$$P(\text{number of on-slots} = n) = (1 - p_1)p_1^{n-1} \quad n = 1, 2, \ldots$$

and

$$P(\text{number of off-slots} = n) = (1 - p_0)p_0^{n-1} \quad n = 1, 2, \ldots$$

Or the transition probability matrix $\mathbb{P}_{\text{on-off}}$ of the two-state Markov chain is given by

$$\mathbb{P}_{\text{on-off}} = \begin{array}{c} \\ 0 \\ 1 \end{array}\!\!\begin{array}{c} \begin{array}{cc} 0 & \quad 1 \end{array} \\ \begin{pmatrix} p_0 & 1 - p_0 \\ 1 - p_1 & p_1 \end{pmatrix} \end{array} \tag{A.1}$$

where state 0 and 1 represents OFF and ON, respectively. For convenience, denote $q_i = 1 - p_i$ for $i = 0, 1$. During an on-period, the packet generate rate is assumed to be a constant β measured in bytes, and during an off-period, no packet will be generated. Let μ_n, $n = 0, 1, \ldots, 14$ be the probability that n bandwidth slots (one slot for transmitting on packet) would be assigned to the connection in a frame. Denote $\mu_{n \geq k} = \sum_{n \geq k} \mu_n$ for k = 0, 1, · · ·. For convenience, let $\mu_n = 0$, if $n > 14$ or $n < 0$.

Since the packet generate rates during an on-period and an off-period are different, the queue length process described in (4.1) is not a Markov chain. For converting this non-Markov chain into an

Markov chain, we introduce a supplementary variable $S_x^i(k)$ with states 0 and 1, representing the status of the voice resource: OFF and ON at time epoch k, respectively. Assume that the system is observed at epochs k, the bandwidth assignment in frame $[k, k + 1)$ is used to send packets observed at k, the bytes generated in frame $[k, k + 1)$ are put in the data buffer, and a possible change in status, either from 0 to 1 or from 1 to 0, occurs immediately before the observation epoch. Then, $\{Q_x^i(k), S_x^i(k)\}$ is a discrete time Markov chain with its state space:

$$\{(0,0),(0,1),(1,0),(1,1),(2,0),(2,1),\ldots,\ldots\}.$$

For convenience, we partition the transition probability matrix $\mathbb{P}_V = \left(p_{(i,r),(j,s)}\right)_{i,j=0,1,\ldots;r,s=0,1}$ of $\{Q_x^i(k), S_x^i(k)\}$ into blocks according to the queue length as follows:

$$\mathbb{P}_V = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ \vdots \\ \vdots \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & \cdots & \cdots & \end{array} \\ \begin{pmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \mathbf{P}_{0,2} & \cdots & \cdots \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & \cdots & \cdots \\ \mathbf{P}_{2,0} & \mathbf{P}_{2,1} & \mathbf{P}_{2,2} & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \end{array} \tag{A.2}$$

where

$$\mathbf{P}_{i,j} = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} p_{(i,0),(j,0)} & p_{(i,0),(j,1)} \\ p_{(i,1),(j,0)} & p_{(i,1),(j,1)} \end{pmatrix} \end{array} \tag{A.3}$$

Since the maximal number of bandwidth slots, which can be possibly assigned to a connection during a frame, is 14 packets per frame or $14L$ bytes, and the packet generate rate is β bytes per frame with $\beta < L$, the transition matrix $\mathbb{P}_V$ possesses a repeating structure:

$$
\mathbb{P}_V = \begin{array}{c|cccccccccccccccccc|}
 & 0 & 1 & 2 & \cdots & \cdots & \beta\text{-}1 & \beta & \cdots & \cdots & 53 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & * & & & & & * & & & & & & & & & & & & \\
1 & & * & & & & & * & & & & & & & & & & & \\
2 & & & * & & & & & * & & & & & & & & & & \\
\vdots & & & & \ddots & & & & & \ddots & & & & & & & & & \\
\vdots & & & & & \ddots & & & & & \ddots & & & & & & & & \\
\beta-1 & & & & & & * & & & & & * & & & & & & & \\
\beta & & & & & & & * & & & & & * & & & & & & \\
\beta+1 & & & & & & & & * & & & & & * & & & & & \\
\vdots & & & & & & & & & \ddots & & & & & \ddots & & & & \\
53 & * & & & & & & * & & & * & & & & & * & & & \\
54 & & * & & & & & & * & & & * & & & & & * & & \\
55 & & & * & & & & & & * & & & * & & & & & * & \\
\vdots & & & & \ddots & & & & & & \ddots & & & \ddots & & & & & \ddots
\end{array}
$$

From this structure, we know that only the following transition blocks $\mathbf{P}_{i,j}$ are nonzero.

1. For $i = 0, 1, \ldots, 52$, $\mathbf{P}_{i,i} \neq 0$ and $\mathbf{P}_{i,i+\beta} \neq 0$.

2. For $i = 53k + i$ with $1 \leq k \leq 14$ and $0 \leq i < 53$, $\mathbf{P}_{53k+i,53j+i} \neq 0$ for $j = 0, 1, \ldots, k$ and $\mathbf{P}_{53k+i,53j+i+\beta} \neq 0$ for $j = 0, 1, \ldots, k$.

Partition the transition matrix into super-blocks (blocks with larger size):

$$\mathbb{P}_V = \left(\mathbb{P}_{i,j}\right) \tag{A.4}$$

where

$$\mathbb{P}_{i,j} = (\mathbf{U}_{m,n})_{53\times 53} \tag{A.5}$$

is a matrix consisting of 53×53 blocks $\mathbf{U}_{m,n}$ and each $\mathbf{U}_{m,n}$ is a matrix of 2×2. Then, $\mathbb{P}_V$ is given by

$$\mathbb{P}_V = \begin{array}{c|cccccccccc} & 0 & 1 & 2 & \cdots & 13 & 14 & 15 & 16 & \cdots & \cdots \\ \hline 0 & \mathbf{B}_1 & \mathbf{B}_0 & & & & & & & & \\ 1 & \mathbf{B}_2 & \mathbf{C}_1 & \mathbf{C}_0 & & & & & & & \\ \vdots & \vdots & \vdots & \ddots & \ddots & & & & & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & & & & & \\ 13 & \mathbf{B}_1 & \mathbf{C}_{13} & \mathbf{C}_{12} & \cdots & \mathbf{C}_1 & \mathbf{C}_0 & & & & \\ 14 & \mathbf{C}_{15} & \mathbf{C}_{14} & \mathbf{C}_{13} & \cdots & \mathbf{C}_2 & \mathbf{C}_1 & \mathbf{C}_0 & & & \\ 15 & & \mathbf{C}_{15} & \mathbf{C}_{14} & \mathbf{C}_{13} & \cdots & \mathbf{C}_2 & \mathbf{C}_1 & \mathbf{C}_0 & & \\ \vdots & & & \ddots & \ddots & \ddots & \cdots & \ddots & \ddots & \ddots & \\ \vdots & & & & \ddots & \ddots & \ddots & \cdots & \ddots & \ddots & \ddots \end{array}, \tag{A.6}$$

where $\mathbf{B}_i$ and $\mathbf{C}_i$ are matrices consisting of 53 row blocks and column blocks with block size 2 × 2.

For the expression of $\mathbf{B}_0$, notice that $\mathbf{B}_0$ possesses the following structure:

$$\mathbf{B}_0 = (\mathbf{U}_{m,n}) = \begin{pmatrix} & & & \\ & & & \\ *_{-1} & & & \\ & *_{-1} & & \\ & & \ddots & \\ & & & *_{-1} \end{pmatrix}. \tag{A.7}$$

Specifically, all $\mathbf{U}_{m,n} = \mathbf{0}$ except for the following cases. For $53-\beta \leq m \leq 52$,

$$\mathbf{U}_{m,m+\beta-53} = \begin{pmatrix} 0 & 0 \\ q_1 & p_1 \end{pmatrix}. \tag{A.8}$$

For the expression of $\mathbf{C}_0$, notice that $\mathbf{C}_0$ possesses the same structure as that for $\mathbf{B}_0$:

$$\mathbf{C}_0 = (\mathbf{U}_{m,n}) = \begin{pmatrix} & & & \\ & & & \\ *_{-1} & & & \\ & *_{-1} & & \\ & & \ddots & \\ & & & *_{-1} \end{pmatrix}. \tag{A.9}$$

Specifically, all $\mathbf{U}_{m,n} = \mathbf{0}$ except for the following cases. For $53-\beta \leq m \leq 52$,

$$\mathbf{U}_{m,m+\beta-53} = \begin{pmatrix} 0 & 0 \\ q_1\mu_0 & p_1\mu_0 \end{pmatrix}. \tag{A.10}$$

For the expression of $\mathbf{C}_i$ with i = 1, 2, . . . , 14: Notice that all $\mathbf{C}_1$, $\mathbf{C}_2$, . . . ,$\mathbf{C}_{14}$ possess the same structure:

$$\mathbf{C}_i = (\mathbf{U}_{m,n}) = \begin{pmatrix} * & & & & *_1 & & & \\ & * & & & & *_1 & & \\ & & \ddots & & & & \ddots & \\ & & & \ddots & & & & *_1 \\ *_{-1} & & & & * & & & \\ & *_{-1} & & & & * & & \\ & & \ddots & & & & \ddots & \\ & & & *_{-1} & & & & * \end{pmatrix}. \tag{A.11}$$

Specifically, all $\mathbf{U}_{m,n} = \mathbf{0}$ except for the following cases.

1. For $0 \le m \le 52$,

$$\mathbf{U}_{m,m} = \begin{pmatrix} p_0\mu_{i-1} & q_0\mu_{i-1} \\ 0 & 0 \end{pmatrix}; \tag{A.12}$$

2. For $0 \le m \le 52 - \beta$,

$$\mathbf{U}_{m,m+\beta} = \begin{pmatrix} 0 & 0 \\ q_1\mu_{i-1} & p_1\mu_{i-1} \end{pmatrix}; \tag{A.13}$$

3. For $53 - \beta \le m \le 52$,

$$\mathbf{U}_{m,m+\beta-53} = \begin{pmatrix} 0 & 0 \\ q_1\mu_i & p_1\mu_i \end{pmatrix}; \tag{A.14}$$

For the expression of $\mathbf{B}_i$ with i = 1, 2, . . . , 15 with $B_{15} = C_{15}$: Notice that all B_1, B_2, . . . , B_{15} possess the same structure:

$$\mathbf{B}_i = (\mathbf{U}_{m,n}) = \begin{pmatrix} * & & & & *_1 & & & \\ & * & & & & *_1 & & \\ & & \ddots & & & & \ddots & \\ & & & \ddots & & & & *_1 \\ & & & & * & & & \\ & & & & & * & & \\ & & & & & & \ddots & \\ & & & & & & & * \end{pmatrix}. \tag{A.15}$$

Specifically, all $U_{m,n} = \mathbf{0}$ except for the following cases.

1. For $0 \leq m \leq 52$,

$$\mathbf{U}_{m,m} = \begin{pmatrix} p_0 \mu_{\geq(i-1)} & q_0 \mu_{\geq(i-1)} \\ 0 & 0 \end{pmatrix}; \tag{A.16}$$

2. For $0 \leq m \leq 52 - \beta$,

$$\mathbf{U}_{m,m+\beta} = \begin{pmatrix} 0 & 0 \\ q_1 \mu_{\geq(i-i)} & p_1 \mu_{\geq(i-i)} \end{pmatrix}. \tag{A.17}$$

A.2 Implementation

To implement this model based on the steps mentioned earlier in Chapter 4, we need to find various mathematical expressions. Some of these expressions cannot be given without explicit knowledge of dependence between connections. In these cases, independent assumptions have to be imposed to derive approximate formulas.

Step 0: Use the initial approximate distribution for μ_k. Specifically,

$$\tilde{\mu}_x^i(k) = \begin{cases} 1, & \text{with probability } (r_x^i / 53) \\ 0, & \text{with probability } 1 - (r_x^i / 53) \end{cases} \tag{A.18}$$

if the total number of the wireless connections is not larger than 14; otherwise we let

$$\tilde{\mu}_x^i(k) = \begin{cases} 1, & \text{with probability } (r_x^i / 53)(14 / n_x) \\ 0, & \text{with probability } 1 - (r_x^i / 53)(14 / n_x) \end{cases} \tag{A.19}$$

This approximates that case, in which only 14 out of n_x connections would be selected for a bandwidth slot. For the queue length process in Step 1 to be stable, we require that the average packet generate rate for the on-off process is smaller than the average service rate; or

$$\frac{1-p_0}{2-p_0-p_1} \bullet 32 \text{ bytes/frame} < \frac{r_x^i}{53} \bullet \frac{14}{\max(n_x, 14)} \text{ bytes/frame} \tag{A.20}$$

Step 1: For the first recursion, use $\tilde{\mu}_k$ given in Step 0, otherwise use $\tilde{\mu}_k$ updated in Step 4. For all recursions, the packet generate rate is zero during an OFF period and during an ON frame,

$$\lambda_i = P(i \text{ bytes generated by a connection in a frame} | \text{ON}) = \begin{cases} 1 & \text{if } i = \beta \\ 0 & \text{if } i \neq \beta \end{cases}$$

- Compute $\mathbf{C}_0$ according to (A.9) and (A.10)
- Compute $\mathbf{C}_i$ for $i = 1, 2, \ldots, 14$ according to (A.11), (A.12), (A.13), and (A.14)
- Compute $\mathbf{B}_0$ according to (A.7) and (A.8)
- Compute $\mathbf{B}_i$ for $i = 1, 2, \ldots, 15$ according to (A.15), (A.16), and (A.17)
- Set $\mathbf{C}_{15} = \mathbf{B}_{15}$

Let $\boldsymbol{\pi} = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots)$ be the stationary probability vector of $\mathbb{P}_V$ partitioned according to the super-blocks. Since $\mathbb{P}_V$ is a *GI/M/*1 type of matrix, according to Neuts [109] $\boldsymbol{\pi}$ has the following matrix-geometric solution:

$$\pi_{1+k} = \pi_1 \mathbf{R}^k, \qquad \text{for } k = 0, 1, 2, \ldots, \tag{A.21}$$

where $\mathbf{R}$ is the minimal non-negative solution of the matrix equation

$$\mathbf{R} = \mathbf{C}_0 + \mathbf{C}_1\mathbf{R} + \mathbf{C}_2\mathbf{R}^2 + \cdots + \mathbf{C}_{15}\mathbf{R}^{15}.$$

Therefore, we compute the stationary probability vector $\boldsymbol{\pi}$ in two steps. In the first step, we compute $\mathbf{R}$ and in the second step, the boundary probability vectors $\boldsymbol{\pi}_0$ and $\boldsymbol{\pi}_1$.

For computing the matrix $\mathbf{R}$, let

$$\mathbf{C}_1^{(0)} = \mathbf{C}_0 + \mathbf{C}_1, \qquad \mathbf{C}_i^{(0)} = \mathbf{C}_i, \qquad i = 2, 3, \ldots, 15 \tag{A.22}$$

and

$$\mathbf{C}_{15}^{(k)} = \mathbf{C}_{15}, \qquad \text{for } k \geq 0 \tag{A.23}$$

Compute for k=1, 2, …,

$$\mathbf{C}_i^{(k)} = \mathbf{C}_i + \mathbf{C}_0 \widehat{\mathbf{C}}_1^{(k-1)} \widehat{\mathbf{C}}_{i+1}^{(k-1)}, \qquad i = 1, 2, 3, \ldots, 14 \tag{A.24}$$

where $\widehat{\mathbf{C}}_1^{(k-1)} = \sum_{n=0}^{\infty} (\mathbf{C}_1^{(k-1)})^n$ with $(\mathbf{C}_1^{(k-1)})^0 = \mathbf{I}$. Then,

$$\mathbf{R} = \lim_{k\to\infty} \mathbf{C}_0 \widehat{\mathbf{C}}_1^{(k)} \tag{A.25}$$

In summary, for computing **R** we need to do the following:

- Set initial values $\widehat{\mathbf{C}}_1^{(k-1)}$ for i = 0, 1, . . . , 15 according to (A.18), referred to as $\mathbf{C}_{old}$.
- For $k = 1$,

  Compute the old $\widehat{\mathbf{C}}_1^{(k-1)} = \sum_{n=0}^{\infty} (\mathbf{C}_1^{(k-1)})^n$;;

  Compute $\mathbf{C}_i^{(k)}$ for i = 1, 2, . . . ,14 according to (A.22), and set $\mathbf{C}_{15}^{(k)} = \mathbf{C}_{15}^{(k-1)}$, referred to as $\mathbf{C}_{new}$;
- Let $k = k + 1$ and repeat the previous step to compute $\mathbf{C}_{new}$ until convergence. After convergence, denote $\mathbf{C}_{15}^{(\infty)} = \mathbf{C}_1^{(k)}$.
- Compute $\mathbf{R} = \mathbf{C}_0 \widehat{\mathbf{C}}_1^{(\infty)}$, where $\widehat{\mathbf{C}}_1^{(k)} = \sum_{n=0}^{\infty} (\mathbf{C}_1^{(k)})^n$.

For computing the boundary probability vectors $\boldsymbol{\pi}_0$ and $\boldsymbol{\pi}_1$, we first compute the following so-called censored matrix

$$\mathbb{P}_V^{(1)} = \begin{pmatrix} \mathbf{B}_1 & \mathbf{B}_0 \\ \mathbf{B}_2 + \sum_{n=1}^{\infty} \mathbf{R}^n \mathbf{B}_{n+2} & \mathbf{C}_1 + \sum_{n=1}^{\infty} \mathbf{R}^n \mathbf{C}_{n+1} \end{pmatrix} \tag{A.26}$$

Then, apply the GTH algorithm to the matrix $\mathbb{P}_V^{(1)}$ for obtaining the boundary vectors $\boldsymbol{\pi}_0$ and $\boldsymbol{\pi}_1$, which differ from the boundary probability vectors by a constant c. Use

$$\pi'_{1+k} = \pi'_k \mathbf{R}, \qquad \text{for } k = 1, 2, \ldots, \tag{A.27}$$

to compute all other non-boundary vectors, which differ from the non-boundary probability vectors by the same constant c. Finally, the constant c is computed according to

$$c = (\pi'_0 + \pi'_1 + \pi'_2 + \cdots)e, \tag{A.28}$$

where **e** is a column vector of ones and

$$\pi_i = \pi'_i / c, \qquad \text{for } i = 0, 1, 2, \ldots \tag{A.29}$$

As a summary, for computing the boundary probability vectors and therefore the whole distribution, we need to do the following:

- Compute the censored matrix $\mathbb{P}_V^{(1)}$ according to (A.26);
- Note that the states in $\mathbb{P}_V^{(1)}$ are presented in (i, j, k), where $i = 0,1$ is the index for the super-block, $j = 0,1,2,\ldots,52$ represents the number of bytes below that for a packet, and $k = 0,1$ is the status of the wireless connection. This presentation is not convenient for directly calling the GTH algorithm function. We re-label the states (i, j, k) into $0,1,2,\ldots,211$ and denote the new transition matrix by $\mathbb{Q}_V = (q_{r,s})$ as follows:

```
for $i, i' = 0, 1$;
    for $j, j' = 0, 1, \ldots, 52$;
        for $k, k' = 0, 1$;
            $q(106i + 2j + k, 106i + 2j' + k') = p((i, j, k), (i', j', k'))$.
```

- Apply the GTH algorithm to $\mathbb{Q}_V$ to compute that the boundary steady-state distribution vector, which is denoted by $q' = (q_0', q_1', q_2', \cdots, q_{211}')$.
- Convert the state label back by

```
for $i = 0, 1$;
    for $j = 0, 1, \ldots, 52$;
        for $k = 0, 1$;
            $\pi_{i,j,k}' = q_{106i+2j+k}'$.
```

Now, the boundary invariant vectors are given by

$$\pi_i' = (\pi_{i,0,0}', \pi_{i,0,1}', \pi_{i,1,0}', \pi_{i,1,1}', \ldots, \pi_{i,52,0}', \pi_{i,52,1}'), \qquad i = 0,1 \qquad \text{(A.30)}$$

- Compute π_{1+k}' for $k = 1, 2, \ldots$ according to $\pi_{1+k}' = \pi_k' \mathbf{R}$.
- Compute the normalization constant c according to (A.28).
- Finally, compute $\pi_i = \pi_i' / c$ for $i = 0, 1, \ldots$.
- The mean buffer content in bytes is computed by

$$\bar{Q} = \sum_{i=0}^{\infty}\sum_{j=0}^{52}(53i + j)(\pi_{i,j,0} + \pi_{i,j,1}) \tag{A.31}$$

When using $\tilde{\mu}_k$, the computed steady-state probability is denoted by $\tilde{\pi}_{i,j,k}$ if the states are labeled according to the super-block level i, remaining bytes j and connection status k, or by $\tilde{\pi}_{106i+2j+k}$ if the states are labeled according to 0, 1, 2, ...

Step 2: The extra resource process is evaluated by subtracting the total of basic bandwidth scheduled to all terminals from the total bandwidth. We divide the evaluation in this step into three sub-steps. Recall that $r = r_x^i$ and Δ_x is a random variable representing the number of bandwidth slots guaranteed to a connection of type x with its distribution given by:

$$\delta_1 = P\{\Delta_x = 1\} = \begin{cases} r_x^i / 53 & \text{if } n_x \le 14 \\ (r_x^i / 53)(14 / n_x) & \text{if } n_x > 14 \end{cases} \tag{A.32}$$

and

$$\delta_0 = P\{\Delta_x = 0\} = \begin{cases} 1 - r_x^i / 53 & \text{if } n_x \le 14 \\ 1 - (r_x^i / 53)(14 / n_x) & \text{if } n_x > 14 \end{cases} \tag{A.33}$$

which has the same distribution as $\tilde{\mu}_x^i(k)$ given in Step 0.

First define

$$H_x = \mathbf{1}_{\{\tilde{Q}_x^i \ge 53\}} \bullet \mathbf{1}_{\{\Delta_x = 1\}} \tag{A.34}$$

which is a discrete random variable with possible values 0 and 1. Therefore,

$$h_x(n) = P\{H_x = n\} = \begin{cases} 1 - \frac{r_x^i}{53}\sum_{i=1}^{\infty}\sum_{j=0}^{52}(\pi_{i,j,0} + \pi_{i,j,1}), & n = 0 \\ \frac{r_x^i}{53}\sum_{i=1}^{\infty}\sum_{j=0}^{52}(\pi_{i,j,0} + \pi_{i,j,1}), & n = 1 \end{cases} \tag{A.35}$$

where $\pi_{i,j,k}$ are the queue length probabilities in Step 1.

Second define

$$K = \min\left(14, \sum_{x,i}\mathbf{1}_{\{Q_x^i \ge 53\}} \bullet \mathbf{1}_{\{\Delta_x = 1\}}\right) = \min\left(14, \sum_{i}\mathbf{1}_{\{Q_x^i \ge 53\}} \bullet \mathbf{1}_{\{\Delta_x = 1\}}\right) \tag{A.36}$$

which is also a discrete random variable with possible values 0, 1, 2, $\cdots$, min(n_x, 14). Its distribution can be evaluated according to

$$\kappa(n) = P\{K = n\} = \begin{cases} \binom{n_x}{n} h_x(1)^n h_x(0)^{n_x - n}, & n = 0,1,2,\ldots,13 \\ \sum_{k=14}^{n_x} \binom{n_x}{k} h_x(1)^k h_x(0)^{n_x - k}, & n = 14 \end{cases} \quad \text{(A.37)}$$

Finally, the probability distribution of the extra resource b_n, measured in packets, is evaluated by

$$b_n = \kappa(14 - n), \qquad n = 0,1,2,\ldots,14 \quad \text{(A.38)}$$

When using an approximate $\tilde{\mu}_k$, the computed distribution of the extra resource is denoted by $\tilde{b}_n$.

Step 3: Note that both the initial fairness function and the updated fairness function share the same expression. To provide a static formula, we assume that $X_1, X_2, \ldots, X_{n_x}$ are i.i.d. random variables. When $X_i = Q_x^i$ or $X_i = E_x^i$, the approximate expression for the initial fairness function or for the updated fairness function is given, respectively. The fairness function allocates the extra bandwidth to the connection(s) with the largest token balance.

Random allocation: All extra bandwidth is assigned to a randomly selected terminal from the terminals with the largest token balance. In this case, the initial fairness function is defined as

$$\tilde{\theta}_x^i(k) = \begin{cases} \tilde{B}(k) \textit{ with probability } 1/m_k, & \textit{if } \tilde{Q}_x^i(k-1) = \max_{j,y} \tilde{Q}_y^j(k-1) \\ 0 & , \textit{if } \tilde{Q}_x^i(k-1) < \max_{j,y} \tilde{Q}_y^j(k-1) \end{cases} \quad \text{(A.39)}$$

and the update fairness function is defined as

$$\widetilde{\theta}_x^i(k) = \begin{cases} \widetilde{B}(k) \textit{ with probability } 1/m_k, & \textit{if } \widetilde{E}_x^i(k-1) = \max_{j,y} \widetilde{E}_y^j(k-1) \\ 0 & , \textit{if } \widetilde{E}_x^i(k-1) < \max_{j,y} \widetilde{E}_y^j(k-1) \end{cases} \quad \text{(A.40)}$$

where m_k is the number of connections with the largest token balance at time epoch k.

Recall that d_x^i is the debt limit and U_b is the token balance upper limit. In steady-state, denote

$$p_l = P\left\{X_x^i = l\right\}, \qquad l = d_x^i, d_x^i + 1, \cdots, -1, 0, 1, 2, \cdots, U_b \quad \text{(A.41)}$$

and

$$q_l = P\left\{X_x^i < l\right\} = \sum_{l=d_x^i}^{l-1} p_l, \qquad l = d_x^i + 1, d_x^i + 2, \cdots, -1, 0, 1, 2, \cdots, U_b \quad \text{(A.42)}$$

For convenience, set $q_{d_x^i} = 0$. Then, for any n = 1, 2, $\cdots$, n_x and i = 1, 2, $\cdots$, n_x,

$$c_n^i = P\{m = n, X_x^j = \max_j X_x^j\} = \binom{n_x - 1}{n-1} \sum_{l=d_x^i}^{U_b} p_l^n q_l^{n_x - n} \tag{A.43}$$

Then, for n = 1, 2, . . . , 14,

$$\begin{aligned} \tilde{f}_n &= P\left\{\tilde{\theta}_x^i = n\right\} \\ &= P\left\{\tilde{B} = n, X_x^j = \max_j X_x^j, \text{ extra bandwidth is lend to terminal } i\right\} \\ &= \sum_{l=1}^{n_x} P\left\{m = l, \tilde{B} = n, X_x^j = \max_j X_x^j, \text{ extra bandwidth is lend to terminal } i\right\} \\ &= \sum_{l=1}^{n_x} \frac{c_l^i b_n}{l} \end{aligned}$$

where n_x is the number of the terminals and m is the steady-state number of connections with the largest token balance. Finally,

$$\tilde{f}_0 = P\left\{\tilde{\theta}_x^i = 0\right\} = 1 - \sum_{n=1}^{14} \tilde{f}_n \tag{A.44}$$

Step 4: In this step, we update the bandwidth allocation process according to

$$\tilde{\mu}_x^i(k) = \mathbf{1}_{\{\Delta_x = 1\}} + \tilde{\theta}_x^i(k) \tag{A.45}$$

Then, the steady-state probability distribution $\tilde{\mu}_n$ of $\tilde{\mu}_x^i(k)$, measured in packets, is given by

$$\tilde{\mu}_0 = \delta_0 \tilde{f}_0 \tag{A.46}$$

and for i = 1, 2, $\cdots$, 13,

$$\tilde{\mu}_i = \delta_0 \tilde{f}_i + \delta_1 \tilde{f}_{i-1} \tag{A.47}$$

and

$$\tilde{\mu}_{14} = \tilde{f}_{14} + \delta_1 \tilde{f}_{13} \tag{A.48}$$

Step 5: According to the definition of the fairness function used here, without loss of generality, we can let the debt limit $d_x^i = 0$. In this case, all negative states are combined into state 0. Then, the token balance process $E_x^i(k)$ is a standard discrete time *GI/GI/*1 queue given by

$$\tilde{E}_x^i(k) = \left[\tilde{E}_x^i(k-1) + r_x^i - L_x^i \min\left(\tilde{\mu}_x^i(k), \left\lfloor \frac{\tilde{Q}_x^i(k-1)}{L_x^i} \right\rfloor \right) \right]^+ \tag{A.49}$$

In this model the arrival and the service processes are described by r_x^i and

$$\overline{\mu}_x^i(k) = \min\left(\tilde{\mu}_x^i(k), \left\lfloor \frac{\tilde{Q}_x^i(k-1)}{L_x^i} \right\rfloor \right) \tag{A.50}$$

respectively. the steady-state probability distribution $\overline{\mu}_n$ for $\overline{\mu}_x^i(k)$ can be computed according to

$$\overline{\mu}_n = \tilde{\mu}_n \sum_{i \geq n} \sum_{j=0}^{52} \sum_{k=0}^{1} \tilde{\pi}_{i,j,k} + \tilde{\mu}_{>n} \sum_{j=0}^{52} \sum_{k=0}^{1} \tilde{\pi}_{n,j,k} \tag{A.51}$$

where $\tilde{\mu}_n$ is the updated service distribution for the queue length process obtained in Step 4, $\tilde{\mu}_{>n} = \sum_{k=n+1}^{14} \tilde{\mu}_k$, and $\tilde{\pi}_{i,j,k}$ is the steady-state queue length distribution computed in Step 1.

The transition matrix for a discrete time *GI/GI/*1 queue possesses a repeating structure. In our case, since the number of arrivals is deterministic and the number of bytes, which can be served in a frame, is a multiple of 53, the repeating structure is very special. Recall that the maximal number of bandwidth slots, which can be possibly assigned to a connection in a frame, is 14 or 14 × 53 bytes per frame, and the token generate rate is $r = r_x^i$ bytes per frame with $r < 14 \times 53$.

The structure of the transition matrix $\mathbb{P}_E$ for the token balance process $E_x^i(k)$ is illustrated as follows:

$$\mathbb{P}_E = \begin{array}{c|ccccccccccccc} & 0 & 1 & 2 & \cdots\cdots & \text{r-1} & \text{r} & \cdots\cdots & 53 & \cdots & \cdots & \cdots & \cdots & \cdots \\ \hline 0 & & & & & & * & & & & & & & \\ 1 & & & & & & & * & & & & & & \\ 2 & & & & & & & & * & & & & & \\ \vdots & & & & & & & & & \ddots & & & & \\ \vdots & & & & & & & & & & \ddots & & & \\ r-1 & & & & & & & & & & & * & & \\ r & & & & & & & & & & & & * & \\ r+1 & & & & & & & & & & & & & * \\ \vdots & & & & & & & & & & & & & \ddots \\ 53 & & & & & & * & & & & & & & \\ 54 & & & & & & & * & & & & & & \\ 55 & & & & & & & & * & & & & & \\ \vdots & & & & & & & & & \ddots & & & & \ddots \end{array} \tag{A.52}$$

All details about the transition blocks can be obtained in a similar fashion to that done in Step 1. Specifically, we partition the transition matrix $\mathbb{P}_E$ into blocks, each consisting of 53 × 53 entries:

$$\mathbb{P}_E = (\bar{\mathbf{P}}_{i,j}) \tag{A.53}$$

where

$$\bar{\mathbf{P}}_{i,j} = (\bar{u}_{m,n})_{53\times 53} \tag{A.54}$$

is a matrix consisting of 53 × 53 entries. Then, $\mathbb{P}_E$ is given by

$$\mathbb{P}_E = \begin{array}{c|cccccccccc} & 0 & 1 & 2 & \ldots & 13 & 14 & 15 & 16 & \ldots & \ldots \\ \hline 0 & \bar{\mathbf{B}}_1 & \bar{\mathbf{B}}_0 & & & & & & & & \\ 1 & \bar{\mathbf{B}}_2 & \tilde{\mathbf{C}}_1 & \bar{\mathbf{C}}_0 & & & & & & & \\ 2 & \bar{\mathbf{B}}_3 & \tilde{\mathbf{C}}_2 & \bar{\mathbf{C}}_1 & \bar{\mathbf{C}}_0 & & & & & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & & & & & \\ 13 & \bar{\mathbf{B}}_{14} & \tilde{\mathbf{C}}_{13} & \bar{\mathbf{C}}_{12} & \cdots & \bar{\mathbf{C}}_1 & \bar{\mathbf{C}}_0 & & & & \\ 14 & \bar{\mathbf{C}}_{15} & \tilde{\mathbf{C}}_{14} & \bar{\mathbf{C}}_{13} & \cdots & \bar{\mathbf{C}}_2 & \bar{\mathbf{C}}_1 & \bar{\mathbf{C}}_0 & & & \\ 15 & & \tilde{\mathbf{C}}_{15} & \bar{\mathbf{C}}_{14} & \cdots & \bar{\mathbf{C}}_3 & \bar{\mathbf{C}}_2 & \bar{\mathbf{C}}_1 & \bar{\mathbf{C}}_0 & & \\ \vdots & & & \ddots & \ddots & \ddots & \cdots & \ddots & \ddots & \ddots & \\ \vdots & & & & \ddots & \ddots & \ddots & \cdots & \ddots & \ddots & \ddots \end{array} \tag{A.55}$$

where $\bar{\mathbf{B}}_i$, $\tilde{\mathbf{C}}_i$ and $\bar{\mathbf{C}}_i$ are 53 × 53 matrices.

For the expression of $\bar{\mathbf{C}}_i$ with i = 1, 2, . . . , 14**,** Notice that all $\bar{\mathbf{C}}_1, \bar{\mathbf{C}}_2, ..., \bar{\mathbf{C}}_{14}$ possess the same structure:

$$\bar{\mathbf{C}}_i = (\bar{u}_{m,n})_{53\times 53} = \begin{pmatrix} & & & & *_1 & & & \\ & & & & & *_1 & & \\ & & & & & & \ddots & \\ & & & & & & & *_1 \\ *_{-1} & & & & & & & \\ & *_{-1} & & & & & & \\ & & \ddots & & & & & \\ & & & *_{-1} & & & & \end{pmatrix} \tag{A.56}$$

Specifically, all $\bar{u}_{m,n} = 0$ except for the following cases. For $0 \le m \le 52 - r$,

$$\bar{u}_{m,m+r} = \bar{\mu}_{i-1}$$

for $53 - \mathrm{r} \le \mathrm{m} \le 52$

$$\bar{u}_{m,m+r-53} = \bar{\mu}_i$$

For the expression of $\tilde{\mathbf{C}}_i$ with i = 1, 2, . . . , 13**:** $\tilde{\mathbf{C}}_i$ possesses the same structure as $\bar{\mathbf{C}}_i$ does. Specifically, all $\bar{u}_{m,n} = 0$ except for the following cases. For $0 \le m \le 52 - r$,

$$\bar{u}_{m,m+r} = \bar{\mu}_{i-1}$$

For $53\text{-}r \le m \le 52$,

$$\bar{u}_{m,m+r-53} = \bar{\mu}_{\ge i}$$

For the expression of $\bar{\mathbf{B}}_i$ with i = 1, 2, . . . , 15 with $\bar{\mathbf{B}}_{15} = \bar{\mathbf{C}}_{15}$, notice that all $\bar{\mathbf{B}}_1, \bar{\mathbf{B}}_2, ..., \bar{\mathbf{B}}_{15}$ possess the same structure:

$$\bar{\mathbf{B}}_i = \bar{\mathbf{P}}_{i-1,0} = (\bar{u}_{m,n})_{53\times 53} = \begin{pmatrix} & & *_1 & & & \\ & & & *_1 & & \\ & & & & \ddots & \\ & & & & & *_1 \\ & & & & & \\ & & & & & \\ & & & & & \end{pmatrix} \tag{A.57}$$

Specifically, all $\bar{u}_{m,n} = \mathbf{0}$ except for the following cases: for $0 \le m \le 52 - r$,

$$\bar{u}_{m,m+r} = \bar{\mu}_{\ge i-1}$$

For the expression of $\bar{\mathbf{B}}_0$, notice that both $\bar{\mathbf{B}}_0$ possesses the following structure:

$$\bar{\mathbf{B}}_0 = \bar{\mathbf{P}}_{i,i+1} = (\bar{u}_{m,n})_{53\times53} = \begin{pmatrix} & & & & \\ & & & & \\ *_{-1} & & & & \\ & *_{-1} & & & \\ & & \ddots & & \\ & & & *_{-1} & \end{pmatrix} \tag{A.58}$$

Specifically, all $\bar{u}_{m,n} = \mathbf{0}$ except for the following cases: for $53 - r \le m \le 52$,

$$\bar{u}_{m,m+r-53} = 1$$

For the expression of $\bar{\mathbf{C}}_0$, notice that both $\bar{\mathbf{C}}_0$ possesses the following structure:

$$\bar{\mathbf{C}}_0 = \bar{\mathbf{P}}_{i,i+1} = (\bar{u}_{m,n})_{53\times53} = \begin{pmatrix} & & & & \\ & & & & \\ *_{-1} & & & & \\ & *_{-1} & & & \\ & & \ddots & & \\ & & & *_{-1} & \end{pmatrix} \tag{A.59}$$

Specifically, all $\bar{u}_{m,n} = \mathbf{0}$ except for the following cases: for $53 - r \le m \le 52$,

$$\bar{u}_{m,m+r-53} = \bar{\mu}_0$$

However, the token balance process $E_x^i(k)$ is not stable since the arrival rate r_x^i is larger than the average service rate of $\bar{\mu}_n$. For implementation, we truncate the process by imposing a maximal possible value N_E for the token balance process, which will be always stable. The transition matrix of this truncated process is the same as the north-west corner, with $(N_E + 1) \times (N_E + 1)$ block entries, of the transition matrix $\boldsymbol{P}_{\mathrm{E}}$ except then block entry in the furthest south-east corner of the truncated matrix, which is equal to $\bar{\mathbf{C}}_1 + \bar{\mathbf{C}}_0$ instead of $\bar{\mathbf{C}}_1$. Use the GTH-algorithm to find the stationary distribution $\tilde{e}_x^i(n)$ for n=0,1,2,...,53(N_{E}+1) of the truncated process, which will be used in Step 3 of the next recursion. The whole recursion is now complete.

Go to Step 1 for a new recursion using the updated service distribution $\tilde{\mu}_k$ obtained in Step 4 and continue the process until convergence. Let $\bar{Q}_x^i(old)$ and $\bar{Q}_x^i(new)$ be the average number of packets in the data buffer of connection i in the type x, obtained from the previous recursion and the current recursion, respectively. In the implementation, we use

$$\max_x \left| \bar{Q}_x^i(old) - \bar{Q}_x^i(new) \right| < \varepsilon \tag{A.60}$$

as a criterion to stop the recursion process.