Bandwidth Allocation for Quality of Service Provision in IEEE 802.16 Systems

Tze Wei Tang

Thesis submitted for the degree of

Doctor of Philosophy

in

Electrical and Electronic Engineering

at

The University of Adelaide

(Faculty of Engineering, Computer and Mathematical Sciences)

School of Electrical and Electronic Engineering

March 3, 2009

Chapter 1

Introduction

In recent years, wireless last mile technology has become a challenging competitor to wired last mile technologies, such as DSL, cable modem and fiber-optic cable. The main reasons for this phenomenon stem from potential advantages in coverage, cost, speed of deployment and improving data-rate when compared with other competing technologies. One of these competing wireless technologies is the Institute of Electrical and Electronics Engineers (IEEE) developed Broadband Wireless Access (BWA) system, which is defined by the IEEE 802.16 standard.

Alternative standards have also been developed. In Europe, the European Telecommunications Standards Institute (ETSI) defines a different standard, which is called High Performance Metropolitan Area Network (HiperMAN). In Korea, the Korean telecoms industry has defined the Wireless Broadband (WiBro) standard. All of these technologies are independent and non-interoperable. However, there is an organisation formed by the industry leaders, called the Worldwide Interoperability for Microwave Access (WiMAX) forum, which is committed to promoting the open interoperability of all products used in broadband wireless access based upon the harmonised IEEE 802.16 or ETSI HiperMAN standard.

Due to various types of traffic carried in the current Internet, such as voice, video and web, it is important for a BWA system to provide various Quality of Service (QoS) strategies. In this thesis, we investigate the bandwidth allocation process in

the IEEE 802.16 systems to provide QoS guarantees to end users.

1.1 Background

1.1.1 Quality of Service

Quality of Service is a general term that defines a set of quality requirements on the collective behaviour of one or more objects in a network [3], [4]. However, the perceived QoS by different end users may be a subjective matter. Service providers typically measure aspects of network performance in an attempt to infer the user perceived QoS. Examples of measured parameters are delay, jitter, throughput and loss rate. QoS can also include aspects such as service availability, security and reliability.

Today's Internet provides large-scale information transfer and sharing without the assurance of QoS. The concept of QoS has evolved considerably with the emergence of interactive and real-time services over the Internet. Real-time services such as Voice over IP (VoIP), teleconferencing and live video streaming, must meet strict requirements in delay, variation of delay, loss rate and throughput. On the other hand, non-real-time elastic services such as file transfer and web browsing, have looser QoS requirements. Providing QoS control mechanisms improves network performance, which may include faster web page loading, faster file download, better real-time audio and video, and nearly no lag real-time gaming. For example, for a real-time VoIP connection to give a good QoS to an end user, a minimum transmission bandwidth must be available on the entire end to end path. On top of that, the end-to-end delay and jitter must not severely deteriorate the audibility of the VoIP session.

Having defined various components of QoS, one of the main objectives of a scheduling mechanism is to fulfill the QoS requirements of the network. In most research, improving QoS is considered as providing delay, jitter, throughput or loss

rate guarantees to the end users. In this thesis, we refer to a customer as receiving "good service" if all the customer's specified QoS requirements are met.

1.1.2 Bandwidth Allocation

There are many techniques that help to improve the QoS of a network, and these techniques can be employed in different layers of the Open Systems Interconnection (OSI) reference model [5], most often at the data link layer (see Figure 1.1.1). The data link layer is further divided into two sub-layers: Medium Access Control (MAC) and Logical Link Control (LLC). The MAC layer is responsible for the organisation of the access to the medium. Whilst, the LLC layer provides an interface for the upper layers to deal with any type of MAC layer. The layer below the data link layer is the physical (PHY) layer. The PHY layer is responsible for the physical communication through the medium, which is wireless in this thesis. The PHY layer describes the electrical properties and the interpretation of the exchanged signals through the medium.

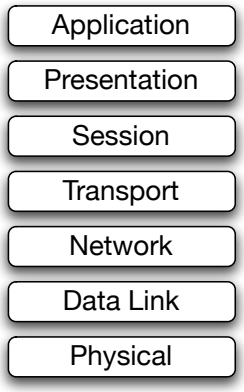

Figure 1.1.1: The 7 layers of the OSI reference model.

Data is generated at the application layer (top layer) and passed down to the layers below. Data passed down from the network layer to the data link layer is known as Service Data Units (SDUs). In the MAC layer, SDUs are stored in a buffer

where they wait to be transmitted. When a SDU is scheduled for transmission, it is pre-pended with a MAC layer header and passed down to the PHY layer for actual physical transmission to one or more receivers.

In Figure 1.1.2, we show the MAC and PHY layers defined in the IEEE 802.16 standard [1]. Service Access Point (SAP) is the point in a protocol stack where a higher layer has access to the services of its lower layer [1]. The MAC layer consists of 3 sublayers: service-specific convergence sublayer (CS), MAC common part sublayer and security sublayer. The convergence sublayer is responsible for mapping of external network data into MAC layer data units and this includes associating each connection a connection identifier (CID). Multiple CS specifications may be provided to interface with various protocols, such as Asynchronous Transfer Mode (ATM) networks. The CS is also responsible for optional Payload Header Suppression (PHS). The MAC common part sublayer provides the core MAC functionalities, such as bandwidth allocation schemes, connection establishment and connection maintenance. Lastly, the security sublayer is responsible for authentication, secure key exchange and encryption.

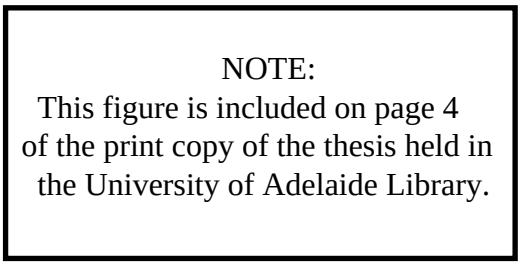

Figure 1.1.2: IEEE 802.16 protocol layers obtained from [1]

In this thesis, we investigate the bandwidth allocation mechanism in the MAC

layer of 802.16 systems. Bandwidth allocation involves mechanisms to distribute bandwidth or transmission opportunities to different users/connections, and therefore, these mechanisms have a great impact on the QoS of connections in the systems. For example, a connection that is only allocated a minimal amount of transmission opportunities may experience a very bad QoS, in terms of throughput and delay. Therefore, by managing the way bandwidth is utilised in the MAC layer of a network, the QoS of the end users may be improved, especially during periods of congestion. In addition, it is also common to consider fairness issues in the design of a bandwidth allocation process.

The term "bandwidth" is typically assigned to the idea of bit rate, or the amount of data a user can transmit, or receive, in a given period. Bandwidth can be thought of, more generally, as access to a finite network resource. Therefore, the bandwidth of a system can be quantified in many ways, such as transmission time, number of bits per second and number of packets per second. In a wired environment, bandwidth is normally quantified in terms of the number of bits per second. However, in a wireless environment, bandwidth may be described in terms of time.

There are a lot of different scheduling disciplines defined for allocating bandwidth. One of the simplest scheduling disciplines is First-In First-Out (FIFO), where connections are allocated bandwidth in accordance to their packets' arrivals. On the other hand, Round Robin (RR) allocates equal bandwidth to each connection in the system. Some of the more complicated scheduling disciplines include Weighted Round Robin (WRR) [6], Deficit Round Robin (DRR) [7], Earliest Deadline First (EDF) [8] and Weighted Fair Queue (WFQ) [9].

In this thesis, we propose the Dual-Queue scheduling scheme to improve the QoS of 802.16 networks in terms of maximising the number of users that experience good service. This scheduling scheme can operate simultaneously with any of the fore-mentioned scheduling disciplines. Hence, this approach provides an extra layer of QoS control for 802.16 systems, which has not yet been considered elsewhere in the literature.

1.2 Thesis Structure

In Chapter 2, we introduce the IEEE 802.16 standard and describe the PHY and MAC specifications, which provide the bandwidth allocation, or the scheduling process, in the MAC layer. We then present related work on the topic of providing QoS in an 802.16 system, and identify the key requirements of a MAC scheduler.

After introducing 802.16 systems in Chapter 2, we describe the scheduler design problem from a general perspective and discuss alternative objectives that a scheduler may try to achieve in Chapter 3. This approach is comprehensive as it covers the requirements of both the network provider and the end users simultaneously. Hence, we propose a MAC scheduler design framework that is driven by a set of objectives. One of the objectives that stems from our investigation is the objective of maximisation of the number of satisfied customers.

Motivated by Chapter 3, we investigate approaches for achieving customer satisfaction in Chapter 4. From the related work, there is strong evidence that a firm causal linkage exists between customer satisfaction and service performance. Hence, we intend to exploit the Dual-Queue (DQ) scheduling discipline proposed by Hayes *et al.* [2] to maximise the number of satisfied connections in 802.16 systems. We first analyse the original DQ discipline in detail and generalise the core DQ mechanisms. We then discuss and explain the reasons why we can not directly deploy the original DQ algorithms from Hayes' work into an 802.16 environment.

Chapter 5 is the core of this thesis, where we propose a Dual-Queue implementation for 802.16 systems that handles real-time services in both the DL and UL. We propose a framework for implementing the DQ scheduling discipline in an 802.16 environment, which covers issues central to DQ scheduler design, such as the structure of the α-queue and the β-queue, handling Automatic Repeat Request traffic and the service scheme deployed by the DQ scheduler. We then propose the core DQ mechanisms for the DL and UL respectively. We compare the performance of our proposed DL and UL Dual-Queue schedulers to a WFQ scheduler, including

the impact of bit errors and the use of Automatic Repeat Request. In addition, we investigate the impact of enhancing the WFQ scheduler with an Explicit Packet Dropping mechanism since the DQ scheduler includes an Explicit Packet Dropping mechanism as an integral component. A performance comparison is carried out based on the overall proportion of "good service" experienced by each connection, the network utilisation and the total throughput. Finally, we show that our DQ system can handle mixed traffic profiles, such as video and voice.

In Chapter 6, we investigate alternative objectives that the DQ scheduler may try to achieve. We then propose the Priority-Based Dual-Queue scheduler, which is made up of multiple DQs differentiated by each connection's priority class. The Priority-Based Dual-Queue ensures connections that belong to the highest priority class are served ahead of connections that belong to lower priority classes at all times. This criterion may not be achieved by our proposal in Chapter 5, in environments in which priorities can dynamically change.

In Chapter 7, we investigate the benefits of carrying out the DL and the UL scheduling jointly. We propose a "basic coordination" scheme that aims to maximise the number of one-directional connections that experience "good service" in the network. We then extend our investigation to scenarios where the network traffic consists of bi-directional sessions, where a video conferencing, or Voice over IP session, consists of bi-directional connections on both the DL and UL. Based on the basic coordination scheme, we then propose two schemes, partial and full coordination, which are able to maximise the number of bi-directional sessions that experience "good service" in the network.

Chapter 8 presents a summary of this thesis and outlines some potential future research avenues.

1.3 Major Research Contributions

This section outlines the major research contributions made by this thesis.

- We have developed a framework for MAC scheduling design in 802.16 systems, which is based on a comprehensive objective framework, incorporating user and network views simultaneously.

- We have included Subscriber Station differentiation into the scheduling design of 802.16 systems.

- We have developed an implementation of the Dual-Queue in both the DL and the UL of 802.16 systems for handling real-time services. We have developed a framework for DQ implementation, which addresses issues concerning the structure of the α-queue and the β-queue, handling Automatic Repeat Request traffic and the service scheme deployed by the Dual-Queue scheduler. We have also developed the core DQ mechanisms for the Downlink and the Uplink of the systems.

- We have compared the performance of our proposed Dual-Queue scheduler against a WFQ scheduler and a modified version of WFQ scheduler enhanced by an Explicit Packet Dropping mechanism. We have investigated various aspects of these schedulers, which include maximising the number of connections that experience "good service", achieving a lower use of network resources and maximising total system throughput that experiences "good service". We have also shown the positive impact of having Explicit Packet Dropping mechanism in 802.16 systems.

- We have developed the Priority-Based Dual-Queue scheduler, which improves the options available to service providers. The Priority-Based Dual-Queue scheduler ensures connections that belong to the highest priority class are served ahead of connections that belong to lower priority classes at all times.

- We have shown the importance of scheduling the DL and UL jointly in order to maximise the number of one-directional connections that experience "good

service" through a joint DL and UL scheduling scheme, called the "basic coordination" scheme.

- We have developed two joint DL and UL scheduling schemes based on the basic coordination scheme, "partial coordination" and "full coordination", which are able to maximise the number of bi-directional sessions that experience "good service" in an 802.16 system. These bi-directional sessions include Voice over IP and video conferencing. Also, we contend that the choice of which scheme to implement depends on the interpretation of the performance requirement of the bi-directional sessions by the network provider.

1.4 List of Publications

- T. W. Tang, D. Green, M. Rumsewicz and N. Bean, "An Architecture for IEEE 802.16 MAC Scheduler Design," in *IEEE International Conference on Networks*, November 2007, pp. 89–94.

1.5 Summary

This chapter has introduced the concept of bandwidth allocation and has discussed the various components of QoS. We have also presented the road map of this thesis and outlined the major research contributions of our work. The following chapter will explore the IEEE 802.16 standard in detail.

Chapter 2

Scheduling in 802.16 Systems

In this chapter, we provide a brief overview of the extensions of the IEEE 802.16 standard and describe the PHY and MAC specifications of the standard. We then review the current literature on 802.16 MAC scheduling and identify the key requirements of the MAC scheduling process.

We note that in the 802.16 standard, the scheduling algorithms applied to different traffic types are intentionally left unspecified. The standard only outlines the framework for scheduling traffic and includes definitions of several classes of service (CoSs). Hence, it is up to the vendors to provide efficient and robust schedulers, which permits considerable flexibility in QoS provisioning.

2.1 The IEEE 802.16 Standard

The first 802.16 standard (IEEE 802.16-2001) was published by the IEEE 802.16 Working Group in April 2002 [10]. It contains the description of the MAC and PHY layers for fixed broadband wireless access (FBWA) targeting the frequency band between 10 and 66 GHz. Due to the high carrier frequency (single-carrier), line-of-sight operation is required. Further extensions of the standard are briefly described below.

- IEEE 802.16a [11]: The first extension includes the frequency band between 2 and 11 GHz, and also includes non-line-of-sight operation. Further, the PHY layer is extended to include multi-carrier transmission.

- IEEE 802.16c [12]: This extension addresses the detailed system profiles for 10- 66 GHz operation. It also adds support for multiple-input-multiple-output (MIMO) systems.

- IEEE 802.16-2004 or IEEE 802.16d [1]: This extension was initiated in order to align the standard with the ETSI HiperMAN standard. It is the most popular extension, and has superceded the IEEE 802.16a and IEEE 802.16c standards. It supports both time division duplexing (TDD) and frequency division duplexing (FDD). In this thesis, we focus on this standard, and hence, most of the PHY and MAC features explained herein are based on this extension.

- IEEE 802.16e-2005 [13]: Mobility aspect is added into the standard through this extension. In order to support mobility, the physical specifications are extended to increase resistance to multi-path interference.

- IEEE 802.16f [14]: The support for a management information base (MIB) is addressed in this extension. A MIB contains information used for managing all the devices in the network in both the base station and the subscriber stations.

- IEEE 802.16g [15]: This extension addresses the management plane procedures and services. It aims to investigate the co-existance of FBWA networks operating in the license exempt bands.

2.2 802.16 System Description

An IEEE 802.16 system consists of two major entities: the Base Station (BS) and the Subscriber Station (SS). The BS acts as a central controller and provides com-

munication services to a set of SSs in its region of service. There are two operational modes defined for the systems: point-to-multipoint (PMP) mode, where SSs are allowed to communicate with the BS only and mesh (MESH) mode, where SSs are allowed to communicate with each other. In this thesis, we only consider 802.16 systems that operate in PMP mode.

Bi-directional communication is maintained between the BS and the SSs: transmission from the BS to the SSs is called the Downlink (DL) and transmission from the SSs to the BS is called the Uplink (UL). The BS is responsible for scheduling the DL and the UL data transmission between the BS and the SSs.

In this section, we briefly describe the PHY and MAC layer features that are useful for MAC scheduling design, and relevant to the research described later in this thesis.

2.2.1 Physical Layer Overview

Physical Layer Type

There are two categories of PHY supported by the 802.16 standard: single-carrier and multi-carrier. The single-carrier PHY has two specifications: WirelessMAN-SC (single-carrier) for frequencies from 10 to 66 GHz and WirelessMAN-SCa for frequencies below 11 GHz.

For multi-carrier PHY, there are also two specifications: WirelessMAN-OFDM (Orthogonal Frequency Division Multiplex) and WirelessMAN-OFDMA (Orthogonal Frequency Division Multiple Access), both for frequencies below 11 GHz.

Channel Bandwidth

The channel bandwidth is the amount of spectrum needed for data transmission in a channel. From the 802.16 standard, channel bandwidth ranges from 1.25 MHz to 28 MHz [16], depending on the PHY type. Note that channel bandwidth is a physical layer parameter. A larger channel bandwidth provides higher available data

rates, as more physical resource is used.

Modulation Schemes

There are different PHY modulation schemes supported in the 802.16 system, which include QPSK (Quadrature Phase Shift Keying), QAM-16 (Quadrature Amplitude Modulation) and QAM-64. QPSK has advantages in robustness, while QAM has higher bit rates. Further, each of these techniques may have different channel codings.

The IEEE 802.16 system supports Adaptive Modulation and Coding (AMC), which allows different modulation schemes and channel codings. This enables different data transfer rates, according to the physical environment, to be experienced by the SSs and/or the technology supported by the SSs. In general, a combination of a PHY modulation scheme and a channel coding rate, to transfer data at a certain physical rate, is called a PHY mode. PHY modes with lower bit rates are more robust transmission modes, as there is a direct trade-off between data bits and robustness.

According to [1] and [17], the highest aggregate raw data rate achievable for an 802.16 system is 135 *Mbps* (based on the modulation schemes and channel bandwidth used) for line-of-sight (LOS) operation. For non-line-of-sight (NLOS) operation, the highest aggregate raw data rate achievable is lower, up to 75 *Mbps* [1], [18], [19].

The PHY mode of a SS can change during operation. Further, the PHY mode used for the DL of a SS may be different from the PHY mode used for its UL. The decision on which PHY mode to be used on either the DL or the UL of a SS is made by the BS.

Burst Profiles

In order to save overhead due to changing PHY modes, data transmissions using the same PHY mode are normally grouped together. Each of the DL or the UL bursts

is described by its burst profile which contains information such as the modulation scheme, forward error correction (FEC, which is used for error correction in the physical layer), and guarding times.

Framing

Data transmission in IEEE 802.16 systems is carried out on a per PHY frame basis. The duration of the PHY frames is chosen during the configuration process and does not change during the operation of the system. The 802.16 standard supports multiple PHY frame durations for each of the PHY types mentioned above. In the literature, the term "MAC frame" is used equivalently to "PHY frame".

Physical Slots

Physical slots (PSs) are used in the IEEE 802.16 standard to express quantities of time (slots) available for bandwidth allocation [16]. PS is the basic unit of bandwidth allocation. The duration of a PS depends on the PHY type. For WirelessMAN-SC and WirelessMAN-SCa, it is defined to be four QAM symbols in duration. For WirelessMAN-OFDM and WirelessMAN-OFDMA, it is defined to be $4/F_s$, where F_s is the sampling frequency. Hence, the duration of a PS doesn't change after configuration or during the operation of an 802.16 system.

The overheads and the amount of data that can be transmitted in a PS vary with the modulation schemes supported in the IEEE 802.16 standard, even though the PSs have a constant duration. In this thesis, the term "slot" refers to a PS unless otherwise specified.

Duplexing

Duplexing defines how bi-directional communication (the DL and the UL) is achieved between a BS and a set of SSs. Stations may be capable of full-duplex (transmit and receive at the same time) or half-duplex (transmit or receive but not both at

the same time) operation. The IEEE 802.16 standard supports both time division duplexing and frequency division duplexing.

In TDD, the DL and the UL transmissions are scheduled at different times during a frame using the same frequency channel, where a frame is divided into the DL and the UL sub-frames, which do not overlap. For FDD, the DL and the UL transmissions may or may not be scheduled at the same time, using different frequency channels. In TDD, the time duration for the DL or the UL sub-frame may be fixed or adaptive. Adaptive TDD (ATDD) allows the DL/UL ratio to be dynamically changed over time, based on traffic conditions, or on some specific goals of the network. Therefore, ATDD is preferred when the traffic on the DL and the UL is asymmetrical.

2.2.2 Medium Access Control Layer Structure

The MAC layer is responsible for controlling access to the transmission medium, or allocating bandwidth to the BS and SSs for the DL and the UL transmissions. Hence, the MAC layer is also responsible for QoS provision. In this thesis, we are interested in the design of the DL and the UL schedulers in the MAC layer for QoS provision. The different aspects of MAC scheduling are discussed in detail in Section 2.5.

Apart from allocating bandwidth, the MAC layer is also responsible for basic functions such as data encapsulation, automatic repeat request (ARQ), fragmentation and packing. Further, security is also an important aspect in the MAC layer.

Data Encapsulation

The MAC layer is responsible for encapsulating higher layer packets or Service Data Units in MAC layer protocol data units (PDUs), for delivery over the medium, to a single receiver or multiple receivers (broadcast). The necessary information for the delivery is stored in the generic MAC header of the MAC PDU (MPDU). The

802.16 standard specifies the size of the MAC header to be 6 bytes.

Automatic Repeat Request

Even though the physical layer may have its own error correction methods, the ARQ mechanism is mandatory in the MAC layer of 802.16 systems for handling MPDU errors. An ARQ algorithm involves error detection of an MPDU, a feedback strategy to inform the sender about the error and a retransmission strategy to be used by the sender.

Fragmentation and Packing of SDUs

Fragmentation of packets involves the sub-division of SDUs into smaller fragments for transmission. Some of the benefits of fragmentation discussed in [16] include higher probability of successful delivery for smaller fragments, higher spectral efficiency and improved QoS support. Similarly, packing of SDUs or fragments is needed when a PS can accommodate more than one SDU or fragment. Note that fragmentation and packing introduces an extra overhead to the network. The 802.16 standard specifies the sub-header for fragmentation and packing to be either 2 or 3 bytes, depending on the ARQ algorithm used.

2.2.3 Class of Service for MAC Layer QoS Provision

In order to differentiate between traffic connections with different QoS requirements, there are 5 CoSs defined in the IEEE 802.16e standard [13], as follows:

1. **Unsolicited grant service (UGS):** UGS is introduced to support real-time traffic connections that generate fixed-size data packets on a periodic basis. Examples of such traffic types are, Voice over Internet Protocol without silence suppression and Constant Bit Rate (CBR) services. UGS traffic will be offered fixed-size grants on a periodic basis, without SSs needing to poll for bandwidth.

2. **Real-time polling service (rtPS):** On the other hand, rtPS is designed to support real-time traffic connections that generate variable-size data packets on a periodic basis, such as video streaming. Hence, it depends on unicast polling opportunities to request bandwidth from the BS and requires more overhead than UGS.

3. **Extended-real-time polling service (ertPS):** This CoS offers the advantages of both UGS and rtPS. It is designed to support real-time traffic connections whose bandwidth requirements change with time. The ertPS traffic will be offered grants on a periodic basis but can also make use of unicast polling opportunities to request more bandwidth. Note that this CoS was not introduced until the IEEE 802.16e standard was proposed.

4. **Non-real-time polling service (nrtPS):** To support delay-tolerant traffic connections with variable-size data packets on a regular basis, nrtPS is also introduced in the 802.16 standard. SSs depend on broadcast polling (contention polling) opportunities to request bandwidth. Even though nrtPS is allowed to make use of unicast polling opportunities, these opportunities are very rare, compared to the unicast polling opportunities for rtPS or ertPS.

5. **Best effort service (BE):** This CoS is designed for traffic connections with minimum QoS support. It makes use of broadcast polling opportunities only for requesting bandwidth.

2.3 Related Work

There has been extensive research work presented in support of QoS in IEEE 802.16 systems, especially on UL scheduling algorithms. In the following, we provide a brief overview of the broad spectrum of research carried out to provide QoS to 802.16 systems. We classify the related work into the following categories.

- **Homogeneous algorithms:** In this category, an initial performance analysis of the MAC and the PHY layers is carried out. The performance metrics considered include throughput, delay, packet loss, network utilisation and fairness. Some of the legacy scheduling schemes used in a wired environment are employed to provide QoS guarantees to all CoSs in the 802.16 wireless environment, such as Weighted Fair Queueing, Earliest Deadline First, Round Robin, Deficit Round Robin, Weighted Round Robin and First-in First-out.

- **Hybrid algorithms:** In hybrid algorithms, different scheduling schemes are combined together to optimise the QoS performance of the 802.16 systems. For each of the CoS, a certain scheduling scheme is proposed. In some cases, a hierarchical scheduling structure is proposed.

- **Algorithms with Adaptive Modulation and Coding:** In this category, the importance of considering AMC in the scheduling process is highlighted, as this aspect affects the performance of the network, in terms of throughput, delay, network utilisation and fairness.

- **Algorithms for Connection Admission Control:** Potential Connection Admission Control (CAC) mechanisms for 802.16 systems are proposed to complement scheduling process in order to provide a greater QoS to the system. The CAC mechanism is not specified and is in fact optional in the 802.16 standard.

- **Specific algorithms:** A specific feature of the 802.16 systems is exploited, which aims to improve the QoS of the systems.

- **Other technologies:** Other technologies that deal with scheduling problems similar to those in 802.16 systems.

2.3.1 Homogeneous Algorithms

There are many papers that discuss the MAC and the PHY operations of the IEEE 802.16 systems, including [17], [20]–[27]. These papers also provide an initial performance evaluation and analysis for the systems, especially in terms of throughput and delay.

Gusak *et al.* [28] investigate the performance of the MAC layer with the WRR and the WFQ scheduling schemes. Their findings show that there is not much difference in delay performance between the two scheduling schemes. They also show that adaptive partitioning of the UL and the DL sub-frames plays an important role in improving network performance as the network load increases or as the channel condition changes.

In [29], Cicconetti *et al.* consider the WRR and the DRR algorithms as possible candidates for scheduling the UL and the DL of the 802.16 systems respectively, operating in the FDD mode. On the other hand, Cicconetti *et al.* [30] discuss the trade-off between average delay and throughput with respect to frame duration, that is, a longer frame duration results in a higher average delay, but a higher throughput. Further, the delay experienced by the UL connections is shown to be highly dependent on the delay introduced by the bandwidth request mechanism. Similarly, Cho *et al.* [31] propose a priority based scheduling scheme with a dynamic bandwidth allocation mechanism for all CoSs defined in the 802.16 systems . The authors evaluate their proposed scheme based on the throughput of different CoS.

Unlike [29], Ruangchaijatupon *et al.* [32] analyse the delay and throughput performance of the EDF and the WFQ algorithms (where a weight is assigned to a CoS, rather than a connection) for different CoSs under various network utilisation conditions. Their findings show that the EDF scheme gives more transmission opportunities to delay sensitive traffic, and hence starvation of the BE traffic could occur. On the other hand, Shang and Cheng [33] propose a hierarchical scheduling model based on the WF^2Q+ algorithm [34], which is designed to distribute weighted

bandwidth to various grouped traffic.

Further, Ruangchaijatupon and Ji [35] propose to use the DRR algorithm with an adaptive quantum to handle polling services in the 802.16 systems, including the ertPS, rtPS and nrtPS traffic. The quantum assigned to each connection is based on its current queue size. The DL connections are given a higher priority than the UL connections, following the reasons given in [36], which aim to prevent overflow of the BS's buffer and guarantee latency requirements. In [37], Ruangchaijatupon and Ji extend their work in [35] by using the deficit fair priority queueing (DFPQ) algorithm instead.

Hawa and Petr [38] propose a new scheduling architecture, which emphasizes a guarantee of the delay requirement of the UGS traffic and provides minimum bandwidth reservation for the rtPS, nrtPS and BE traffic. In order to provide a tight delay guarantee to the UGS traffic, the scheduler implements a strict semi-premptive priority for the UGS traffic. For the rtPS, nrtPS and BE traffic, a priority-enchanced WFQ is used. Whenever two connections have the same chance of being served under the WFQ scheduling rule, the connection with the highest priority will be served. The authors also introduce a dynamic mini-slot allocation scheme, which they claim improves the performance of their proposed scheduling scheme under varying load conditions.

In [39], Niyato and Hossain propose a queue-aware adaptive uplink bandwidth allocation and rate control mechanism to handle polling services, in order to maintain packet-level QoS performance at the desired level. They also present a comprehensive analytic queueing model for their proposed scheme in both steady and transient states. The same authors propose a joint bandwidth allocation and CAC algorithm based on queueing theory in [40]. A threshold is assigned to each CoS in order to limit the amount of bandwidth allocated to each. The authors also formulate an optimisation problem for allocating bandwidth to each connection of a CoS, based on utility functions. The utility of a connection depends on bandwidth, average delay, throughput and admission control decisions.

2.3.2 Hybrid Algorithms

Chu *et al.* [41] propose a QoS architecture for 802.16 systems, which is based on priority scheduling and dynamic bandwidth allocation. For the UGS and the rtPS traffic that have a higher priority, the WFQ algorithm is used. For the nrtPS and BE traffic that have no delay requirement, the WRR algorithm is used. However, the authors have not provided any simulation results to show the benefits of their proposed architecture.

Unlike [41], Wongthavarawat and Ganz [42] propose the use of the EDF algorithm for the rtPS traffic, the WFQ algorithm for the nrtPS traffic and the FIFO algorithm for the BE traffic. The UGS traffic, which has the highest priority, is scheduled ahead of the other traffic. Further, the overall allocation of bandwidth is done in a strict priority order. Hence, a higher priority traffic is given transmission opportunities ahead of a lower priority traffic, which may cause the lowest priority traffic to starve when the network is congested with higher priority traffic.

In [36], Chen *et al.* propose a two-layer scheduling algorithm for the bandwidth allocation process. At the first layer, a DFPQ algorithm is used to distribute bandwidth among all CoSs on both the DL and the UL, except UGS traffic, which is scheduled ahead of the other CoSs. At the second layer, the EDF, WFQ and RR algorithms are used to handle connections that belong to rtPS, nrtPS and BE traffic respectively. Further, DL traffic is given a higher priority than UL traffic in order to prevent overflow of the BS's buffer and to guarantee latency requirements. We contend that these reasons are insufficient to justify giving a higher priority to the DL traffic.

Unlike the other approaches, Lin and Sirisena [43] propose a scheduling architecture called the multi-class uplink fair scheduling structure to support bandwidth and delay guarantees for all CoSs. The proposed scheduling scheme for the DL is a modified WRR algorithm. For the UL, the authors propose a modified WFQ algorithm to handle the UGS and the rtPS connections, a modified WRR algorithm to han-

dle the nrtPS connections, and the FIFO algorithm to handle the BE connections. However, detail of the modifications are not provided in the paper.

On the other hand, Vinay *et al.* [44] propose a hybrid algorithm to schedule real-time services, which is based on a combination of the EDF and the WFQ algorithms. The authors show that their proposed hybrid algorithm has a better delay performance than one using only the EDF algorithm to schedule the UL real-time services. The authors also show that their algorithm has a better delay performance with the grant per subscriber station (GPSS) scheme than with the grant per connection (GPC) scheme (see Section 2.5 for the description of the GPSS and the GPC schemes.)

A summary of the different scheduling algorithms used in these hybrid schemes is shown in Table 2.3.1. Note that in all these papers, the ertPS traffic is not considered as this CoS is newly added in the 802.16e standard.

Table 2.3.1: Summary of the different scheduling algorithms used in the hybrid schemes.

Paper / CoS	UGS	rtPS	nrtPS	BE
[41]	WFQ	WFQ	WRR	WRR
[42]	fixed	EDF	WFQ	FIFO
[36]	DFPQ	DFPQ + EDF	DFPQ + WFQ	DFPQ + RR
[43]	modified WFQ	modified WFQ	modified WRR	FIFO
[44]	EDF + WFQ	EDF + WFQ	EDF + WFQ	EDF + WFQ

2.3.3 Algorithms with Adaptive Modulation and Coding

Rath *et al.* [45] propose a modified version of the DRR algorithm called the opportunistic DRR (O-DRR), which takes into account variations in the wireless channel and delay requirement of various traffic. This scheduling algorithm is based on

the polling interval of the SSs, and so the authors provide an analytical technique to obtain an optimal poling interval. The authors claim that their low-complexity scheduling scheme is able to achieve good fairness-delay performance with multi-class traffic.

On the other hand, Sayenko *et al.* [46] propose a simple solution for allocating slots based on the QoS requirement, bandwidth request size and slot size of various modulation and coding schemes. The authors contend that their scheduling scheme should allocate a minimum number of slots to each connection to ensure that its basic QoS requirement is met. They also show that the ordering of slots has an impact on QoS guarantees.

Similar to [40], which treats a scheduling process as an optimisation problem, Singh and Sharma [47] propose a scheduling scheme based on minimising unsatisfied demand, which is the sum of the number of un-transmitted bytes in the queue. In fact, this is equivalent to maximising the throughput of the network, provided AMC is supported. The authors also propose a fair version of their algorithm.

In [48], Settembre *et al.* propose the use of the WRR algorithm to handle rtPS and nrtPS traffic and the RR algorithm to handle BE traffic. In addition, they consider variable size data packets, fragmentation and packing, payload header suppression (PHS) and AMC. Their algorithm also consists of a strict priority policy among different CoSs.

In order to differentiate between connections, Liu *et al.* [49] propose a scheduling scheme based on the priority of connections for the rtPS, nrtPS and BE traffic, where a priority is assigned to a connection based on its channel quality, QoS satisfaction and CoS. Connections are served in order from the highest priority to the lowest priority. When a connection experiences good QoS, its priority is reduced to give more transmission opportunities to other connections.

2.3.4 Algorithms for Connection Admission Control

In [50], Tsai *et al.* propose an uplink scheduling scheme based on the EDF algorithm and bandwidth estimation information. The rate of connections in the network is controlled by a token bucket. UGS traffic is given the highest priority and hence, it is served ahead of other CoSs. Further, the authors propose a CAC mechanism that also uses a token bucket. The authors also present a mathematical model and claim that it predicts the delay and loss of rtPS connections precisely.

Similar to [50], Wang *et al.* [51] propose a CAC scheme that provides the highest priority for UGS traffic and maximises bandwidth utilisation by a bandwidth borrowing and degradation mechanism. The authors also propose an analytic model to evaluate their proposed scheme in terms of throughput. The same authors analyse the upper bound of the blocking probability of all CoSs using the Chernoff bound method and the Gaussian model for aggregated traffic in a large network [52]. Further, the authors claim that their proposed upper bound of blocking probability is more efficient than the Erlang B formula from a numerical perspective.

On the other hand, Yang and Lu [53] propose a CAC and a scheduling mechanism for real-time video applications. They claim that their CAC and scheduling schemes are the first that considers throughput expectation, delay requirement and fairness simultaneously. They claim that their proposal is able to achieve a high throughput by introducing a pending period for every session to precisely arrange its access time, with help from coordination of the I-frames and the Non-I-frames of the video session. Their scheduling scheme is based on the EDF algorithm, and it makes use of the latest starting time (LST) as a measure of the urgency of a packet.

Unlike the other CAC algorithms proposed, which are predominantly based on connections' bandwidth, Chandra and Sahoo [54] propose a CAC scheme that considers QoS guarantees in terms of bandwidth, delay and jitter. Their proposed CAC scheme admits a new connection if the QoS of all the admitted connections can be guaranteed. Further, the authors propose a method for estimating the bandwidth

of rtPS and nrtPS connections.

2.3.5 Specific Algorithms

In [55], She *et al.* propose an application-driven MAC layer buffer management framework based on active dropping for real-time video streaming. A comprehensive analytic model is formulated to determine whether a video frame can be delivered to meet its delay requirement; if not, this video frame can be actively dropped to release precious transmission resources for the subsequent frames or the frames of the other competing streams.

On the other hand, Bacioccola *et al.* [56] consider the bandwidth allocation problem with half-duplex SSs, operating in a FDD mode. The authors propose a grant allocation algorithm called the Half-Duplex Allocation algorithm. The authors investigate the performance of SSs with a half-duplex capability in terms of delay through extensive simulations.

Another interesting area of research is cross-layer optimisation. Triantafyllopoulou *et al.* [57] introduce a heuristic cross-layer mechanism that aims to improve the performance of real-time applications. This is done by adaptively controlling the modulation and data-encoding rates at the application layer, based on the information obtained from the PHY and MAC layers.

In [58], Ben-Chimol *et al.* propose a mapping and allocation representation algorithm to increase the total network utilisation, by reducing the UL mapping overheads for constant bitrate voice sessions.

2.3.6 Other Technologies

Data over cable service interface specifications (DOCSIS) networks have many similarities with 802.16 networks. A DOCSIS network is made up of a headend, called a Cable Modem Termination System (CMTS), and multiple terminating Cable Modems (CMs) [38], [59]. In a similar manner to 802.16 networks, where a BS

controls the operation of a set of SSs, the CMTS controls the operation of all connected CMs.

The shared medium between the CMTS and CMs are divided into multiple upstream and downstream channels using Frequency Division Duplexing (FDD). Each of these channels is also a shared medium and they are further divided into fixed-size time minislots. According to [38], the main access scheme to these channels in DOCSIS 1.0 and 1.1 is time division multiple access (TDMA), while DOCSIS 2.0 allows frequency division multiple access (FDMA) and synchronous code division multiple access (S-CDMA) to complement the TDMA scheme.

Similar to 802.16 networks, the DOCSIS MAC protocol also relies on a request/grant mechanism to schedule transmissions between the CMs and the CMTS. This mechanism takes place during a contention interval where collisions may happen. To avoid collisions, a simple binary exponential backoff algorithm is used. Furthermore, the CMTS sends bandwidth allocation instructions periodically to the CMs regarding their upstream transmission opportunities, which is also a feature in 802.16 networks.

To support quality of service, the concept of service flows is introduced in DOCSIS networks. A flow can be classified as UGS, rtPS, nrtPS and BE. This is the same classification defined in the 802.16 standard [1]. In the following, we discuss some of the scheduling schemes used in DOCSIS networks.

In [59], Kuo *et al.* propose a multi-level priority collision resolution scheme with adaptive contention window adjustment to improve the QoS of the DOCSIS network. The authors also propose a new service class that is claimed to experience similar QoS as that experienced by UGS traffic, but it also results in good channel utilisation, similar to that provided by rtPS traffic. Different from the others, Sdralia *et al.* [60] propose a prioritised first-come-first-served scheduling scheme to be used as a baseline reference to compare with other scheduling algorithms.

Using another approach, Ju and Liao [61] propose a new adaptive scheduling scheme called long packet deferment. This technique reduces the frequency of trans-

mission for long packets and these packets are always scheduled to be transmitted at the end of a transmission period.

In [62], Heyaime-Duverge and Prabhu propose a bandwidth allocation scheme based on the statistics of source traffic models. This scheme aims to reduce bandwidth request waiting time for a connection based on its inter-packet time distribution. Hence, this scheme claims to allow more users to be accommodated in the network.

To improve QoS, Yin *et al.* [63] propose a two-phase minislot scheduling algorithm. The two phases are the scheduling sequence determination phase and the minislot assignment phase. The scheduling sequence determination phase is used to determine the sequence estimator for each flow in the network. The sequence estimator of a flow reflects the importance of the requested minislots by this flow, relative to the other flows in the network. Hence, a flow with a smaller sequence estimator value is scheduled ahead of the other flows. In the minislot assignment phase, another estimator called assignment estimator is used to determine the minislot allocation between accepted flows. Each flow attempts to use the minislots least requested by the other flows.

Another technology that involves sharing of time-slots in upstream channels controlled by a central node are the passive optical networks (PON) [64], [65] that utilise time division multiplexing (TDM) as their media access algorithm. The other possible media access technologies are wavelength division multiplexing (WDM) and code division multiplexing (CDM).

PON is made up of a central unit called an optical line terminal (OLT). The OLT resides in a local exchange, connecting optical access networks to the Internet backbone. In a similar manner to the BS of 802.16 and the CMTS of DOCSIS networks, an OLT is responsible for scheduling transmissions of upstream traffic from optical network units (ONUs). However, there are no service classes defined in the PON standard.

In [66], Choi and Huh propose 3 priority classes for PON networks: high, medium

and low priority classes. Furthermore, a strict priority scheme is employed, where the high priority class traffic is always served ahead of the medium priority class traffic, followed by the low priority class traffic. The authors also present the control message formats required to handle the priority classes proposed.

Similar to [66], Luo and Ansari [67] propose 3 service classes, according to the Differentiated Services classification defined by Blake *et al.* in [68]: expedited forwarding, assured forwarding and best effort. The authors also propose to apply priority queueing to each of these service classes. For the upstream bandwidth request mechanism, the authors employ traffic prediction to reduce delay experienced by each transmission frame. Using an alternative approach, Assi *et al.* [69] also present a priority-based scheduling scheme with minimum bandwidth guarantee to each service class. A weight is also assigned to each ONU according to the network requirements.

In [70], Kim and Park propose short-term deficit round robin (SQ-DRR) for PON networks, which is a modified version of deficit round robin with varying quantum. The quantum of SQ-DRR for delay sensitive traffic is updated based on a short history of traffic arrival behaviour and bandwidth requirements. The authors claim that their proposed scheduling scheme is able to guarantee delay constraints for short-term aggregate traffic bursts while maintaining long term fairness.

All the scheduling techniques discussed in DOCSIS and PON networks may be applied to 802.16 networks. However, to the best of our knowledge, none of the related work in DOCSIS or PON networks can achieve the goals of the DQ discipline, which maximises the number of users that experience good service. Hence, the DQ discipline may also be applied to the DOCSIS and PON networks to add another layer of QoS control.

2.4 Motivation and Gap Analysis

There are 3 criteria identified as important for an 802.16 system scheduler in [71]: CoS differentiation, dynamic sub-frame partition and SS differentiation. In this thesis, we add another criterion to the list, which is Adaptive Modulation and Coding support. These criteria are described below.

- **CoS differentiation:** The 802.16e standard [13] describes 5 CoSs to enable differentiation between traffic with different QoS requirements. In the literature, several scheduler schemes have been proposed to achieve CoS differentiation, such as the WFQ, WRR, DRR and EDF algorithms. Furthermore, some authors propose hybrid schemes based on these scheduling algorithms

- **Dynamic sub-frame partition:** In this thesis, we contend that dynamic allocation of available bandwidth for the DL and the UL is important. According to Chen *et al.*, this is only achievable for 802.16 systems that operate in the ATDD mode [36]. In ATDD, the size of the DL and the UL sub-frames can be varied on a per MAC frame basis to achieve a better system performance. For instance, when there is more high priority traffic in the UL direction, the UL sub-frame can be increased accordingly. In order to support dynamic sub-frame partition, the DL and the UL scheduling processes must be carried out jointly. However, most of the current work has overlooked this aspect of the scheduler.

- **SS differentiation:** Since we consider a centralised 802.16 system, it is easier to provide SS differentiation than in a distributed 802.16 system. To the best of our knowledge, there is no proposed scheduler which has considered SS differentiation. There are significant potential advantages in supporting SS differentiation to provide different priorities of SSs. For instance, a commercial service provider may wish to provide better service to SSs who are willing to pay more for access. In all earlier work, the proposed 802.16 systems only

provide air-time fairness, that is, all SSs are treated equally in some sense. When SS differentiation is provided, the 802.16 systems must also consider the SS factor, as well as the CoS factor, in dynamically partitioning the sub-frames.

- **Adaptive Modulation and Coding:** One of the main features of the 802.16 systems is AMC that provides multi-rate functionalities of the systems. With AMC, it becomes harder to manage network parameters that can change dynamically, such as network utilisation, achievable throughput and fairness. Let us consider an example where there are two connections that compete for bandwidth. Suppose that one of these connections has a higher PHY mode than the other. The connection with a higher PHY mode is able to accommodate more payload in an allocated data slot. In other words, more data can be transmitted in a slot for the connection with a higher PHY mode. Without considering AMC, a scheduler may allocate the same number of slots to each of these connections, giving the connection with a higher PHY mode a chance to transmit more data. Hence, it may achieve a higher throughput, although this may be considered to be unfair. Therefore, the QoS of the network can be affected if the scheduler does not consider AMC in its scheduling decision.

We have categorised the related work described in Section 2.3 according to their support for the aforementioned four criteria. A summary of the findings is shown in Table 2.4.1, where a tick indicates that the relevant criterion is explicitly incorporated in the schemes analysed in the referenced work.

From Table 2.4.1, it is clear that the focus of the related work is on providing CoS differentiation, as they have over-looked the importance of dynamically partitioning the DL and the UL sub-frames and AMC.

In [28], the authors propose a simple algorithm for partitioning the sub-frames based on aggregate traffic conditions, without considering the different CoSs. On the other hand, the authors in [36] propose a dynamic UL and DL partitioning

Table 2.4.1: Comparison of the related work in 802.16 literature.

Paper/ Criterion	CoS diff	Dynamic sub-frame	SS diff	AMC
[25], [26], [29]–[33], [35], [38] [39], [41]–[43], [50]	✓	×	×	×
[45]–[49]	✓	×	×	✓
[28], [36], [40]	✓	✓	×	×

scheme which gives higher priority to the DL sub-frame. Giving higher priority to the DL sub-frame may not be appropriate in certain cases, especially when the system is fully loaded. In [40], the authors propose an optimisation-based scheduling scheme, in which they maximise utility functions of different CoSs. This scheme is carried out for the DL and the UL jointly, and hence dynamic sub-frame partition is achieved. Note that none of these three papers that support dynamic sub-frame partition consider AMC.

None of the related work has considered SS differentiation. There are many possible methods for implementing SS differentiation and in the next chapter, we propose a general scheme to support SS differentiation and explain the factors that may influence the scheme selection.

2.5 MAC Layer Scheduling

In this section, we discuss the various MAC layer components that we consider to be important for DL and UL scheduling.

2.5.1 Connection-Oriented Scheduling

A SS has at least one unique 48-bit MAC address (more than one MAC address is needed for multiple radios to support multihop or multichannel communication) [16]. However, the MAC address is used for initial registration and authentication only,

but not for scheduling transmissions in the 802.16 systems. Hence, the MAC address is not carried in every MPDU.

In order to reduce overheads and provide an easy way to differentiate traffic, the IEEE 802.16 standard makes use of the 16-bit CID for scheduling data, management and broadcast transmission. Thus, every connection in the network has a unique CID which is used to identify and differentiate traffic for QoS provisioning. All higher layer services, connection-oriented or connectionless, are mapped over to an IEEE 802.16 connection. This is carried out in the convergence sublayers by a one-to-one or a multiple-to-one mapping process. For example, one or more IP layer flows can be mapped to a unique CID at the 802.16 MAC layer.

The scheduling algorithms of different traffic types in 802.16 system is intentionally left outside the scope of the IEEE 802.16 standard. The standard only outlines the framework for scheduling traffic and hence it is up to the vendors to provide efficient and robust schedulers for both DL and UL.

2.5.2 Bandwidth Request

In order to carry out the UL scheduling in the BS, the SSs send UL information to the BS through the Bandwidth Request (BR) mechanism on a per connection basis.

There are two types of bandwidth request information: incremental and aggregate. When a BS receives an incremental bandwidth request from a connection, it adds the amount of bandwidth requested to its current perception of the bandwidth required for that connection. When an aggregate bandwidth request is received, the BS resets its perception of the bandwidth required for a connection.

There are also two methods that allow a connection to send bandwidth request information to the BS: unicast polling and broadcast polling.

Unicast Polling

There are two primary options for unicast polling: the BS allows a SS to piggyback the additional 2 bytes bandwidth request information as a grant management sub-header or the BS may allow a SS to send a stand-alone 6 bytes bandwidth request header to the BS. Both are normally carried out on a periodic basis. Unicast polling is used for rtPS and nrtPS traffic in order to minimise delay and jitter due to the BR mechanism.

Broadcast Polling

The BS also provides broadcast polling for SSs to send bandwidth request information during the contention periods in the UL sub-frame. The disadvantage of broadcast polling is that a collision can occur when two connections try to occupy the same contention slot. In order to deal with collisions, an exponential backoff algorithm is deployed.

2.5.3 Bandwidth Grant

IEEE 802.16 specifies two bandwidth granting modes for UL connections [20]: grant per connection and grant per SS.

Grant per Connection

In GPC, the BS treats each connection separately and grants bandwidth on a per connection basis. This mode simplifies the scheduler at the SSs in which the SSs only have to transmit each connection as ordered by the BS. This option incurs more overheads to the network and hence it is normally not preferred, unless simplicity in the SS implementation is the first priority.

Grant per Subscriber Station

In GPSS, the BS aggregates the bandwidth requests from a single SS and grants bandwidth on a per SS basis. Therefore, a scheduler is needed at the SS to determine the bandwidth allocations for each connection, given a collective bandwidth grant from the BS. The SS scheduler is expected to follow the bandwidth granting policy in the BS. This mode reduces the overhead required for sending individual connection grants to the SSs.

2.5.4 DL-MAP and UL-MAP

The DL-MAP and UL-MAP are used to describe the DL and UL bandwidth allocations and usage for all SSs. These maps store scheduling and PHY information for DL and UL communication between the BS and the SSs, and they are sent to all SSs in the network on a per MAC frame basis during the start of a MAC frame.

In TDD, the DL-MAP is always used to describe DL allocations for the current MAC frame. In the case of UL-MAP, it can be used to describe UL allocations for the current MAC frame or for the next MAC frame if the SSs do not have enough time to decode the UL-MAP for transmission in the current MAC frame.

Hence, the scheduling decisions made by the BS are stored in the DL-MAP and the UL-MAP. The responsibility of the SSs is to decode these maps and act accordingly. Therefore, these maps play an important role in QoS provisioning.

2.5.5 Adaptive Time Division Duplexing MAC Frame

In this thesis, we consider 802.16 systems that operate in ATDD mode. In ATDD, a MAC frame is divided into the DL and the UL sub-frames, where the DL sub-frame is scheduled prior to the UL sub-frame. The ratio of the DL/UL sub-frames can change dynamically depending on the traffic environment and some other system parameters.

A simplified diagram of a MAC frame in ATDD mode is as shown in Figure

2.5.1. For illustration purposes, the MAC frame is not drawn to scale. There are safe guarding periods called TTG (transmit-receive transition gap) and RTG (receive-transmit transition gap) between the DL and the UL sub-frames as shown in Figure 2.5.1.

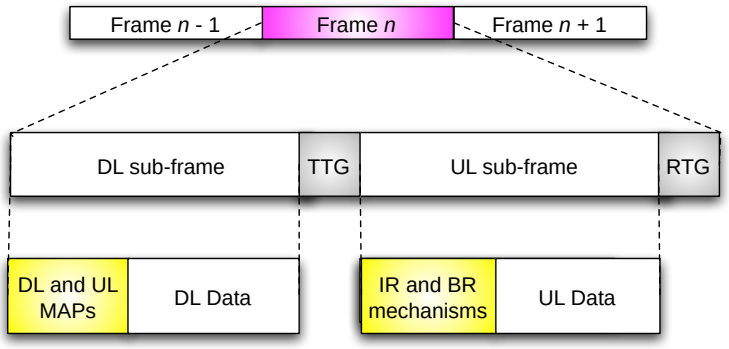

Figure 2.5.1: Structure of a MAC frame in ATDD mode.

At the start of the DL sub-frame, some time is allocated for scheduling, creating the DL-MAP and UL-MAP and sending these MAPs to all the SSs. Similarly, at the start of the UL sub-frame, some time is reserved for the initial ranging (IR) mechanism, which is responsible for all the initialisation procedures when a new SS joins the network, and the BR mechanism. The time duration allocated for the bandwidth request mechanism is further divided into multiple contention slots. The data transmission time duration in the DL and the UL sub-frames is divided into smaller physical slots for actual transmissions.

2.5.6 Equivalence of MAC Scheduling and Slot Allocation

We assume that the duration of a MAC frame is of fixed length, say D_{mac}, and the fixed overheads (which include DL and UL overheads and the guarding times) take up a portion, H, of a MAC frame. The total number of slots available for data transmission in a MAC frame, M, is given by

$$M = \frac{D_{mac} - H \times D_{mac}}{p},$$

where, p is the duration of a physical slot.

The main task of scheduling in 802.16 systems is to assign slots to each connection in the system, by the BS, on a per MAC frame basis. The output of the scheduling process will be broadcast to all the SSs in the form of the DL-MAP and UL-MAP. These maps contain information detailing what transmission should be made in each of the slots in the current or next MAC frame.

2.5.7 Base Station and Subscriber Station Schedulers

In the BS there are two schedulers, which are responsible for scheduling data connections for DL and UL transmissions respectively. These schedulers carry out their scheduling process at the beginning of every MAC frame, before any of the DL or the UL data transmissions in that MAC frame.

For the DL scheduler, data connections can be scheduled easily, since the BS has complete information required for the scheduling process. The UL scheduler, however, needs to obtain the UL information from SSs in order to carry out the UL scheduling process.

In the case of the SSs, a scheduler is needed if the bandwidth grant mechanism in use is GPSS. A SS scheduler is expected to allocate bandwidth to its UL connections based on the same policy used by the UL scheduler at the BS in granting an aggregate bandwidth to the SS. On the other hand, a scheduler at the SS is not required if the bandwidth grant mechanism in use is GPC.

2.6 Summary

In this chapter, we have provided a brief description of the different extensions of the IEEE 802.16 standard. We have also briefly described the PHY and the MAC specifications of the standard.

In 802.16 systems, it is the responsibility of the MAC scheduler to provide QoS to the systems. In this chapter, we have discussed the five CoSs defined in the 802.16e

standard [13]. Each of these CoSs has different QoS requirements. However, the actual scheduling algorithms for providing QoS to these CoSs are not specified in the 802.16 standard.

There are altogether four criteria that we have identified as important for an 802.16 MAC scheduler: CoS differentiation, dynamic sub-frame partition, SS differentiation and the ability to take advantage of AMC. We have examined related work focussing on their support for these criteria and found that the predominant theme in the related work is that of providing CoS differentiation. To the best of our knowledge, none of the earlier work has considered SS differentiation, which we will address in conjunction with dynamic sub-frame partition and AMC support.

Further, we have highlighted the key components of the PHY and MAC layers that affect the MAC layer scheduling process. We have also discussed various aspects of the MAC layer scheduling process, such as the concept of CID, bandwidth request/grant mechanisms and the creation of the maps (DL-MAP and the UL-MAP) for describing the bandwidth allocation and usage of the network. Based on the MAC and PHY features of the 802.16 systems, we have described the MAC scheduling process as a process to allocate slots to all connections in the systems.

Chapter 3

MAC Layer Scheduling: A Network Manager's Decision

In the previous chapter, we discussed various aspects of the IEEE 802.16 MAC layer scheduler design. It is clear that there are many ways to design a scheduler to meet various QoS requirements for all types of traffic. In this chapter, we examine the MAC layer scheduler design problem, or more specifically, the question of how transmission opportunities should be assigned beyond traditional concepts of QoS support.

3.1 Introduction

We describe the scheduler design problem from a general perspective, rather than with a specific design goal in mind. That is, rather than proposing a specific algorithm, we discuss alternative objectives that schedulers may try to achieve. Subsequently, we discuss an associated optimisation problem that needs to be solved in order to achieve those objectives.

Traditional approaches to 802.16 scheduler design focus on CoS differentiation. A lot of schemes have been proposed to meet various QoS requirements of all the CoSs defined in the standard, and there is also some work available on dynamic

allocation of the downlink and the uplink sub-frames. Note that transmission from the base station to the subscriber stations is called the DL, while transmission from the subscriber stations to the base station is called the UL.

In this chapter, we include another key parameter, SS differentiation, into our scheduler design. This adds an extra feature for the network provider to target a certain group of SSs or customers. To the best of our knowledge, we are the first to formulate SS differentiation into the 802.16 scheduler design. In our approach, SS differentiation can play a significant role in the bandwidth allocation process. Based on the priority assigned to the SSs, some may have a higher transmission priority than the others. When SS differentiation is deployed, it provides an option to enhance the priority assigned to the different CoSs. In the bandwidth allocation process, the scheduler may allocate more bandwidth to the SSs with a higher priority. Hence, SS differentiation is another feature that should be included in the scheduler.

In [72], [73], Moraes and Maciel propose a non-standard compliant 802.16 MAC protocol with priorities assigned to messages and/or SSs. This priority policy is utilised to determine the order of data transmission during the transmission period. The authors demonstrate the different delays experienced by the SSs by changing the SSs' transmission ordering. However, the work in both [72] and [73] does not utilise the priority policy to determine the bandwidth allocation and usage for the SSs. They assume that the SSs' bandwidth allocation for a certain transmission period is given. Thus, [72], [73] only investigate the effect of the transmission ordering on the delay experienced by the SSs, where the transmission ordering is based on the SSs priority. Therefore, the work in both [72], [73], can not be considered as true SS differentiation in our context.

By using our approach, we provide the network manager with the flexibility to configure their own system to suit their needs. For example, for a network manager that has decided to build a system that treats all users equally, fairness is an important factor in their scheduler design. On the other hand, another network manager interested in maximising revenue may choose to provide SS differentiation

and thus trade away fairness amongst subscribers.

We note that Niyato and Hossain [40] propose a queuing-theoretic and optimisation-based model for bandwidth allocation in 802.16 systems. However, the authors overlook the importance of adaptive modulation and coding in their proposal and SS differentiation is not considered. Hence, we contend that our approach is comprehensive, as it covers the requirements of both the network provider and the end users, including support for AMC and SS differentiation.

In the real world, the scheduler may have multiple objectives. We consider these objectives in a hierarchical order in our framework, in order to provide a consistent treatment of CoS, DL/UL, SS and AMC. This allows us to fulfill the multiple goals of the network provider, as well as the needs of the customers. In the related work described in Chapter 2, various authors claim that their approach is better than others in achieving specific goals. However, their approaches are limited in scope and presuppose particular objectives, or more explicitly ignore the possibility of alternative objectives. For instance, none of the related work described in Chapter 2 examines the revenue performance of the whole system in their scheduler design.

Our framework further extends all previous work through its inclusion of SS differentiation. The framework is driven by a set of objectives, selected by the network provider, but based on both the needs of the network provider and the customers.

3.2 802.16 Scheduler Design Framework

The scheduler determines the allocation of available bandwidth to connections in a network. This process affects the QoS perceived by the end users and the overall performance of the network. In this section, we propose a framework for designing the 802.16 scheduler, which is flexible and comprehensive, as it considers the requirements and goals of both the end users and the network provider.

In Chapter 2, we identified four requirements of a good scheduler: CoS differen-

tiation, dynamic sub-frame partition, SS differentiation and support for AMC. We argue that an 802.16 MAC scheduler should meet all of these four requirements. None of the existing approaches consider these four requirements together, and to the best of our knowledge, we are the first to consider SS differentiation in the scheduler, which we describe in detail below.

We treat the scheduling process as an optimisation process, which also meets the four requirements of a good scheduler. This approach is more flexible than the existing approaches as it is able to consider the possibility of alternative objectives in the scheduler.

3.2.1 Subscriber Station Differentiation

In our scheduler design framework, we provide an option to include SS differentiation in the scheduling process. Explicitly including the SS as an input to the scheduling decision making process enables network providers to offer differentiated service levels to their customers. This provides a mechanism for targeting certain types of customers.

In the commercial world, SS differentiation is beneficial in deploying differentiated charging schemes to customers. As a simple example, customers can be categorised as gold/silver/bronze customers. The gold customers are those who pay more for their service, and so, they should have higher transmission opportunities during congested periods. The silver customers are those who pay the normal rate, hence, they have less transmission opportunities than the gold customers during congested periods. Lastly, the bronze customers pay the lowest price for their service (they could be thought as the best-effort customers), and hence, they simply get whatever is left during congested periods.

Alternatively, SS differentiation can be utilised to differentiate between customers who have a service level agreement (SLA) with the network provider and those who do not. An SLA is a contract between a customer and a service provider.

It states the provision of a certain level of QoS for the service from the service provider, and in return, the customer agrees to pay (can be in the form of incentive) the service provider for satisfying these QoS guarantees.

By including SS differentiation, our scheduler design framework is comprehensive. In certain cases, there is a need for the network provider to give superior transmission priority to traffic from a particular set of SSs. Clearly, when SS differentiation is deployed, all users are not treated equally or fairly.

3.2.2 A General Optimisation Problem

The way that a base station allocates slots can be treated as an optimisation problem. The BS scheduler takes both the BS's and the SSs' queue information as inputs, and then allocates bandwidth to connections in order to achieve a desired outcome defined as an objective, which can be formulated into an objective function.

The allocation consists of the set of CoS, DL/UL and SS values assigned to each of the available slots. The allocation is represented in the form of the DL-MAP and UL-MAP as described in Chapter 2. These maps are transmitted to all SSs and used to describe the bandwidth allocation and usage on the DL and the UL respectively. This is shown pictorially in Figure 3.2.1.

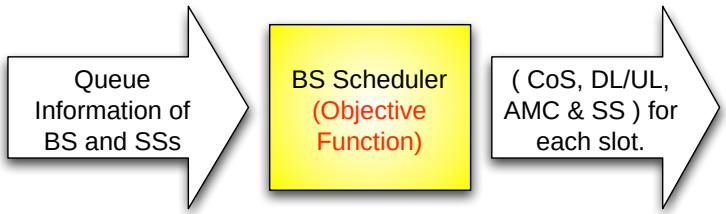

Figure 3.2.1: Scheduling process of a base station scheduler.

The specific required input for the BS scheduler is the queue information from both the BS and the SSs, in the form of the number of bytes requested for each CoS, for each DL/UL and for each SS. This information is gathered by the BS either by

direct access to the queues in the BS, or by the bandwidth request mechanism for information from the SSs.

The objective defines the goals of the system in terms of the management priorities. The objective of the BS scheduler can be structured to achieve a variety of optimisation goals, such as maximising system throughput, or the throughput of a particular group of SSs, maintaining fairness among SSs or maximising revenue, just to name a few. All scheduling decisions made by the BS will take into account CoS (as done by all proposed schedulers), DL/UL, support for AMC (as suggested in some earlier work) and the SSs themselves (not previously proposed, to our knowledge), in order to maximise this objective.

Including SS as a differentiator for MAC layer access enables, for example, a charging policy to determine the priority of a SS. Therefore, achieving fairness between SSs is no longer a fundamental "philosophy" of the system. Furthermore, support for AMC may affect the fairness, throughput and revenue of a network. A flexible scheduling mechanism will enable the service provider to determine the major criteria of interest, such as their definition of a reasonable blend of fairness, revenue or throughput. However, fairness among CoSs and DL/UL can still be considered.

3.3 Potential Objective Functions

We first define the notation to be used in the remainder of this chapter and describe the basic system constraints. Subsequently, we discuss some examples based on the framework that we have formulated. Our approach here is to use a series of examples to illustrate the different objectives that may be achieved using a generalised MAC scheduler, and where possible, explicitly formulate the corresponding optimisation problem. Via these examples we demonstrate the rich array of possibilities our generalised framework can support.

Notation

The terms utilised in this section are listed below:

S_{mac} (slots):	total number of data transmission slots in a MAC frame.
s_{ijk} (slots):	number of allocated slots for CoS i, SS j and transmission direction k (i.e. either DL or UL) in a MAC frame.
x_{ijk} (slots):	number of requested slots for CoS i, SS j and transmission direction k in a MAC frame.
T_{ijk} (bytes/sec):	throughput of CoS i, SS j and transmission direction k in a MAC frame.
r_j (bytes/slot):	number of bytes transmitted per slot for SS j according to AMC schemes.
p (sec):	slot length.
N_{CoS}:	number of CoSs (default value of 5).
N_{SS}:	number of SSs.
N_{DU}:	number of transmission directions (default value of 2; implying DL and UL).

Constraints and Throughput

We are interested in defining a general optimisation problem aimed at achieving the goals of the service provider in defining a MAC scheduler. However, there are a number of constraints imposed by the system itself. In particular, we enforce the obvious constraints on the total number of slots in a MAC frame,

$$\sum_{i=1}^{N_{CoS}} \sum_{j=1}^{N_{SS}} \sum_{k=1}^{N_{DU}} s_{ijk} \leq S_{mac}, \tag{3.3.1}$$

and the bounds on individual slot allocations,

$$s_{ijk} \leq x_{ijk}, \qquad \forall\ i, j, k. \tag{3.3.2}$$

The 802.16 physical layer supports different modulation schemes (PHY mode), which correspond to the different physical transmission rates. This provides the multi-rate feature in 802.16 systems. The amount of data that can be transmitted in a slot depends on the PHY mode of a SS, and r_j is defined as the number of bytes transmitted in a slot for SS j. Therefore, the throughput of the connection i, j, k is

$$T_{ijk} = \frac{r_j s_{ijk}}{pS_{mac}}. \tag{3.3.3}$$

3.3.1 Test Example

In order to demonstrate the flexibilities of our scheduler design framework, we consider a simple 802.16 example. In the example, there are 4 SSs connected to a BS, as shown in Figure 3.3.1. We assume that the 802.16 network is operating in point-to-multipoint mode, using adaptive time division duplexing. Each of these SSs carries a traffic connection on the DL.

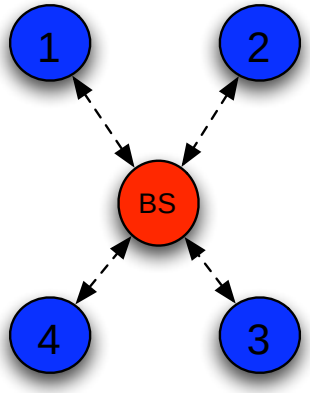

Figure 3.3.1: A simple 802.16 network with 4 SSs connected to a BS.

The SSs have different PHY modes; two of the SSs ($SS1$ and $SS3$) have high (H) PHY mode, and the other two ($SS2$ and $SS4$) have low (L) PHY mode. For simplicity, the amount of data that can be transmitted in a slot (that is, the r_j value) for high and low PHY modes is assumed to be 200 and 100 bytes per slot respectively,

thus $r_1 = r_3 = 200$ and $r_2 = r_4 = 100$. We also introduce SS differentiation in this example, where $SS1$ and $SS2$ belong to the gold class, and $SS3$ and $SS4$ belong to the silver class. The gold class SSs pay a higher rate for transmission, which corresponds to generating a revenue of 3 units per byte of data transmitted for the network provider. Whilst the silver class SSs generate a revenue of only 1 unit per byte of data transmitted. Hence, the revenue per slot is calculated as the product of r_j and revenue per byte, which is shown in Table 3.3.1. In addition, the revenue per MAC frame from a SS can be calculated as the product of revenue per slot and the actual number of slots allocated to the SS

Table 3.3.1: The number of slots and data rate required by each connection in our test example.

Connection	r_j	Revenue per byte	Revenue per slot	Slots required	Data rate required
$SS1$(gold)	200(H)	3	600	4	800
$SS2$(gold)	100(L)	3	300	4	400
$SS3$(silver)	200(H)	1	200	4	800
$SS4$(silver)	100(L)	1	100	4	400

In this example, the total number of available slots is 12 and the number of slots required for each connection is 4 slots per MAC frame. Hence, the data rate of each connection, in term of bytes per MAC frame can be calculated as the product of the number of slots required and r_j. Given the total number of available slots of 12 and the total number of requested slots per MAC frame from the SSs of 16 (4 from each SS), the objective defined for the system determines the allocation of the available slots to the SSs.

3.3.2 Objective Function A: Maximising Throughput

In possibly the simplest case, we seek to maximise the total throughput of the system, irrespective of CoS, DL/UL and SS. That is, we seek to maximise:

$$\sum_{i=1}^{N_{CoS}} \sum_{j=1}^{N_{SS}} \sum_{k=1}^{N_{DU}} \frac{r_j s_{ijk}}{pS_{mac}},$$

or equivalently,

$$\sum_{j=1}^{N_{SS}} r_j \{ \sum_{i=1}^{N_{CoS}} \sum_{k=1}^{N_{DU}} s_{ijk} \}. \tag{3.3.4}$$

Here, the number of available slots will first be allocated to traffic connections that have a higher PHY mode, or equivalently, higher r_j value. If the total number of slots requested from all the high PHY mode connections can be accommodated, these connections will be satisfied, as they are allocated with the number of slots that they request. The remaining slots, after serving the high PHY mode connections, will then be allocated to connections with a low PHY mode. There are many ways to distribute the remaining slots to these low PHY mode connections. This decision has no impact on the total system throughput, but it is important as it may affect other performance aspects of the system, such as revenue and fairness.

However, if the total number of slots requested by all the high PHY mode connections can not be accommodated, all the available slots will be allocated to the high PHY mode connections. In this case, the low PHY mode connections get none of the available slots, due to the limited bandwidth and the objective selected. By giving all the available slots to the high PHY mode connections, we can achieve the goal of maximising the system throughput. Similarly, there is then the issue of how the slots are distributed among the high PHY mode connections.

In Table 3.3.2, $SS1$ and $SS3$ of the test example are first allocated 4 slots each, since they have the highest PHY mode, and their total requests can be accommodated. The remaining 4 slots will then be distributed among the other connections. As mentioned earlier, we have options on distributing the remaining slots to the low

PHY mode connections, but there are some factors that may affect the way we distribute the remaining slots. For example, if the connection from $SS2$ is a real-time service connection, and $SS4$ is a non-real-time service connection, we could decide to allocate slots to a real-time service connection ahead of a non-real-time service connection as the real-time service connection has a tighter QoS requirements. If both connections from $SS2$ and $SS4$ are non-real-time service connections, the remaining slots could be distributed evenly among them. In fact, these factors can be formulated as secondary objectives. Since secondary objectives are not defined in this example, we choose to evenly distribute the slots among these SSs.

Table 3.3.2: Slot allocation to maximise throughput.

Connection	r_j	Revenue per byte	Revenue per slot	Slots allocated	Data rate allocated	Revenue per MAC frame
$SS1$ (gold)	200(H)	3	600	4	800	2400
$SS2$ (gold)	100(L)	3	300	2	200	600
$SS3$ (silver)	200(H)	1	200	4	800	800
$SS4$ (silver)	100(L)	1	100	2	200	200
				12	2000	4000

Under this objective, the total data transmitted per MAC frame is equal to 2000 bytes. This objective is clearly not fair as it gives more transmission opportunities to SSs with high PHY mode. If necessary, the fairness issue can be considered as a secondary performance aspect of the system to achieve fairness among connections with the same PHY mode.

3.3.3 Objective Function B: Max-min Air-time Fairness and Proportional Bit Fairness per SS

This approach seeks to maximise the minimum number of slots allocated to each SS. We now outline the slot allocation process. First, the total transmission request, in

terms of the number of slots for each SS, is ordered from the smallest to the largest. Note that each SS will be allocated all of its requested slots when the total number of requested slots of all connections is less than the total number of available slots.

When the total number of requested slots of all connections is greater than the total number of available slots, the scheduler attempts to allocate each SS with the smallest number of slots requested, followed by the second smallest number of slots and so on. That is, each SS will first be allocated the smallest number of slots requested if possible. Any unallocated slots are then assigned equally across SSs up to the next smallest number of slots requested, and so on. If the total number of available slots is not even enough to allocate to each SS with the smallest number of slots requested, the available slots will be distributed evenly among the SSs, regardless of their request.

This approach focuses on air-time fairness rather than throughput, but the equivalence of this objective (max-min air-time fairness) with proportional bit fairness is shown in [74]. To achieve max-min air-time fairness per SS, which is equivalent to achieving proportional bit fairness per SS [74], we seek to maximise,

$$\prod_{j=1}^{N_{SS}} \{ \frac{r_j}{pS_{mac}} \sum_{i=1}^{N_{CoS}} \sum_{k=1}^{N_{DU}} s_{ijk} \},$$

or equivalently,

$$\prod_{j=1}^{N_{SS}} \{ r_j \sum_{i=1}^{N_{CoS}} \sum_{k=1}^{N_{DU}} s_{ijk} \}. \tag{3.3.5}$$

Under fully loaded conditions, all SSs have enough data waiting for transmission, and so

$$\sum_{j=1}^{N_{SS}} \{ \sum_{i=1}^{N_{CoS}} \sum_{k=1}^{N_{DU}} s_{ijk} \} = S_{mac},$$

and therefore the maximum is achieved when,

$$\sum_{i=1}^{N_{CoS}} \sum_{k=1}^{N_{DU}} s_{ijk} = \frac{S_{mac}}{N_{SS}}, \qquad \forall j. \tag{3.3.6}$$

This implies that the air-time fair share of the number of slots allocated for each SS is equal to $\frac{S_{mac}}{N_{SS}}$. If this number of allocated slots is greater than or equal to the total number of slots requested for SS j (i.e. $\frac{S_{mac}}{N_{SS}} \geq \sum_{i=1}^{N_{CoS}} \sum_{k=1}^{N_{DU}} x_{ijk}$), we mark SS j as a "satisfied" SS, otherwise, we mark it as an "unsatisfied" SS. The unused slots from these "satisfied" SSs will then be divided proportionally among the "unsatisfied" SSs using the same method. This process is repeated as necessary.

In this example, the number of requested slots is the same for all the connections, that is, 4 slots, and so the number of available slots is not sufficient to allow each SS to be allocated with the minimum of the number of slots requested. Therefore, in order to achieve max-min air-time fairness, the number of slots available should be divided evenly amongst all the connections. The air-time fair share of the number of slots allocated for each SS equals, $\frac{S_{mac}}{N_{SS}} = \frac{12}{4}$. Hence, 3 slots are allocated to each SS. This is shown in Table 3.3.3.

Table 3.3.3: Slot allocation to maintain max-min air-time fairness and proportional bit fairness per SS.

Connection	r_j	Revenue per byte	Revenue per slot	Slots allocated	Data rate allocated	Revenue per MAC frame
$SS1$ (gold)	200(H)	3	600	3	600	1800
$SS2$ (gold)	100(L)	3	300	3	300	900
$SS3$ (silver)	200(H)	1	200	3	600	600
$SS4$ (silver)	100(L)	1	100	3	300	300
				12	1800	3600

This objective is fair to all SSs in term of sharing the number of slots available to them. Note that this objective still favours SSs with a higher PHY mode as they have a larger payload in each slot, giving them a higher achievable data rate.

3.3.4 Objective Function C: Max-min Bit Fairness

This approach focuses on the throughput of each SS, and seeks to maximise the minimum throughput achieved by each SS, regardless of SS differentiation and CoS.

The approach to achieve max-min bit fairness is quite similar to the approach to achieve max-min air-time fairness. Instead of using the number of slots allocated as the unit for distributing bandwidth, we utilise the amount of data being allocated for transmission in terms of bytes. If the total number of slots required for all SSs is less than or equal to the number of slots available, then all requests can be accommodated and each SS will be allocated the number of slots that it requested. This is straight forward and there is no fairness issue to be considered.

When the total number of slots required for all SSs is greater than the number of slots available, the scheduling process needs to consider the fairness issue as defined in the objective. In objective B, we considered the bandwidth allocation process in terms of the number of slots. However, for this objective, we have to carry out the bandwidth allocation process in two domains; the slot domain and the byte domain. We can not map everything from the slot domain into the byte domain because we can not calculate the available resources in terms of the number of bytes due to the different PHY modes of the SSs. The available resources in terms of the number of bytes is dynamic, as it depends on the number of slots allocated to each SS and hence PHY mode. Therefore, the scheduler has to switch between the slot and the byte domains to achieve max-min bit fairness among SSs.

First, the total transmission request in terms of the number of bytes for each SS is ordered from the smallest to the largest. Then, the corresponding number of slots each SS will require to achieve their desired volume of transmission is determined. Each SS will be allocated a number of slots that equals the lowest request in bytes, if this can be accommodated. Then, we subtract an amount equal to the lowest request in bytes from each SSs request, and repeat the allocation process until all slots are assigned.

In the case where not even the lowest transmission request can be fulfilled, we have to work out the number of slots for the smallest request in terms of the number of bytes that can be supported. Then, we assign this number of slots to each SS.

In this example, we seek to achieve bit fairness among the 4 SSs. The minimum data rate of the connections is 400 bytes per MAC frame, which is calculated from the low PHY mode connections. Hence, we need to work out the total number of slots required to support this data rate for all the SSs. It turns out that the total number of slots required is equal to 12, corresponding to allocating 2 slots to both connections with a higher PHY mode and 4 slots to both connections with a lower PHY mode. This is equal to the total number of slots available and therefore, the achievable data rate of each SS is equal to 400 bytes per MAC frame. Since there are no remaining slots, the scheduling process stops here, and the summary is shown in Table 3.3.4.

Table 3.3.4: Slot allocation to maintain max-min bit fairness.

Connection	r_j	Revenue per byte	Revenue per slot	Slots allocated	Data rate allocated	Revenue per MAC frame
$SS1$ (gold)	200(H)	3	600	2	400	1200
$SS2$ (gold)	100(L)	3	300	4	400	1200
$SS3$ (silver)	200(H)	1	200	2	400	400
$SS4$ (silver)	100(L)	1	100	4	400	400
				12	1600	3200

This objective is fair to all SSs in terms of achieving the same data rate, regardless of their PHY mode. This objective also gives protection to the low demand SSs as they will not be blocked from transmission opportunities even though the other SSs are demanding more bandwidth.

For this example, the numbers have been selected in order to produce ideal slot allocations. There are, however, other complex issues that in general need to be

considered. One of these issues is the rounding up or down of numbers when we carry out the mapping process between the byte and the slot domains. Therefore, it is better to consider the max-min bit fairness for SSs in a longer time period, across multiple MAC frames. By considering the scheduling process across multiple MAC frames, the rounding effects can be minimised, if not removed. One of the available scheduling algorithms that strives to achieve a similar goal is the deficit round robin algorithm [7].

3.3.5 Objective Function D: Maximising Revenue

In this objective, we include SS differentiation in our scheduling process. The gold class SSs are given transmission opportunities ahead of the silver class SSs. A SS with a higher revenue per slot, has potential to produce more revenue to the network provider. This is summarised in Table 3.3.5.

Table 3.3.5: Slot allocation to maximise revenue.

Connection	r_j	Revenue per byte	Revenue per slot	Slots allocated	Data rate allocated	Revenue per MAC frame
$SS1$ (gold)	200(H)	3	600	4	800	2400
$SS2$ (gold)	100(L)	3	300	4	400	1200
$SS3$ (silver)	200(H)	1	200	4	800	800
$SS4$ (silver)	100(L)	1	100	0	0	0
				12	2000	4400

The maximal revenue is achieved by allocating slots to SSs with the highest revenue per slot first. When sufficient allocations have been made to meet these SSs needs, we move to the SSs with the next highest revenue per slot, as long as there are slots available. If the number of slots requested for the SSs with the highest revenue per slot is greater than the number of slots available, we allocate all the available slots to the SSs with the highest revenue per slot. However, we

have options regarding how to allocate these slots among the SSs with the highest revenue per slot, as this decision will not affect the total revenue received by the network provider. Similar to objective A, this decision can be formulated in terms of secondary objectives for the system.

In Table 3.3.5, it is shown that $SS1$ to $SS4$ are in order of decreasing revenue per slot. Hence, the available slots are allocated to $SS1$ first, followed by $SS2$, $SS3$ and lastly $SS4$. As a result, 4 slots are allocated to each of the first 3 SSs, leaving none for the last SS. The total revenue received per MAC frame for this objective is 4400 units.

This objective is not fair as it gives more transmission opportunities to the traffic that generates the most revenue per slot (in this case, it is the gold class SSs). Clearly, this objective may be good for the network provider and for those who are willing to pay.

3.3.6 Discussion

These simple examples show the different possibilities for allocating slots to SSs based on the chosen objective. The performance of the selected objective is evaluated in terms of achieving total throughput and total revenue and the summary is shown in Table 3.3.6.

Table 3.3.6: Summary of total throughput and revenue under different objectives.

Objective	Total data transmitted per MAC frame	Total revenue
A	2000	4000
B	1800	3600
C	1600	3200
D	2000	4400

Among the four objectives discussed, the maximum throughput is achieved in objective A, which matches with the objective defined to maximise system through-

put. Incidentally, the total throughput achieved in objective D that aims to maximise revenue, is also equal to the maximum throughput. This is partly because the charging scheme is based on the total number of data bytes transmitted, and partly, because of the way the basic parameters were selected. On the other hand, the maximum revenue is only achieved in objective D. Note that both objectives A and D trade off fairness among SSs, in order to achieve the objective defined for the system respectively.

In objective B, max-min air-time fairness or equivalently proportional bit fairness is achieved as all the SSs are allocated with the same number of slots. In objective C, max-min bit fairness is maintained as the SSs are respectively allocated with the number of slots, which produces the same data rate. Hence, the respective fairness goals are achieved in both objectives B and C, but at the cost of throughput and revenue. If a network provider aims to maximise system throughput or revenue, they have effectively decided that fairness is not the most important performance measure.

In both objectives A and D, we find that there is a need to define secondary objectives to guarantee secondary performance aspects of a network. In the real world, it is common for the network provider to have multiple objectives in their network. Hence, we propose that these multiple objectives be considered in a hierarchical order. This is discussed in detail in Section 3.4.

In this example, we have demonstrated the rich potential of our framework. Overall, it is shown that such a framework encompasses a broad range of service provider's objectives.

3.4 Hierarchical Objectives

In Section 3.2, we consider a single objective in our framework. In fact, this is not realistic in the actual networking world and we have shown this point in Section 3.3. Therefore, we must extend our framework to consider multiple objectives. We

propose to consider these multiple objectives in a hierarchical ordering.

Firstly, the network provider has to rank their objectives. The hierarchical relationship of these multiple objectives is shown in Figure 3.4.1. The objective with the highest rank is appointed as the primary objective. The second highest ranked objective is appointed as the secondary objective and so forth.

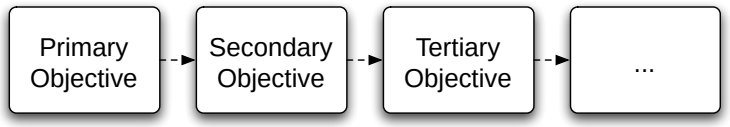

Figure 3.4.1: A hierarchy of objectives considered in our framework.

The primary objective is considered first during the scheduling process. When the primary objective is satisfied, the scheduler checks for any possibilities to include the secondary objective, and so on. By considering multiple objectives hierarchically, our framework ensures that there are no contradictions in the scheduling process.

3.5 Are the Customers Satisfied?

Providing satisfaction to the customers, or users, is an important objective of telecommunication network engineering. For a network provider, customer satisfaction is important to achieve higher customer retention [75] and increased market share [76]. Hence, we need to make sure that the objectives chosen are able to support customer satisfaction.

Customer satisfaction is, however, a subjective concept. Opinion may vary from one individual to another, and therefore, it is important to investigate various methods to achieve customer satisfaction.

For the simple example that we considered throughout Section 3.3, we think of a SS as being satisfied when it is allocated its requested number of slots for transmission, and hence its QoS requirements are satisfied. In contrast, if the number of slots allocated is less than the number of slots requested for a SS, it is very likely

that certain QoS requirements of the SS will not be met. Therefore, as shown in Figure Table 3.5.1, if a SS is allocated 4 slots, we mark it with a "✓" as a satisfied SS, and conversely with a "×" if it is not.

Table 3.5.1: Summary of the number of satisfied customers achieved in different objectives.

	Objective A		Objective B		Objective C		Objective D	
$SS1$	4	✓	3	×	2	×	4	✓
$SS2$	2	×	3	×	4	✓	4	✓
$SS3$	4	✓	3	×	2	×	4	✓
$SS4$	2	×	3	×	4	✓	0	×
Total satisfied customers	2		0		2		3	

Table 3.5.1 shows the number of satisfied SSs for each of the objectives discussed above. Both objectives A and C have the same number of satisfied SSs, that is, 2. They are, however, not referring to the same set of SSs. In objective B, all SSs are treated equally in terms of the number of slots allocated, hence they are all allocated 3 slots, and so none of them are satisfied. On the other hand, there are 3 satisfied SSs in objective D; the first three SSs are allocated 4 slots each, leaving none for the last SS.

From the customer satisfaction perspective, objective D represents a promising scheme as it offers the maximum number of satisfied customers. This leads us to another meaningful objective for our base station scheduler, which is the objective to maximise the number of satisfied customers or SSs.

To explain this new objective further, let us revisit objective A, where we aim to maximise system throughput. In objective A, slots are firstly allocated to SSs with a higher PHY mode, that is, $SS1$ and $SS3$. Each of them is allocated 4 slots and hence they are satisfied. The remaining 4 slots are then evenly allocated to $SS2$

and $SS4$. If we have a secondary objective that aims at maximising the number of satisfied customers, instead of evenly dividing the remaining 4 slots, we could allocate the entire remaining 4 slots to either $SS2$ or $SS4$, while the other gets none. By doing this, we could have 3 satisfied customers instead of 2. Hence, we are able to maximise the primary and secondary objectives defined for the system.

It might be useful to maximise the number of satisfied customers as the primary objective. If we think of an unsatisfied customer as a complaint, then maximising the number of satisfied customers is equivalent to minimising the number of complaints a network provider receives.

3.6 Summary

In this chapter, we have described the 802.16 MAC layer scheduling process as an optimisation process. Past work has focused heavily on the scheduling question, as it pertains to CoS differentiation. We, on the other hand, have approached the problem from a general perspective, in which the goal of the scheduling process is defined in terms of a set of objectives.

Our proposed framework also meets the four requirements of a good scheduler that we have identified in Chapter 2, that is, CoS differentiation, dynamic DL and UL sub-frames partition, SS differentiation and adaptive modulation and coding. The set of potential objectives that we have explicitly illustrated in this chapter is clearly only a subset of all possible objectives. If there are multiple objectives involved, these objectives are considered in a hierarchical order.

With our scheduler design framework, any requirements from both the network provider and the customers that affect the scheduling process, can be taken into consideration. Further, we have included SS differentiation into our scheduler design framework. This feature allows the network provider to target a certain group of SSs, or, customers. We have used a simple SS differentiation scheme as an example, by assigning gold or silver class to SSs, to demonstrate the importance of SS

differentiation.

On the other hand, we have demonstrated the flexibility of our framework, through examples in Section 3.3 of this chapter, that highlight how a service provider might achieve different throughput and revenue goals based on different objectives. From these examples, we have contended that fairness is not necessarily be a sensible measure if the network provider desires to maximise throughput or revenue.

Furthermore, we have discussed the issue of achieving customer satisfaction in the network and this leads us to an interesting objective, which is the objective to maximise the number of satisfied customers. In the following chapters, we investigate this objective in detail, and develop a possible algorithm to maximise the number of satisfied customers.

Chapter 4

Approaches to Customer Satisfaction

In this chapter, we first provide an introduction to customer satisfaction and outline the possible benefits of achieving customer satisfaction. We then investigate methods to achieve customer satisfaction, and contend that fairness is not a direct measure of the satisfaction as perceived by the customers. We propose to improve network performance during periods of congestion by reducing fairness in order to achieve better customer satisfaction. We outline the Dual-Queue (DQ) scheduling discipline proposed by Hayes *et al.* [2], which aims to maximise the number of satisfied customers in a wired network. The DQ algorithms in [2], however, are not applicable to the 802.16 environment. To end this chapter, we discuss issues related to employing the DQ scheduling discipline in the 802.16 environment, and provide reasons why the original DQ algorithms from Hayes' work can not be directly employed into the 802.16 environment.

4.1 What is Customer Satisfaction?

Customer satisfaction is a highly subjective and abstract concept. It is complex to define because it depends on both psychological and physical factors. The psycho-

logical factors depend on the customers, while the physical factors are related to the product. The level of satisfaction may vary from individual to individual, and product to product. Therefore, it is difficult to compare customer satisfaction across different individuals and different products. In all businesses that compete for customers, achieving customer satisfaction plays an important role in business strategy. By measuring customer satisfaction, we obtain an indication of how successfully a service is provided to the customers.

In [77], customer satisfaction is specified as a function of perceived quality and disconfirmation, where disconfirmation happens when the perceived quality fails to match the prepurchase expectations. In other words, satisfaction is more likely to be achieved when the customer's expectations are met. In [78], the author explores different sources of customer expectations and proposes a model to manage expectations. If customer expectations are managed properly, customer satisfaction can be easier to achieve.

The research on customer satisfaction has received considerable attention in various service sectors, including fast food service [79], [80], telecommunication service [81], [82], hotel service [83], [84], tourism service [85], [86] and medical service [87], [88]. These studies show that there is strong causal linkage between customer satisfaction and service performance, that is, good service performance produces customer satisfaction.

The outcomes, or consequences, of customer satisfaction have also been actively studied. There are studies that show customer satisfaction brings the following benefits: increased customer loyalty [89], [76], increased customer retention [75], improved market share [90] and increased profitability [91]. In recent work, two previously neglected outcomes have been added to the list, namely increased advertising and promotion efficiency, and increased human capital performance [92]. Customer satisfaction is therefore very important to the success of many businesses.

4.2 Achieving Customer Satisfaction

With the benefits mentioned in the previous section, there is strong motivation for achieving customer satisfaction. Furthermore, it has been shown that service performance is the major determining factor for customer satisfaction [79]–[88], and so there is a need to consider methods of improving service performance.

Service quality, or service performance, can be categorised into two critical aspects: operational and relational performance [79]. Operational performance is related to the physical aspects of the service, while relational performance is about understanding customers' needs and expectations.

From a telecommunication perspective, operational performance corresponds to network performance; while relational performance corresponds to customer service given by the service provider. According to [93], in the telecommunication industry, customers put more weight on operational performance than relational performance. That is, customers emphasize the impact of the network on their traffic rather than the quality of customer service that they receive.

Telecommunication network performance as perceived by the customers is not necessarily the same as the network performance perceived by the network provider. From the customers' perspective, the impact of the network on their traffic flows is the major factor that influences satisfaction [93]. The customers are concerned with network measures that affect their traffic flows, such as traffic throughput, delay, jitter and loss. Furthermore, opinions from different customers may vary about the performance of a network even though they are treated identically, due to the subjective opinion of the customers.

From the network provider's perspective, overall network performance measures such as the total system throughput and efficiency are important, as these measures affect the profitability of the network. A comparison of the different network measures considered by the customers and the network provider is shown in Figure 4.2.1.

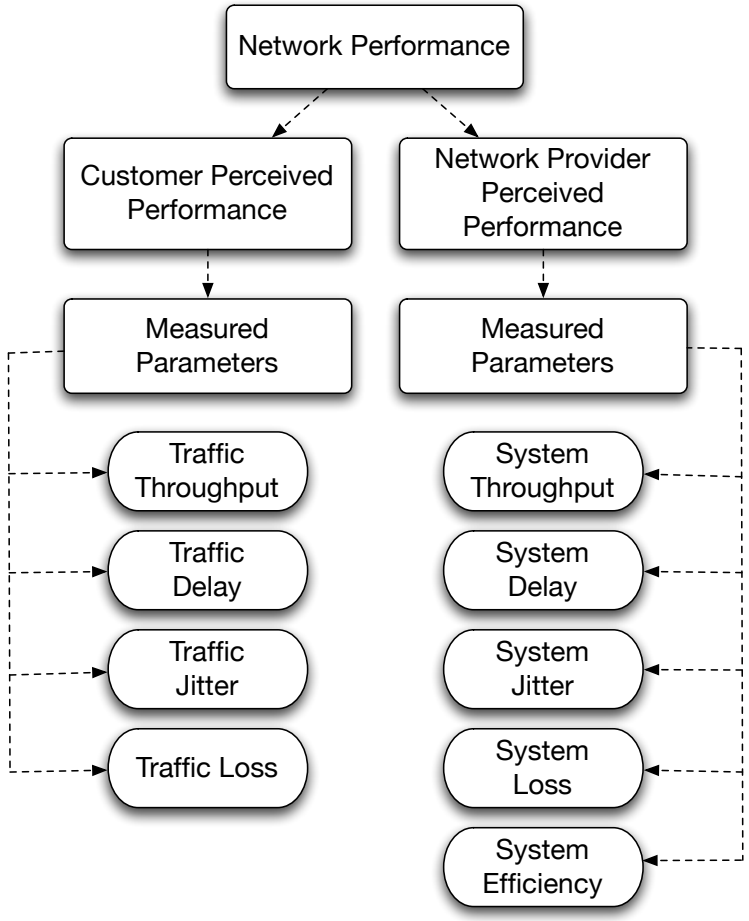

Figure 4.2.1: Network performance perceived by the customers and network provider.

In this research, we investigate methods to improve network performance, in order to achieve customer satisfaction. Improving network performance can be a difficult and complex task. One of the simplest methods to improve network performance is to add more bandwidth to a network. There will be no QoS problem in the network if the available bandwidth is always greater than the bandwidth demand. However, this is not cost-efficient. Therefore, a lot of research focuses on finding methods to make better use of the available bandwidth, especially during periods of congestion, which improves the network performance.

Some of the techniques used by a network provider for making better use of

available resources are Low Latency Queue, Random Early Detection [94], Priority Queue [95], Class-Based Queueing [96] and Differentiated Services [97]. All of these techniques have their own strengths and weaknesses, and hence are deployed depending on network conditions and application requirements.

We investigate methods for making better use of the available bandwidth in a network, rather than just adding more bandwidth, especially during periods of congestion, in order to maximise customer satisfaction.

We define the term "good service" to describe a service that meets the customer's requirements. Similarly, we define the term "degraded service" to describe a service that does not meet the customer's requirements. For example, if the customer's concern is delay and a delay of 50 ms is accepted by the customers, then we can imply that the customer receives "good service" when the traffic delay is less than or equal to 50 ms. However, there may not be a clear measure of a customer's requirements.

There is another issue which is tightly related to customer satisfaction, that of fairness. Achieving fairness among customers is common in the literature [98]–[100]. This is achieved by assigning available bandwidth to the customers fairly according to certain policies defining fairness. These policies ensure that during periods of congestion, the customers get at least their fair share of available bandwidth, which can be in terms of throughput, air-time usage and utility [101]. Further, there are different definitions of fairness itself, such as max-min, proportional [101] and fairness index [102].

However, fairness is not a direct measure of the satisfaction as perceived by the customers. In fact, the customers do not know if they are receiving fairness as promised by the service provider. Hence, fairness can not be used to measure QoS perceived by the customers because they can not "see" the fairness policy. The customers will not be satisfied if they get poor service quality during a congestion period, even though they are treated fairly. Fairness is only visible to the service provider and it is used to determine bandwidth allocation when congestion occurs.

In [2], Hayes *et al.* contend that a more relevant measure of QoS is the proportion of customers receiving "good service" at any time (including the congested and non-congested periods), because the customers who receive "degraded service" are likely to complain to their service provider. In order to minimise the number of complaints that a service provider receives, the authors develop a scheduling discipline which maximises the number of satisfied customers.

The scheduling discipline proposed by Hayes *et al.* [2] is called the Dual-Queue (DQ) discipline. The DQ discipline aims to maximise the number of customers receiving "good service", rather than maintaining fairness among the customers. This is achieved by selecting certain customers to suffer "degraded service" during periods of congestion.

The DQ discipline can be described using the "Titanic analogy", in that the number of life-boats was not sufficient to carry all the passengers on board the Titanic [103]. The captain of the Titanic therefore needed to make a decision on how to allocate passengers to the life-boats, given that the Titanic was sinking. This decision is similar to the decision of a packet scheduler during congestion episodes, during which time some packets will not be able to be scheduled. When there are too many packets waiting for transmission, it is the decision of the scheduler as to how to allocate the limited bandwidth to the packets waiting for transmission.

Should the Titanic captain be fair and try to save everyone's life by overloading the emergency boats? However, overloading the emergency boats can be catastrophic, causing more casualties. On the other hand, the captain may select certain passengers to board the emergency boats. By ensuring that the boats are not overloaded, these boats are able to bring the selected passengers to safety. This choice sacrifices some passengers in order to save the lives of the others. The DQ discipline is similar to the latter choice, choosing to sacrifice some customers for the benefit of others.

4.3 Dual-Queue Scheduler in Wired Environment

4.3.1 Background

In 1999, Hayes *et al.* [2] proposed the DQ discipline that aims at maximising the number of customers receiving good service. The DQ discipline provides a different approach for achieving QoS for customers as it ignores, or cares less about, fairness among customers. With this in mind, the discipline provides the option to sacrifice the QoS of certain customers to maintain the QoS requirements of the others. The DQ discipline can be applied to schedule real-time services such as Voice over IP (VoIP) and video streaming, which are delay sensitive.

When a system is operating under uncongested conditions, the QoS requirements of all flows can always be met. However, during congestion episodes, the system is unable to meet the QoS requirements of all the flows. Hence, some or all flows will experience degraded service. In [2], the QoS requirements considered are packet delay and packet loss rate.

If congestion is unavoidable, one of the ways to minimise the impact of congestion on the system is to confine the spread of the impact to as small a number of flows as possible. This is achieved by selecting certain flows to suffer the impact of congestion, or degraded service, while giving the others good service. The decision on which flow to select can be based on the priority assigned to each flow. In other words, the flows with lower priorities should be selected to suffer the impact of congestion, while giving the flows with the highest priority good service. Hence, the flows that experience good service do not see the congestion event. Furthermore, the number of selected flows that experience "degraded service" should be minimised, or equivalently, the number of good service flows should be maximised during congestion episodes.

The only work in the literature that extends the DQ discipline to wireless environments is by Ranasinghe et al. [104]. There are two major contributions from their work. Firstly, the authors propose a novel scheduling scheme, called Embedded

Round Robin (ERR) to generate poll requests with varying inter-poll time, in which the stations with data to transmit (defined as the "busy" stations) are polled more often than the others. The authors contend that this approach reduces the polling overhead and consequently increases the bandwidth available for data transmission. Secondly, based on the ERR algorithm, the authors investigate the extension of the DQ discipline in the uplink (transmission from mobile stations to access point) of the IEEE 802.11 network. The increase in the polling latency of the busy stations is used as an indication of congestion in the proposed DQ system.

The authors assume that the size of the busy station list would gradually grow when congestion occurs. We contend that this assumption is not valid if the congestion is not due to the increase in the number of stations transmitting data, but rather due to an increase in the traffic load offered by the existing busy stations. Hence, congestion can occur despite the fact that there is no increase in the number of busy stations. Similarly, when congestion abates which is due to the reduction in transmission rate of each busy station, the number of busy stations could remain unchanged. Therefore, we contend that the number of busy stations can not be used for detecting congestion.

Other DQ related work by Bedford and Zeephongsekul [105], [106] introduces the Multi Priority Dual Queue (MPDQ) with a preemptive priority scheme. After writing down the states of the system, the authors provide a new recursive algorithm for solving the linear systems of equations governing the steady-state probabilities. The authors also give a detailed waiting time analysis of the MPDQ.

4.3.2 The Original Dual-Queue Concept

In the following, we explain the original DQ concept proposed by Hayes *et al.* in [2].

Queue Structure

The DQ approach makes use of two queues (as shown in Figure 4.3.1); one for buffering flows that experience "good service" and the other for buffering flows that experience "degraded service". These are labelled as the α-queue and the β-queue respectively. Under uncongested conditions, all packets are sent to the α-queue only. When congestion occurs, the β-queue will be used to store packets from flows that have been selected explicitly to experience degraded QoS. Under all conditions, any transmitted packets only leave the system through the α-queue. When congestion abates, one or more flows at the β-queue will be redirected back to the α-queue.

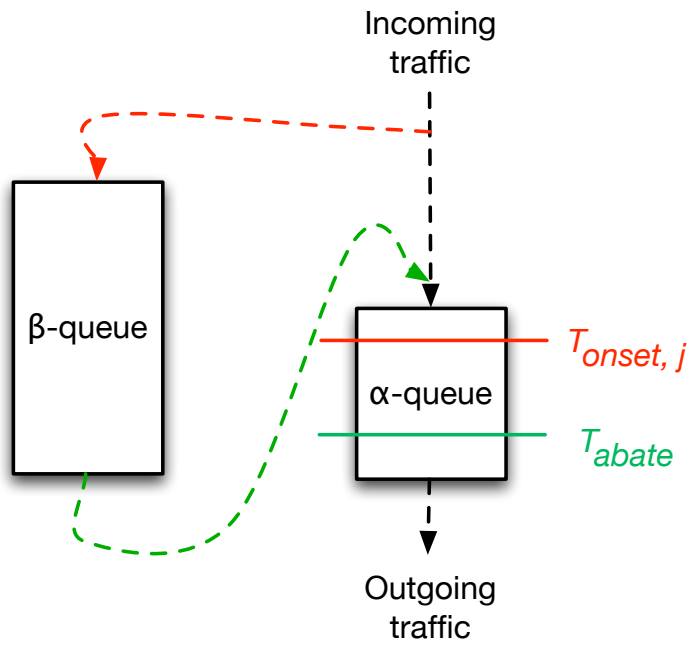

Figure 4.3.1: The original Dual-Queue scheduling discipline.

In [2], the authors utilise a short FIFO queue to represent the α-queue. As the QoS requirement considered by the authors is delay, the α-queue aims to meet the delay requirement (D_{req}) of the packets served in this queue. In a wired network, the outgoing transmission rate would normally be constant and hence the length of the α-queue can be used to determine the maximum delay experienced by the packets

at any time. If the α-queue was to overflow, it would indicate that the network is congested. Therefore, D_{req} is used to determine the size of the α-queue.

In the case of the β-queue, the authors propose various queue structures, depending on the selection criteria for moving flows from the β-queue back to the α-queue. For instance, the length of the β-queue may be significantly larger than the length of the α-queue, and this is determined by the packet loss requirements of the system. In other words, the length of the β-queue is dependent on the difference between the maximum amount of traffic that is allowed in the system and the amount of traffic that can be handled by the α-queue.

Note that the DQ concept can operate along with any existing scheduling schemes employed in a network. Also, the DQ concept could be turned off by using the α-queue only.

Service Sequence

Under any conditions, the DQ scheduler only serves packets through the α-queue. Flows that are in the β-queue would only be served after being redirected back to the α-queue by a mechanism that indicates congestion has abated. However, it is not necessary that all the flows in the β-queue will be redirected back to the α-queue at the same time.

In [2], the number of flows at the β-queue to be redirected back to the α-queue at one time is left open. In fact, we show in the next chapter that it is preferred to redirect a single flow in the β-queue back to the α-queue at any time, in order to maintain the QoS of the flows in the α-queue.

Congestion Detection

Since the main goal of the DQ discipline is to maximise the number of good service flows in the network, the DQ scheduler strives to accommodate as many flows in the α-queue as possible, without overflowing the α-queue.

When a packet arrives to the system, the packet is checked to see whether it

should be placed in the α-queue or β-queue. During uncongested operation, all packets are placed in the α-queue, hence the β-queue remains empty. The α-queue must not overflow under any operational conditions. This feature is guaranteed by the set of thresholds, $T_{onset,j}$, $j > 0$, in the α-queue. Note that these thresholds are described in terms of the number of packets. If the queue length of the α-queue exceeds the first threshold ($T_{onset,1}$) when a packet arrives to the system, it indicates an increase in delay experienced by the flows in the network, as well as the possibility that the α-queue will overflow, due to a congestion episode in the network. When $T_{onset,1}$ is crossed, the scheduler moves a single flow into the β-queue, that is any current packets in the α-queue for this flow, as well as any subsequent packets from this flow will be placed in the β-queue. This reduces the traffic load served by the α-queue. If this reduced traffic load can be handled by the α-queue, we expect to see the queue length of the α-queue to be maintained, if not decreased. If the reduced traffic load can not be handled by the system, the queue length of the α-queue will continue to increase, which may eventually cross subsequent thresholds, $T_{onset,j}$, $j > 1$. This indicates that the traffic load has to be reduced further by sending more flows to the β-queue. The number of flows that should be sent to the β-queue as subsequent thresholds are crossed is left open in [2].

We note that the congestion detection mechanism in Hayes' work reacts pre-emptively to a congestion event, that is, when the set of threshold ($T_{onset,j}$) is crossed, congestion is predicted to occur. Hence, the delay requirement of packets being served would still be met unless the α-queue was to overflow. Note also that the DQ discipline does not predict the bandwidth of any flows in the system to detect congestion. Instead it observes the queue length of the system to predict the delay experienced by all flows in the system.

There are many algorithms suggested for selecting a flow to be moved to the β-queue when congestion is detected. Selection of the algorithm is important to achieve the various goals of the DQ discipline. Some of these algorithms are discussed in detail in Section 4.3.3.

Congestion Abatement

When the length of the α-queue drops below the abatement threshold, T_{abate}, one or more flows in the β-queue will be moved back to the α-queue. All packets in the β-queue which belong to any particular flow that is being moved to the α-queue, as well as any subsequent packets arriving from the same flow, will be moved to the α-queue.

There are also many algorithms suggested for selecting a flow to be redirected back to the α-queue, and an appropriate selection is again made to satisfy the goals of the DQ discipline. However, the system will not move any packets back to the α-queue that have waited for too long (longer than the delay requirement of the α-queue) in the β-queue. These packets are simply dropped.

Flow Selection

Flows are moved between the α-queue and the β-queue under the DQ scheduling scheme. There are numerous algorithms that can be used for selecting flows to be moved to the β-queue, depending on the desired objectives. In [2], some algorithms are described and we re-iterate these here as follows:

1. Random: Choose a flow randomly among the active flows being served in the α-queue. This algorithm assumes all flows are equal (regardless of their load) and hence the performance degradation is spread across all flows over their lifetime.

2. Fair: Choose the flow that has the most packets in the α-queue when QoS Violation is detected. This algorithm penalises the flows whose instantaneous traffic load is higher than the others.

3. Next packet: Choose the flow of the packet that triggers the QoS Violation alarm. This algorithm is a mix between the two algorithms explained above. The next packet is likely to be from the flows with the highest traffic load and

there is also some randomisation effect by choosing the next packet's flow to be penalised.

4. Historical: Choose a flow that has been in the β-queue recently. This algorithm attempts to minimise the number of flows that experience performance degradation over their lifetime. Clearly, this is not fair to all the flows being served.

There are also many algorithms for selecting which flows should be redirected back to the α-queue when the QoS Recovery Detection indicates the α-queue can accommodate another flow. In [2], the authors discussed 4 possible algorithms for redirecting flows back to the α-queue; two of them involve moving multiple flows simultaneously and the other two involve moving a single flow at a time.

As we have described before, the selected algorithm for transferring flows/packets from the β-queue back to the α-queue determines the β-queue structure. The β-queue structure should support the selected algorithm for moving flows back to the α-queue. Each of these algorithms is explained in detail below.

1. FIFO: This algorithm is considered one of the simplest transferral algorithms. The β-queue structure is a simple FIFO queue. Packets are inserted into the β-queue in FIFO order. When the length of the α-queue drops below T_{abate}, the flows/packets in the β-queue are also transferred in FIFO order. Therefore, multiple flows from the β-queue are moved at the same time.

2. Fair: This algorithm attempts to provide fairness to flows currently in the β-queue. When the length of the α-queue drops below T_{abate}, the β-queue is served using a Fair Queuing (FQ) approach [9].

3. FIFO-FIFO: Each flow from the α-queue is sent to a unique β-queue. When the length of the α-queue drops below T_{abate}, packets from the first flow sent to the β-queue are redirected back to the α-queue, so that flows are served in order of how they were moved out of the α-queue (FIFO). Since each flow is

maintained separately in FIFO β-queues, the packets within a flow are also served in FIFO order.

4. LIFO-FIFO: Similar to the previous algorithm, each flow moved from the α-queue is sent to a unique β-queue. When the length of the α-queue drops below T_{abate}, packets from the flow most recently sent to the β-queue are redirected back to the α-queue, so that flows are served in reverse order of how they were moved out of the α-queue. Within a flow, the packets are still served in FIFO order. This algorithm is similar in concept to the idea of penalising a few flows as much as possible in order to minimise the number of flows that experience performance degradation at any point in time.

4.3.3 Core Dual-Queue Mechanisms

We now generalise the core DQ mechanisms in the discipline. The core DQ mechanisms include flow prioritisation, QoS Violation Detection, response to QoS Violation Detection, QoS Recovery Detection, response to QoS Recovery Detection and EPD mechanism. The flow diagram for these mechanisms is shown in Figure 4.3.2.

Flow Prioritisation

We claim that a flow prioritisation mechanism is needed to determine which flow is to be moved between the α-queue and the β-queue. This flow prioritisation mechanism must be formulated based on objectives defined for the network, such as maximising throughput and maintaining fairness. In addition, each flow must be assigned a unique priority.

In order to assign a unique priority to each flow, some tie-breaker algorithms may be required. For example, if the network's objective is to maximise throughput, the highest priority would be assigned to the flow with the highest PHY mode. When there are two or more flows with the highest PHY mode, one of the possible tie breaker algorithms may be based on flows' index.

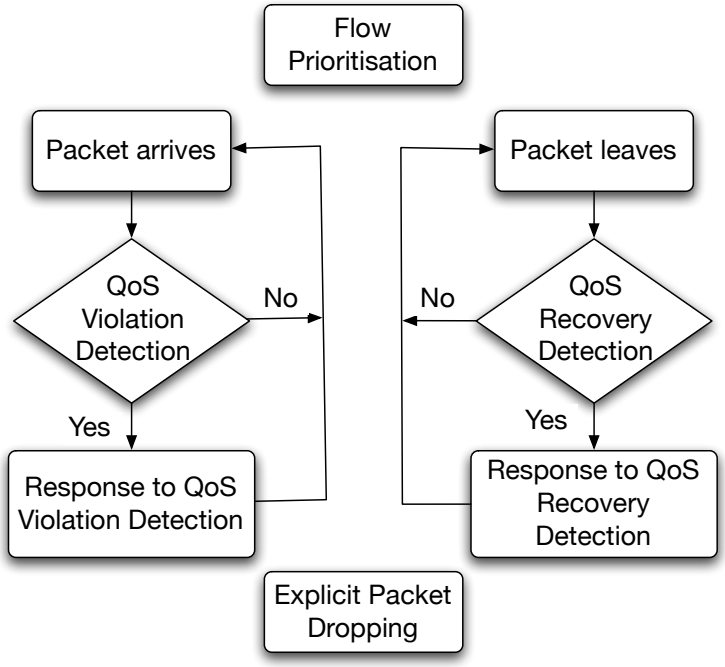

Figure 4.3.2: Flow diagram of the original DQ scheduling discipline.

Since the β-queue is designed to store flows with degraded service, the β-queue should not store any flows that have a higher priority than those in the α-queue.

QoS Violation Detection

The main mechanism of the original DQ discipline is to detect an increase in delay experienced by the network. This delay detection mechanism is used to indicate whether the delay experienced by the flows in the network has exceeded the delay requirement of the network. We generalise this delay detection mechanism as QoS detection as we contend that other QoS parameters can be considered, such as throughput, packet loss or jitter performance, if these aspects are more appropriate.

We utilise the term "QoS Violation" rather than "congestion" because we apply the detection process to the α-queue only. For the flows in the α-queue, there should be no QoS Violation even though the network is congested, as they will not experience the impact of the congestion. Hence, we differentiate a QoS Violation

event in the α-queue from a congestion event.

QoS Violation Detection can be explained using bandwidth demand and supply concepts at the α-queue, as shown in Figure 4.3.3. In Figure 4.3.3, we assume that the bandwidth demand and supply values are calculated over a long-term, and as such, transient effects have been removed. In other words, whenever the bandwidth demand is higher than the bandwidth supply, a QoS Violation has occurred. Hence, all the flows in the α-queue will experience "good service" when the α-queue's bandwidth demand is less than its bandwidth supply. However, when the α-queue's bandwidth demand is more than its long-term bandwidth supply, some flows in the α-queue will experience "degraded service" and hence a QoS Violation event will occur. Therefore, the QoS Violation Detection is also equivalent to determining whether the long-term α-queue's bandwidth demand is more than its long-term bandwidth supply.

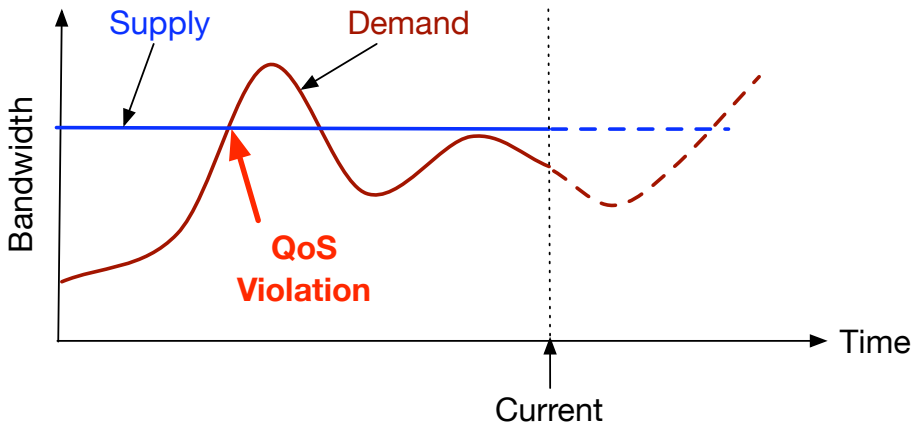

Figure 4.3.3: Long-term bandwidth supply and demand graph to explain QoS Violation Detection.

In [2], the transmission rate in a wired network is considered constant and hence the length of the α-queue in bytes can be used to indicate the maximum waiting time of the packets. This corresponds to the maximum delay of the packets served by the α-queue (assuming the packet processing time is negligible). Since the size of

the α-queue corresponds to the delay requirement of the network, and the α-queue never overflows because one or more flows are being moved to the β-queue when one or more of the thresholds, $T_{onset,j}$, $j > 0$, are crossed, we notice that the QoS Violation Detection in [2] acts to avoid a QoS Violation event at the α-queue. This is possible because the long-term α-queue's bandwidth supply is known in advance, which is constant. Hence, when the α-queue's bandwidth demand, which is detected by the set of $T_{onset,j}$ thresholds, approaches its bandwidth supply, a QoS Violation event is detected in advance or a QoS Violation alarm is detected. If the α-queue were to overflow, it would indicate that a QoS Violation event had occurred and the flows served by the α-queue could no longer maintain their QoS requirements.

Response to QoS Violation Detection

When QoS Violation Detection detects a QoS Violation event, it indicates there is a need to reduce the amount of traffic going into the α-queue, which can be achieved by moving one or more flows to the β-queue. This implies that any current packets in the α-queue from the flow being moved, as well as any subsequent packets from this flow should be placed in the β-queue.

Since the purpose of moving one or more flows from the α-queue to the β-queue is to reduce the offered load of the α-queue, the flow to be sent to the β-queue must be an active flow, where we define an active flow as a flow with packets arriving or waiting for transmission at the queue. This flow also should have the lowest priority among the flows at the α-queue, to maintain the prioritisation goals set by the network provider.

QoS Recovery Detection

Having explained the mechanism to detect a QoS Violation event, we now consider a mechanism for the network to indicate that the network has recovered from the QoS Violation event, and so some of the flows being stored at the β-queue could successfully be re-accommodated by the α-queue. We call this QoS Recovery Detec-

tion. Equivalently, QoS Recovery Detection checks if the α-queue can accommodate any flows at the β-queue.

In Hayes' work, the QoS Recovery Detection is introduced via the abatement threshold ($T_{abate} \geq 0$). When a packet leaves the system, QoS Recovery Detection is carried out. If the length of the α-queue drops below T_{abate}, the QoS Recovery Detection indicates that the α-queue is able to accommodate one or more flows from the β-queue without causing a QoS Violation alarm being detected by the QoS Violation Detection, that is the $T_{onset,j}$ thresholds are not crossed. Therefore, selecting $T_{abate} = 0$ is one of the most simplest choices and this is what used in [2].

Response to QoS Recovery Detection

When the QoS Recovery Detection indicates that the α-queue is able to accommodate one or more flows, one or more flows will be redirected back to the α-queue. Hence, all packets from the chosen flow(s) at the β-queue will be moved to the α-queue. Note that the number of existing packets being moved from the β-queue to the α-queue is such that none of the $T_{onset,j}$ thresholds are crossed. Similarly, the flow from the β-queue that is selected to be redirected back to the α-queue must be an active flow, and should have the highest priority among the flows at the β-queue.

Explicit Packet Dropping

In [2], Hayes *et al.* propose that any packets at the β-queue that have waited for too long (at least longer than the delay requirement of the network) must not be redirected back to the α-queue when the QoS Recovery Detection indicates the α-queue is able to accommodate one or more flows. These packets are simply dropped by a mechanism called the Explicit Packet Dropping (EPD) mechanism. In Hayes' work, the actual implementation of the EPD mechanism is left open.

The motivation behind the EPD mechanism is as follows. The DQ is designed to handle real-time traffic, which has tight delay requirements. Hence, packets that have experienced a delay of more than their delay requirement are most likely to

have less value to the receiver. Therefore, they should be dropped if their delay is more than D_{drop}, where $D_{drop} \geq D_{req}$, in order to save bandwidth, and hence serve other packets that have potential to meet their delay requirement..

4.4 Issues of Dual-Queue in Wireless 802.16 Environment

As we discussed in the previous section, the DQ discipline consists of 6 core DQ mechanisms, namely flow prioritisation, QoS Violation Detection, response to QoS Violation Detection, QoS Recovery Detection, response to QoS Recovery Detection and EPD mechanism. In this section, we explore the issues of implementing the DQ discipline in the IEEE 802.16 environment.

In [2], Hayes *et al.* propose the DQ scheduling discipline for real-time data services such as multimedia data, MPEG video streaming and VoIP. In our work, we employ the DQ discipline to handle real-time services in the 802.16 environment. From the 802.16 standard, there are some CoSs defined specifically for real-time services, such as rtPS and ertPS. These CoSs have slightly different QoS requirements, but all share a common delay requirement for achieving good service. The main problem in employing the DQ discipline in 802.16 environments is the design of the QoS Violation Detection and QoS Recovery Detection algorithms for both the DL and the UL. Since the DL and the UL scheduling processes are quite different, we investigate the QoS Violation Detection and QoS Recovery Detection algorithms for the DL and the UL separately. Further, we discuss the queue structure needed in order to support the packet fragmentation and packing feature in the 802.16 standard, which is important to improve transmission efficiency and other systems parameters, as explained in Chapter 2. Another issue that we discuss in this section is ARQ.

In this section, we discuss all of the above issues in detail explaining why the techniques of Hayes *et al.* need to be redefined for the 802.16 environment. In

Chapter 5, we give a detailed description of our proposal for implementing the DQ concept into an 802.16 system.

4.4.1 QoS Violation Detection and QoS Recovery Detection for the DL

Similar to Hayes' work, we focus on real-time services, however, in an 802.16 systems. Hence, we consider delay as the main QoS requirement in both QoS Violation Detection and QoS Recovery Detection. However, the QoS Violation Detection and QoS Recovery Detection algorithms in Hayes' work can not be applied directly to the DL scheduling of a 802.16 system. The QoS Violation Detection and QoS Recovery Detection algorithms in Hayes' work use the queue length of the α-queue for determining the expected delay experienced by the packets in the queue, which indicates either congestion has occurred or abated. In the 802.16 environment, we contend that the queue length of the α-queue can not be used to determine the delay experienced by the packets in the queue.

In [2], the outgoing transmission rate in a wired network is constant, which corresponds to a constant bandwidth supply to the wired network. Hence, the QoS Violation Detection is equivalent to detecting if the bandwidth demand is more than or equal to the bandwidth supply, where the queue length of the α-queue corresponds to the bandwidth demand. When the outgoing transmission rate is constant, the queue length can be used to accurately determine the delay of the packets in the queue. For example, let us consider a queue with 100 packets of fixed size at a certain time. If all of these packets are identical and the data rate is assumed to be constant, says the queue can serve 1 packet per 1 ms, the queue requires 100 ms to finish transmitting the 100 packets, therefore the last packet of these packets is expected to experience a delay of 100 ms.

If we consider the α-queue's bandwidth demand and supply in an 802.16 system, we observe that the α-queue's bandwidth supply varies with time. This is because

the bandwidth supply (available) to serve the DQ traffic that is made up of real-time services may vary from time to time. Therefore, we can not carry out QoS Violation Detection based on the bandwidth demand only, which is inferred from the queue length of the α-queue.

There are two factors that determine the meaning of bandwidth supply or bandwidth available to serve the DQ traffic: the number of slots available to serve the DQ traffic and the amount of traffic that can be carried on these slots, due to the support for AMC.

As discussed in Chapter 2, the scheduling process of 802.16 systems is carried out on a per MAC frame basis. Hence, the amount of DL bandwidth (in terms of the number of slots) available to transmit the DQ traffic (real-time services) may vary every MAC frame. Without knowing how many slots are allocated to the DQ traffic in advance, we can not use the length of the α-queue to indicate the delay experienced by the packets.

Further, 802.16 systems support AMC, where each SS has a set of supported PHY modes, which correspond to different physical transmission rates. The decision on which PHY mode to use for each SS may dynamically change according to physical conditions. Hence, without knowing the PHY mode being used for each SS in advance, again the length of the α-queue can not be used to indicate the delay experienced by the packets at the queue.

Similarly, the QoS Recovery Detection algorithm in Hayes' work can not be used in the 802.16 environment because it depends closely on QoS Violation Detection. Therefore, $T_{abate} = 0$ in the α-queue can not be used in the 802.16 environment.

Let us revisit the "Titanic analogy" described earlier in Section 4.2. The number of slots allocated to serve the DQ traffic is analogous to the number of emergency boats, whilst the amount of traffic that can be carried on each slot due to AMC is analogous to the number of people that can be carried on each emergency boat. The number of packets queueing at the α-queue is analogous to the number of people waiting to board to the emergency boats. Without knowing the number of

emergency boats and the number of people that can be carried on each emergency boat, the Titanic captain is not able to know whether any people are going to miss out on an emergency boat, assuming he takes the DQ approach.

In conclusion, the technique described in Hayes' work, which is based on the queue length of the α-queue can not be used for QoS Violation Detection or QoS Recovery Detection in the 802.16 environment, and so we need to construct a different method to carry out QoS Violation Detection, and hence using similar approaches, construct a method to carry out QoS Recovery Detection.

4.4.2 QoS Violation Detection and QoS Recovery Detection for the UL

Similar to DL scheduling, the queue length of the α-queue can not be used to determine the delay experienced by the packets in the queue for the UL scheduling. In fact, the scheduling process for the UL is more complicated due to the limited information available to the BS for UL scheduling.

In [2], the α-queue and the β-queue are physically located at the same place as the scheduler. However, the queues involved in the Uplink Dual-Queue (ULDQ) scheduling for the 802.16 network are physically located in their respective SSs, while the ULDQ scheduling process is carried out in the BS. Therefore, in order to support the ULDQ scheduling, the BS needs "virtual" queues that represent the actual queues in the SSs. This complicates the ULDQ scheduling process as the virtual queues may have errors, and these would introduce additional errors to the scheduling process.

Furthermore, in order for the BS to represent the queues in the SSs, the SS queue information has to be explicitly transmitted to the BS, through the bandwidth request mechanism defined by the 802.16 standard. However, the SS queue information transmitted from the SSs to the BS is limited. The SS queue information available for the ULDQ scheduling is the number of bytes of data waiting

for transmission for each CoS from each SS. For instance, the exact arrival time or departure time of packets in the SSs is not known to the BS.

Given the same issues as discussed for the DL, plus the limited amount of UL information, the BS has to construct more complicated solutions for the QoS Violation Detection and QoS Recovery Detection algorithms for the UL of 802.16 systems.

4.4.3 Queue Structure

This problem is related to the implementation of the buffers in the 802.16 MAC layer for holding packets that are waiting for transmission. Even though this is not mandatory, it is important to improve the transmission efficiency of the network. In [2], a single FIFO queue implementation for the α-queue is feasible because the α-queue has a single outgoing destination to maintain a pure FIFO serving order. However, we contend that the single FIFO queue implementation is not suitable in 802.16 environments.

Fragmentation and packing of packets is common practice in the 802.16 standard, in order to improve transmission efficiency because the amount of data that can be accommodated into a slot is fixed, depending only on the PHY mode of the SS. Fragmentation of packets involves the sub-dividion of packets into smaller fragments for transmission when a whole packet can not be placed into a slot for transmission. In contrast, packing of packets is needed when a slot can accommodate more than one packet. Therefore, fragmentation and packing of packets is practised in 802.16 systems to improve transmission efficiency and hence other systems parameters.

For example, a packet at the head of an α-queue can be fully transmitted with some amount of remaining bandwidth available within this slot for another full packet or partial packet. However, if the next packet in the α-queue does not have the same destination as the packet at the head of the α-queue, the scheduler has to search for another packet with the same destination as the current packet. This behaviour violates the pure FIFO order of the α-queue. Hence, a pure FIFO

relationship in the α-queue does not exist when fragmentation and packing of packets is allowed in 802.16 systems.

Furthermore, data transmission in 802.16 networks is carried out according to the DL or the UL burst, which is characterised by a set of parameters called a burst profile. A burst profile contains the PHY mode information of the DL or UL burst. Data which belongs to the same burst profile is grouped and transmitted together as a burst. These bursts are ordered from the more robust burst profiles (the low PHY mode), to the less robust burst profiles (the high PHY mode). Due to the burst-based transmission of the 802.16 network, a pure FIFO queue structure for the α-queue is not feasible.

Therefore, we contend that a single FIFO queue structure for the α-queue is not suitable in the 802.16 environments. We need a queue structure for the α-queue that supports the scheduling features of the 802.16 standard. Similarly, we need a suitable queue structure for the β-queue.

4.4.4 Automatic Repeat Request

As we have mentioned in Chapter 2, Automatic Repeat Request (ARQ) is responsible for handling transmission errors due to the medium. In a wired environment, transmission errors due to the medium are relatively rare and hence their consideration in a formal performance analysis is relatively unimportant. Consequently, there was no significant motivation for Hayes *et al.* to consider ARQ in their work. However, in wireless environments, transmission errors are common and hence ARQ is an important component of the MAC layer. Some related work that investigates the performance and impact of ARQ traffic in 802.16 systems includes [107], [108] and [109]. From our perspective, the main concerns about significant amounts of ARQ traffic involve: impact of the ARQ traffic on QoS and decision making with respect to bandwidth allocation.

In [107], ARQ traffic that includes retransmission traffic and ARQ feedback

traffic is transmitted at highest priority, before untransmitted packets. In fact, this is a traditionally sensible approach as ARQ traffic has been shown to impact the QoS of the 802.16 systems in [107]–[109]. Consequently, if the bit error rates are high and there is a significant volume of ARQ traffic, the QoS perceived by connections can still be seriously impacted. Essentially, this is due to a lower effective transmission rate of the wireless channel.

Within a DQ environment, we have the choice of continuing the traditional approach, or placing the ARQ traffic back into the front of the respective connection transmit buffer. Using the latter approach, if a specific connection is experiencing poor radio performance and hence higher rate of re-transmission, its impact on other connections is lessened. We note that this impacts the bandwidth allocation process as the ARQ traffic for each connection, in effect, is included as part of the overall bandwidth allocated to that connection. Further, a choice needs to be made as to whether or not the EPD mechanism is also be applied to the ARQ traffic.

However, these issues must be considered in conjunction with how ARQ is implemented in a given 802.16 system. In the 802.16 standard, the actual implementation of ARQ mechanisms is not specified. The standard only provides a framework for the ARQ mechanisms and allows for example, the ARQ retransmissions to be carried out on a per slot or per packet basis.

4.5 Summary

In this chapter, we have discussed approaches for achieving customer satisfaction. Since customer satisfaction is a highly subjective and abstract concept, it is complex to define and difficult to compare customer satisfaction across different individuals and different products. Research on the topic of customer satisfaction can be found in various service sectors, such as the fast food service, telecommunication service, hotel service, tourism service and medical service.

From these studies, we have found that there is strong evidence showing a firm

causal linkage between customer satisfaction and service performance. Therefore, in order to achieve customer satisfaction in telecommunication services, the network provider should provide excellent service performance. Service performance in the telecommunication field is affected by customer service and network performance. In our research, we propose to improve network performance during periods of congestion, in order to achieve a better customer satisfaction. We have contended that providing fairness to the customers does not achieve the best outcome for customer satisfaction.

We have proposed the Dual-Queue scheduling discipline to maximise the number of satisfied customers in an 802.16 network, as opposed to providing fairness to customers. We have also generalised and identified 6 core DQ mechanisms of the DQ discipline: flow prioritisation, QoS Violation Detection, response to QoS Violation Detection, QoS Recovery Detection, response to QoS Recovery Detection and Explicit Packet Dropping mechanism.

Towards the end of the chapter, we have discussed the issues behind employing the DQ scheduling discipline in the 802.16 environment. We have explained the reasons why direct deployment of the original DQ algorithms from Hayes' work into the 802.16 environment is not possible. The problems that we have identified, have been discussed in detail.

In the next chapter, we propose a particular deployment of the DQ discipline into the 802.16 environment. In doing so, we are able to achieve the goal of maximising the number of satisfied customers in an 802.16 system for real-time services.

Chapter 5

Dual-Queue for 802.16 Environments

5.1 Introduction

In this chapter, we propose a DQ implementation in 802.16 systems for handling real-time services on both the Downlink and the Uplink. Firstly, we investigate and propose the framework required for implementing a DQ scheduling discipline in an 802.16 environment. This covers issues including the structure of the α-queue and the β-queue, handling ARQ traffic and the service scheme deployed by the DQ scheduler.

We propose approaches to address the challenges associated with the variable bandwidth available to serve the DQ traffic. These challenges stem from the unpredictability of the number of slots allocated to the DQ traffic and the support for AMC, where a PHY mode for a connection may change dynamically according to physical conditions. These approaches are formulated into the core DQ mechanisms on the DL and the UL separately. Through examination, we demonstrate that our proposed approach is able to cope with these issues and provide excellent QoS to the end users.

Further, we compare the performance of our proposed DL and UL DQ schedulers to a WFQ scheduler under the existence of the ARQ traffic. We also investigate the impact of enhancing the WFQ scheduler with the EPD mechanism since the DQ scheduler is integrated with the EPD mechanism. The performance comparison is carried out based on the overall proportion of "good service" experienced by each connection, the use of network resources and the total "good service" MAC throughput achieved by all connections.

Recall from Chapter 2 that in this thesis, we only consider 802.16 systems that operate in the point-to-multipoint mode using ATDD.

5.2 Dual-Queue Framework for 802.16

In this section, we propose a framework for the Downlink Dual-Queue and the Uplink Dual-Queue schedulers in 802.16 systems. We investigate the DLDQ and the ULDQ schedulers independently, in which the DLDQ scheduler is more straightforward than the ULDQ scheduler because the BS has access to complete queue information.

In the following, we show the differences between the DLDQ and the ULDQ schedulers in the structure of the α-queue and the β-queue. We defer the detailed discussion of the core DQ mechanisms on the DL and the UL to Sections 5.3 and 5.6 respectively. However, we note that the DLDQ and ULDQ schedulers share some similarities in handling ARQ and in employing their service scheme.

5.2.1 Structure of the α-queue and the β-queue

DL queue structure

As we discussed in Chapter 4, a single FIFO α-queue structure is not applicable in an 802.16 environment. Since the scheduling process in 802.16 systems is carried out on a per connection basis, we propose to allocate a single FIFO queue for each of the connections at the BS. Hence, the α-queue is made up of multiple FIFO queues,

as shown in Figure 5.2.1.

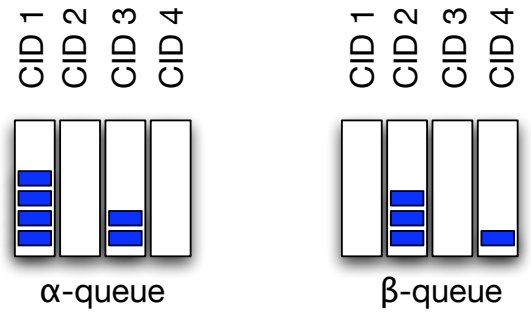

Figure 5.2.1: Multiple FIFO queues for the α-queue and the β-queue.

By utilising multiple FIFO queues to buffer the connections separately, packing and fragmentation of packets can be readily supported. Further, this queue structure also provides the flexibility to apply different queueing schemes across the queues within the α-queue, such as the WFQ and RR algorithms. Other queueing schemes can be employed as required because the DQ scheduler can operate along with any queueing schemes.

In the case of the β-queue, we propose the same structure as that for the α-queue, since packets are moved to the β-queue on a per connection basis and the β-queue is also served on this basis.

UL queue structure

For the UL, each connection is allocated a single FIFO queue at the SSs. In order to carry out UL scheduling for all SSs, the BS makes use of the Bandwidth Request mechanism to obtain queue information from the SSs. Hence, the UL α-queue and the β-queue at the BS are used to store bandwidth request information for all UL connections. When a UL connection is sent to the β-queue, its Bandwidth Request information will also be moved to the β-queue within the BS.

5.2.2 Handling Automatic Repeat Request

The ARQ process is responsible for retransmission of data corrupted in the previous frames. We propose to store the ARQ traffic in a separate queue, and to serve the ARQ traffic ahead of the α-queue and the β-queue. By doing this, we attempt to guarantee the QoS of the ARQ traffic since it is given more transmission opportunities than the α-queue and the β-queue. Note that the ARQ traffic only exists for connections that have been allocated transmission slots in the previous frame.

We assume that the amount of ARQ traffic is relatively small and hence a strict limit on the amount of ARQ traffic to be transmitted is not necessary, but it can be implemented if required.

Since there is no guarantee that retransmitted data will not be corrupted again, we do not apply QoS Violation Detection on the ARQ traffic, in order to avoid detecting a QoS Violation event due to the ARQ traffic that happens to be retransmitted a few times. However, we apply the EPD mechanism on the ARQ traffic to ensure no packets have a delay of more than D_{drop}.

There are different ARQ algorithms defined in the 802.16 standard. Since the ARQ process is not our research focus, we implement a specific ARQ algorithm for the DLDQ scheduler. Our simple ARQ algorithm is carried out on a per slot basis. For every slot transmitted, an acknowledgement of receipt is required. If a slot is corrupted during transmission, this slot will be retransmitted in the next MAC frame. When there is a change in PHY mode for a SS that has ARQ traffic, its ARQ traffic has to be repacked according to the new PHY mode.

5.2.3 Service Scheme

In the following, we propose the same service scheme for the DLDQ and the ULDQ schedulers. We use the DLDQ scheduler as an example to explain the service scheme proposed.

Under uncongested conditions, all DL connections will be served at the α-queue.

When QoS Violation events are detected, one or more connections will be sent to the β-queue. Whenever there are connections at both the α-queue and the β-queue, the DQ scheduler always serves the α-queue first, followed by the β-queue. If there is bandwidth available after serving the α-queue, the β-queue will be served. This is slightly different from the original DQ algorithm proposed in Hayes' work, where all packets are allowed to leave the system only through the α-queue.

In order to give the β-queue a chance of being served, the total number of requested slots for the DL real-time services ($X_{dl,n}$) must be calculated as the sum of the number of slots requested for the α-queue ($X_{dl,\alpha,n}$) and the β-queue ($X_{dl,\beta,n}$).

We also assume that the number of slots required for the ARQ traffic ($X_{dl,ARQ,n}$) is always less than the total number of slots allocated for data transmission ($S_{dl,n}$), that is, $X_{dl,ARQ,n} < S_{dl,n}$, since, if $X_{dl,ARQ,n} \geq S_{dl,n}$, the α-queue will not be served.

Given the total number of DL slots allocated in MAC frame n ($S_{dl,n}$) and $X_{dl,ARQ,n} > 0$, the number of DL slots allocated for the α-queue, $S_{dl,\alpha,n}$, is given by

$$S_{dl,\alpha,n} = \min([S_{dl,n} - X_{dl,ARQ,n}]^+, X_{dl,\alpha,n}), \tag{5.2.1}$$

where

$$[A]^+ = \begin{cases} A, & \text{if } A \geq 0, \\ 0, & \text{if } A \leq 0. \end{cases}$$

Hence, the number of slots left to serve the β-queue, $S_{dl,\beta,n}$, is

$$S_{dl,\beta,n} = S_{dl,n} - S_{dl,\alpha,n} - X_{dl,ARQ,n}. \tag{5.2.2}$$

As discussed in Chapter 4, the slots allocated to serve the β-queue ($S_{dl,\beta,n}$) can be assigned to a single connection or multiple connections at the β-queue. For the α-queue, we allocate slots to all connections in the α-queue according to the WFQ algorithm when $S_{dl,\alpha,n} < X_{dl,\alpha,n}$, where all connections have equal weight.

5.2.4 Scheduling Flow Diagram

In Figure 5.2.2, we show the flow diagram of a DQ scheduler. The main differentiators of the DLDQ and the ULDQ schedulers lie in the core DQ mechanisms, which include QoS Violation Detection, QoS Recovery Detection and Explicit Packet Dropping. They are described in detail in Sections 5.3 and 5.6 respectively.

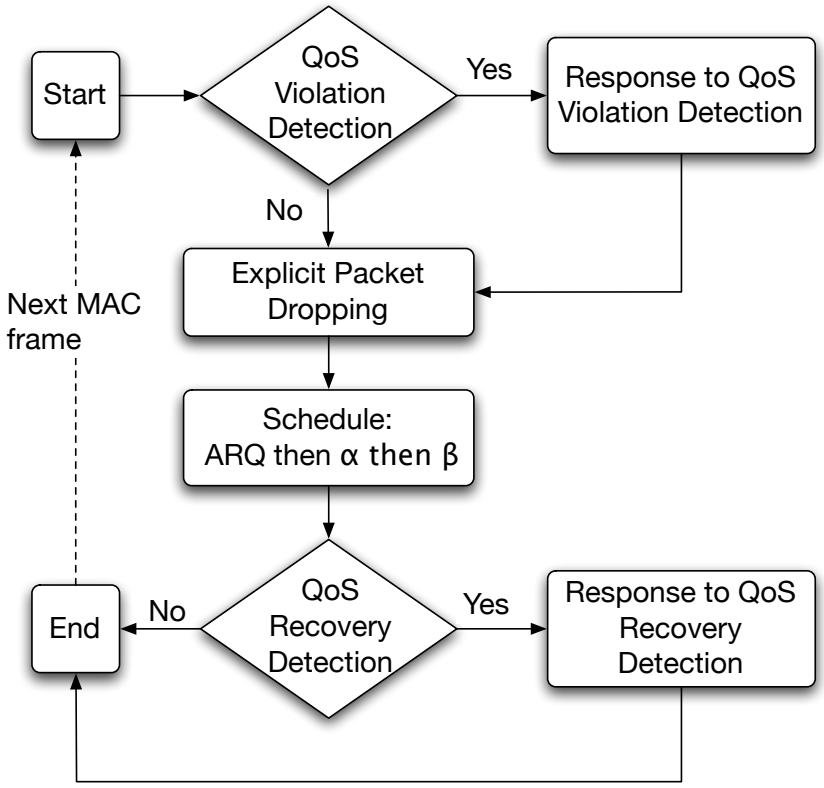

Figure 5.2.2: Flow diagram of a DQ scheduler in 802.16 systems.

5.3 Core Downlink Dual-Queue Mechanisms

In the following, we propose the QoS Violation Detection and QoS Recovery Detection algorithms for the DLDQ scheduler. We also define the response taken by the scheduler when QoS Violation Detection indicates the α-queue is congested and when QoS Recovery Detection indicates the α-queue is able to accommodate another

connection. Further, we propose an algorithm for the EPD mechanism.

5.3.1 Connection Prioritisation

The connection prioritisation mechanism is based on the objectives defined for the 802.16 system under consideration, such as maximising throughput and revenue. Hence, various connection priorisation algorithms can be employed. Further, we need tie-breaker algorithms to assign a unique priority to each connection in the system. The DQ scheduler, hence, makes sure that no connections at the α-queue have lower priority than those connections at the β-queue.

5.3.2 QoS Violation Detection

QoS Violation Detection is carried out to detect any QoS Violation event at the α-queue. Since the QoS constraint under consideration is the delay requirement of the packets being served at the α-queue, one obvious solution is to time-stamp these packets on arrival and check their delay on transmission.

Similarly, we can approach the problem using frame-stamping for 802.16 systems, due to their framed nature. Instead of stamping the arrival time, we stamp the arrival MAC frame number on arriving packets and check the delay of these packets on transmission in terms of the number of MAC frame durations. In fact, the frame-stamping approach is equivalent to time-stamping with timing resolution that equals the duration of a MAC frame. In general, frame-stamping requires less computational resources in actual deployment.

A QoS Violation event is indicated where the delay of a transmitted, or, currently being transmitted packet, exceeds the defined delay requirement. In the original DQ discipline described for a wired environment [2], the QoS Violation alarm is triggered when at least one of the set of thresholds, $T_{onset,j}$, $j > 0$ is crossed. Hence, the QoS Violation Detection acts to avoid a QoS Violation event since one or more connections are moved to the β-queue when these thresholds are crossed, indicating

the possibility of congestion. However, in 802.16 systems, we argue that there is no effective way of acting before a QoS Violation event, as we are not able to predict the bandwidth available (that is the number of slots available) to serve DQ traffic and the PHY mode changes in advance. Hence, the QoS Violation Detection algorithm is designed to detect a QoS Violation event in which the QoS of at least one connection at the α-queue is actually violated, indicating congestion at the α-queue.

Note that the original DQ scheduler does not consider any detections that involve measurement or prediction of the bandwidth of any connections in the given 802.16 system, and we have chosen to be consistent with this condition.

In this thesis, we select the frame-stamping approach for the QoS Violation Detection algorithm as it requires less computational resources in actual deployment. In summary, when a DL packet p arrives at the MAC layer of the 802.16 system, the packet is stamped with its arrival MAC frame number ($F_{a,p}$) and sent to either the α-queue or the β-queue.

At any given time, the delay of the packet p being served (D_p) at the current MAC frame is calculated using the current MAC frame number (F_c) and $F_{a,p}$, as shown in equation (5.3.1), where D_{mac} equals a MAC frame duration. In fact, if the α-queue is a FIFO queue, the maximum delay of the packets served can be obtained by checking the delay of the packet at the head of the queue, since this packet is the oldest packet at the queue. The delay experienced by packet p being served at the α-queue is calculated as the worst case scenario, assuming that packet p could have arrived at the start of the MAC frame $F_{a,p}$, but it is transmitted at the end of the current MAC frame, as shown in Figure 5.3.1.

$$D_p = [(F_c - F_{a,p}) + 1] \times D_{mac}. \tag{5.3.1}$$

Instead of checking the delay of the packets being transmitted in the current MAC frame, we can predict the delay of the packets being scheduled for transmission in the next MAC frame. Based on the number of slots of data waiting for transmission and the number of slots available to serve the α-queue in the current

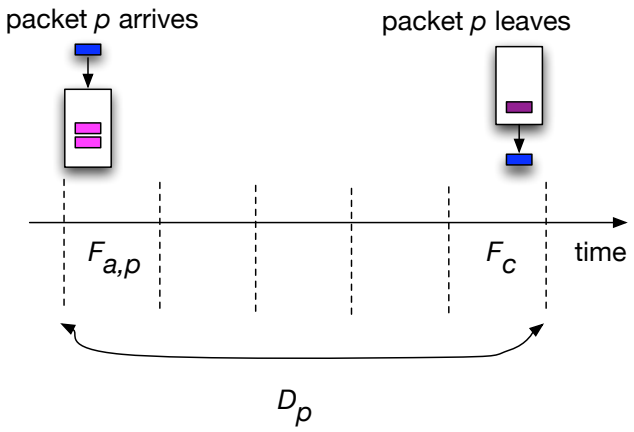

Figure 5.3.1: Delay calculation for packet p that arrives at $F_{a,p}$ and leaves at F_c.

frame, the number of packets scheduled for transmission in the current MAC frame can be calculated.

Given the calculated number of packets scheduled for transmission in the current MAC frame, QoS Violation Detection can be carried out to predict any QoS Violation events in the next MAC frame, based on the packets present but not scheduled for transmission in the current MAC frame. Assuming the first packet that is not scheduled for transmission in the current MAC frame is p^*, a QoS Violation event is detected when,

$$D_{p^*} = [(F_c - F_{a,p^*}) + 2] \times D_{mac} > D_{req}. \tag{5.3.2}$$

Notice that we can not predict the delay of packets being scheduled for transmission in MAC frames further than the next MAC frame because we only have bandwidth allocation information for the current MAC frame. For example, given $D_{req} = 50\ ms$ and $D_{mac} = 10\ ms$, a QoS Violation event is detected when $D_{p^*} > 50$ ms or equivalently when $F_c - F_{a,p^*} > 3$.

We propose to carry out QoS Violation Detection on the DL before the scheduling process in every MAC frame.

5.3.3 Response to QoS Violation Detection

Moving a connection to the β-queue

When a QoS Violation event is detected in a MAC frame, one single active connection with the lowest priority from the α-queue will be chosen and sent to the β-queue. This implies that any current packets at the α-queue for this connection, as well as any subsequent packets from this connection, will be placed in the β-queue until further notice.

How long should we wait before moving another connection?

From the above, we have seen that a single connection will be moved from the α-queue to the β-queue when a QoS Violation event is detected in a MAC frame. We must now consider how long the scheduler should wait before moving another connection to the β-queue. There is, in fact, a high probability that another QoS Violation will be detected in the next MAC frame, even though one connection has been moved to the β-queue.

If the movement of one connection from the α-queue to the β-queue is not sufficient to maintain the QoS (delay) requirement of the connections being served at the α-queue, extra connections from the α-queue should be sent to the β-queue. We contend that the DLDQ scheduler should send a minimal number of connections from the α-queue to the β-queue, in order to maximise the number of connections maintained at the α-queue at any time. After a connection at the α-queue is sent to the β-queue, the DLDQ scheduler should wait for a certain time before sending another connection, in order to avoid sending too many connections to the β-queue due to the same QoS Violation event. However, if the DLDQ scheduler waits for too long before sending another connection, the α-queue would be congested, which could result in failure to maintain the QoS requirements of the connections at the α-queue. We now propose a method to handle this problem.

When a QoS Violation event is detected, the DLDQ scheduler sends an active

connection to the β-queue and records the time (or equivalently, the current MAC frame number) that the QoS Violation event was detected (F_{qv}), as shown in Figure 5.3.2. The scheduler also marks the next packet to arrive at the α-queue after the QoS Violation event. This packet is the first to experience an α-queue with one less connection. Hence, this packet can be used to determine when to move another connection to the β-queue if required. Further, the DQ scheduler suspends the response to QoS Violation Detection for $D_{req} - 2 \times D_{mac}$ seconds, counting from MAC frame $F_{qv} + 1$. That is, the result of QoS Violation Detection will be ignored, and we consider this to be in an "off" state, until MAC frame F_{on}, where

$$[(F_{on} - F_{qv}) + 1] \times D_{mac} = D_{req}. \tag{5.3.3}$$

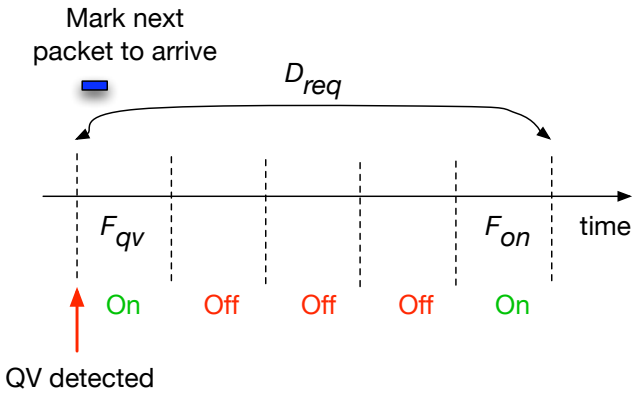

Figure 5.3.2: Required waiting time before moving another connection to the β-queue.

The DLDQ scheduler also checks if the marked packet is scheduled for transmission before MAC frame F_{on}. If the marked packet is scheduled for transmission at a MAC frame before F_{on}, the result of QoS Violation Detection will no longer be ignored, and we consider this to be in an "on" state. This is because the delay experienced by the marked packet is less than D_{req}, which means that the previous round of congestion at the α-queue has been controlled. If the α-queue is still congested after moving one connection to the β-queue, the marked packet would not

be transmitted before F_{on}.

By doing this, the scheduler makes sure that the next QoS Violation Detection process is carried out on packets that arrive after the first QoS Violation event. This allows enough time for the removal of a connection from the α-queue to take effect without unnecessarily moving extra connections to the β-queue due to the same QoS Violation event. However, this incurs a minimum waiting time between moving another connection to the β-queue after the previous connection is moved to the β-queue, which is equal to D_{req}.

As an example, if the current number of connections being served without congestion at the α-queue is 10 and there is a change in the network conditions that allows only 6 connections to be fully served at the α-queue, the DLDQ scheduler will need a minimum time of $3 \times D_{req}$ to move the remaining 3 connections to the β-queue after the first connection is moved to the β-queue following a QoS Violation detection.

5.3.4 QoS Recovery Detection

In order to maximise the number of connections at the α-queue, or equivalently maximise the number of connections with good service, QoS Recovery Detection is carried out to check if there is sufficient bandwidth available for the α-queue to accommodate a connection from the β-queue.

Unlike the original DQ algorithm, the β-queue will be served whenever there is left over bandwidth after serving the α-queue in every MAC frame. Hence, $T_{abate} = 0$ in the α-queue can not be used in the 802.16 environment. Further, we have seen that the number of slots left to serve the β-queue can be allocated to a single or multiple connections.

In order to make sure a connection can be redirected back to the α-queue as soon as possible, we propose to allocate all the slots available to serve the β-queue to a specifically chosen connection from the β-queue in every MAC frame (instead

of distributing these available slots evenly to multiple connections). This connection has the highest priority among the connections at the β-queue.

If there are slots left over after serving this connection, the remaining slots will be allocated to another specifically chosen connection, which has the next highest priority. This process is repeated until all the slots left to serve the β-queue slots are assigned to connections at the β-queue.

At any time, a single connection will be served from the β-queue whenever there is any bandwidth available after serving the α-queue. One of the simplest algorithms for QoS Recovery Detection is to detect whether a connection being served from the β-queue has no packets left to send in its own queue. However, to avoid making a false detection, we have to make sure that the packets of a connection at its queue are not emptied by the EPD mechanism. This is achieved by only applying QoS Recovery Detection to connections actually allocated transmission slots at the β-queue in the current MAC frame.

There are other possible algorithms for QoS Recovery Detection. For example, by calculating the bandwidth left over after serving the α-queue and the bandwidth required for each connection at the β-queue, the scheduler could decide whether to redirect any connections from the β-queue back to the α-queue. However, this alternative method requires the prediction of the bandwidth required for each connection at the β-queue. This conflicts with our general approach of not trying to predict bandwidth requirements and availability, but instead relying on actual observations.

5.3.5 Response to QoS Recovery Detection

When QoS Recovery Detection indicates the α-queue can accommodate another connection, the connection being served at the β-queue that has exhausted its queue, will be redirected back to the α-queue. Note that this connection must have the highest priority among the connections at the β-queue.

5.3.6 Explicit Packet Dropping

If packets have been waiting for too long in the system, they are not transmitted. Therefore, the DLDQ scheduler needs a mechanism to check the delay of the packets in the system. If these packets have been waiting for more than the dropping threshold time (D_{drop}), where $D_{drop} > D_{req}$, they are dropped. We propose that a packet p should be dropped if its delay, D_p, (assuming it is being transmitted in the current MAC frame) is greater than or equal to D_{drop}, that is,

$$D_p \geq D_{drop}. \tag{5.3.4}$$

Note that the delay of packets at the α-queue is controlled by QoS Violation Detection. A QoS Violation event is detected by QoS Violation Detection when the delay experienced by the connections at the α-queue is more than D_{req}. Hence, QoS Violation Detection strives to maintain the delay experienced by the connections at the α-queue below D_{req}. However, there is still a chance that the connections being served at the α-queue will experience a delay that is greater than D_{req} when the α-queue is congested. Therefore, it is very unlikely that a packet will be removed by the EPD mechanism at the α-queue unless there is a sudden increase in the total offered load, or a sudden drop in the bandwidth available to serve DQ traffic, at the α-queue, or D_{drop} is close to D_{req}.

Further, we propose that QoS Violation Detection has to be carried out ahead of the EPD mechanism to ensure that a QoS Violation event is not missed due to packets being removed by the EPD mechanism.

5.4 Simulation Tool and Assumptions

In this thesis, we use Network Simulator 2 (NS-2) version 2.29 to carry out our experiments. Since there was no source code for the MAC layer of an 802.16 system in NS-2 when we began our research, we have had to develop an implementation of

the MAC layer for an 802.16 system based on the source code of other MAC layers, such as 802.11 and Time-Division Multiple Access (TDMA) systems.

We make the following assumptions with regard to the operation of 802.16 systems, before we describe our experiments in the next section.

Physical Layer

In Chapter 2, we mentioned four different physical layers available in the 802.16 standard: WirelessMAN-SC, WirelessMAN-SCa, WirelessMAN-OFDM and WirelessMAN-OFDMA. The MAC layers for these physical layers are slightly different to each other. Given that the core of our research is on MAC layer scheduling designs and our work is independent of the physical layer, we only consider an example of WirelessMAN-OFDM's MAC layer in our NS-2 model. We specifically chose WirelessMAN-OFDM as our example physical layer because it is the most prevalent physical layer in the literature.

DL and UL Bursts

Given a set of data from different connections scheduled for transmission in a MAC frame, the DL transmission is carried out in order from connections with the lowest PHY rate to connections with the highest PHY rate in every MAC frame. Hence, the DL transmission with the same PHY rate is grouped and known as a DL burst, as specified in the standard.

For the UL, each SS scheduled for transmission in a MAC frame is given a UL burst. There is no particular order for these UL bursts defined in the 802.16 standard. We assume the UL bursts are ordered from the SS with the lowest index to the SS with the highest index, where a SS's index is a unique number assigned to that SS.

DL-MAP and UL-MAP

In the 802.16 standard, the DL-MAP is defined to describe the bandwidth, or slot, allocation in the current MAC frame. However, the UL-MAP can be defined to describe bandwidth allocation in either the current MAC frame or the next MAC frame. In this thesis, we assume that both the DL-MAP and UL-MAP are used to describe the DL and UL bandwidth allocation in the current MAC frame.

Connections in 802.16 systems

In the standard, each connection in an 802.16 system is assigned with a CID. For simplicity, we assume that a SS can have at most one data connection in the DL and another data connection in the UL. This allows a Voice over IP user in a SS to maintain bi-directional connections. We also assume that there is a one-to-one mapping of IP layer flows to the MAC layer connections.

Note that the term "flow" is used in discussion of the DQ discipline in [2]. In order to maintain consistency, we use the term "connection" to replace the term "flow" when we discuss the DQ scheduler in the 802.16 context. We identify the index of a connection from a SS as the index of that SS. Further, we also identify the PHY mode of a connection from a SS as the PHY mode of that SS.

Active connections

We define an active connection as a connection having data waiting for transmission at a queue in the MAC scheduler, either at the BS or the SS.

Bandwidth Request

The DQ scheduler handles real-time services which use unicast polling for sending the Bandwidth Request information to the BS. We assume that the BR information is polled on a per MAC frame basis and hence the BS has the BR information for all UL connections in every MAC frame.

Fragmentation and packing

Fragmentation and packing of packets are implemented in our NS-2 802.16 model. This feature introduces an additional 2 B overhead per fragment.

Inter-leaving

Inter-leaving is a common physical layer practice to improve error correction, and hence network performance. This is, however, not implemented in our model because our main research focus is on MAC layer scheduling.

Fixed overheads

In the standard, the amount of bandwidth available for data transmission may vary due to the variable overheads, such as the size of the DL-MAP and the UL-MAP, scheduling time and the BR overhead. However, in our experiments, we assume that the time allocated for actual data transmission in a MAC frame is fixed, which can be quantified in terms of the number of slots. Therefore, the number of slots available for data transmission (S_{mac}) is assumed to be constant for every MAC frame.

Number of slots for DQ scheduling

We use the notation that the total number of slots allocated to DL and UL real-time services in MAC frame n are $S_{dl,n}$ and $S_{ul,n}$.

PHY modes

We consider 5 PHY modes as defined in Table 5.4.1. These sample 802.16 system values are obtained from [23], [46], [110]. The relationship used to calculate these values is given in Appendix A.

Table 5.4.1: The parameters of each PHY mode considered in our 802.16 systems.

Modulation	Channel coding	Rate ($Mbps$)	Slot size (B)	PHY header (B)	MAC header (B)	Payload (B)
64-QAM	3/4	22.5	108	27	6	75
64-QAM	2/3	20.0	96	24	6	66
16-QAM	3/4	15.0	72	18	6	48
16-QAM	1/2	10.0	48	12	6	30
QPSK	1/2	5.0	24	6	6	12

5.5 Downlink Experiments

In this section, we validate our proposed algorithms for the DLDQ scheduling. In Chapter 4, we described two factors that affect the bandwidth available to serve the DQ traffic: changing the number of slots to serve DQ traffic and changing the PHY mode. Therefore, we conduct two experiments (one for each of these factors) for testing our proposed algorithms.

In order to have a predictable environment to demonstrate the DQ scheduling features, we use Constant Bit Rate traffic sources in our experiments. We use rtPS as the DQ traffic being handled by the DLDQ scheduler. In the following, the term "DQ bandwidth" refers to the bandwidth available to serve the DQ traffic, which is based on the number of slots to serve the DQ traffic and the PHY mode of connections in the 802.16 systems. We also define the term "MAC throughput" of a connection referring to the amount of data, measured as the number of bits received by the MAC layer of its receiver minus the MAC header overhead, per second, which is calculated for every MAC frame.

For the rest of the experiments in this thesis, unless otherwise specified, the system parameters in Table 5.5.1 are used. Note that one of the most delay sensitive real-time traffics is VoIP. According to [111], an end-to-end delay for a local VoIP call of less than 150 ms is generally accepted by most application users. For an

international call, an end-to-end delay of up to 400 ms is acceptable. In [112], Bolot shows that most of the data points in his investigation for round-trip packet delay in the Internet, fall in the range of 130 ms to 600 ms. This corresponds to a symmetric one-way delay in the range from 65 ms to 300 ms. Other research carried out by Corlett *et al.* [113] shows that the one-way delay of a small network consisting of 11 hops, can have a peak averaged value of 50 to 60 ms.

Table 5.5.1: System parameters used in our experiments.

MAC frame duration (D_{mac})	10 ms
Delay requirement (D_{req})	50 ms
Maximum waiting time (D_{drop})	100 ms
Number of data slots in a MAC frame (S_{mac})	218
Bit Error Rate (BER)	0

Further, we assume that the end-to-end path of a VoIP session consists of two 802.16 networks, one at each end as shown in Figure 5.5.1. Given that an acceptable one-way delay of an Internet path could be 300 ms for international calls and 50 to 60 ms for a local call, this leaves us with an acceptable delay of 50 ms for each wireless end. Hence, we choose the delay requirement for our DQ system to be 50 ms. Also, D_{drop} is chosen to be 100 ms because a VoIP packet that experiences a delay of 500 ms (100 + 100 + 300) for international calls or a delay of 250 ms (100 + 100 + 50) for local calls, is not acceptable to an end user, according to our literature search.

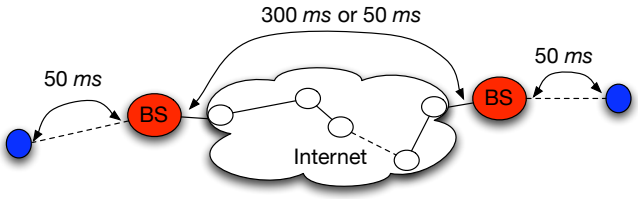

Figure 5.5.1: End-to-end delay of a VoIP packet.

The aim of deploying the DQ scheduler is to maximise the number of satisfied customers. Hence, the primary objective of each experiment in this section is to maximise the number of connections that experience good service. The secondary objective is to maximise the throughput of the system. In order to maximise throughput, the connection prioritisation mechanism is first based on PHY mode, where the highest PHY mode corresponds to the highest priority. In cases where there are two or more connections with the same PHY mode, a tie breaker is based on the indices of the connections, where the connection with the highest index is given the highest priority. Other tie breakers, including random selection can be implemented as required. Hence, the connection with the highest PHY mode and the highest index is given the highest priority in the system. This is summarised in Table 5.5.2.

Table 5.5.2: The objectives defined for the system and the connection prioritisation mechanism.

Objectives	1. Maximising the number of connections with good service. 2. Maximising MAC throughput.
Connection Prioritisation	First, based on PHY mode, where the highest PHY mode corresponds to the highest priority. Then, based on index, where the highest index corresponds to the highest priority.

With regard to the transmission ordering for the DL connections allocated with transmission slots in a MAC frame, connections are first grouped into bursts, according to their PHY mode. Then these bursts are transmitted starting from the lowest PHY mode burst, followed by the higher PHY mode burst and within a burst, connections are arranged from the lowest to the highest index.

5.5.1 Experiment 1: Changing number of slots to serve DQ traffic

This experiment investigates the performance of the DLDQ scheduler in response to changing DQ bandwidth due to changes in the number of slots to serve the DQ traffic. We consider a network with four identical SSs connected to a BS. All the SSs have the same PHY mode, with PHY rate = 22.5 $Mbps$. Each SS carries a single DL rtPS connection generated by a CBR traffic source at a rate of 2 $Mbps$. The packet size of the CBR traffic source is 100 B, including 20 B of Internet Protocol (IP) header.

With a PHY rate of 22.5 $Mbps$, the amount of MAC layer data that can be transmitted in a slot is 75 B (obtained from Table 5.4.1). Therefore, the number of slots required to transmit a 100 B packet (including 20 B of IP header) is equal to $(100 + \lceil 100/75 \rceil \times 2)/75 \approx 1.39$, allowing 2 B for the fragmentation and packing sub-header. The MAC throughput of the network transmitting at a PHY rate of 22.5 $Mbps$ is hence $(S_{mac}/1.39) \times (100 - 20) \times 8/D_{mac} \approx 10.0$ $Mbps$.

To create a changing number of slots to serve the DQ traffic in this experiment, we introduce some UGS traffic, which is a UL connection from one of the SSs to the BS, generated using a CBR traffic source with packet size of 100 B (including 20 B of IP header). The reason for choosing UGS traffic is that the UGS traffic has a higher priority than the rtPS traffic, that is, the UGS traffic is served ahead of the rtPS traffic at all times. We also apply a strict priority, where it is possible for the UGS traffic to be allocated with all data slots available, leaving none to the rtPS traffic. Therefore, the DQ bandwidth is equal to the MAC throughput of the network (10 $Mbps$) minus the MAC throughput of the UGS traffic.

Experimental Regions

This experiment is carried out for 20 seconds. For the purpose of discussion, we divide this experiment into 6 regions in order to show the different DLDQ scheduling

features with respect to changing bandwidth available to serve the DQ traffic, due to changes in the number of slots to serve the DQ traffic.

- Region 1: $0 \leq t < 5$, where t is time in seconds. At $t = 3$, each connection in the network is started and the rate of each connection is set to 0.1 $Mbps$. No congestion occurs in this region since the DQ bandwidth is greater than the total offered load of the DQ traffic. This region is used for the initialisation process in NS-2.

- Region 2: $5 \leq t < 6$. The total offered load of the DQ traffic is increased to 8 $Mbps$ at $t = 5$ by increasing the rate of each DQ connection to 2 $Mbps$ and this load is maintained throughout the remainder of the experiment. Also, the DQ bandwidth is decreased to 7.5 $Mbps$ by increasing the UGS rate to 2.5 $Mbps$, and hence, congestion occurs.

- Region 3: $6 \leq t < 11$. The DQ bandwidth starts to reduce linearly at $t = 7$ until $t = 10$ due to a linear increase in the UGS rate. This region is designed to investigate the reaction of the DLDQ scheduler to a gradual decrease in DQ bandwidth.

- Region 4: $11 \leq t < 15$. The DQ bandwidth starts to increase linearly at $t = 12$ until $t = 15$ due to a linear decrease in the UGS rate. This region is designed to investigate the reaction of the DLDQ scheduler to a gradual increase in DQ bandwidth.

- Region 5: $15 \leq t < 18$. The DQ bandwidth is dropped instantaneously to 3 $Mbps$ at $t = 15$ by setting the UGS rate to 7 $Mbps$. This region is designed to investigate the reaction of the DLDQ scheduler to a sudden decrease in DQ bandwidth.

- Region 6: $18 \leq t \leq 20$. The DQ bandwidth is increased instantaneously to 9 $Mbps$ at $t = 18$ by setting the UGS rate to 1 $Mbps$. Note that there is

no congestion as the DQ bandwidth is greater than the total offered load of the real-time service. This region is designed to investigate the reaction of the DLDQ scheduler to a sudden increase in DQ bandwidth.

Region 1 ($0 \leq t < 5$)

In Figure 5.5.2 (a), we observe an increase in the total offered load of the DQ traffic and a decrease in the DQ bandwidth at $t = 3$. Note that all the graphs in this chapter except the QoS graphs, are plotted using a resolution that equals D_{mac}. Hence, average values over the D_{mac} time interval are used.

Region 2 ($5 \leq t < 6$)

Up to $t = 5$, the four DQ connections are all being served at the α-queue. At $t = 5$, the total offered load of the DQ connections exceeds the DQ bandwidth, and congestion occurs as the number of packets at the α-queue builds up and an increase in average delay of these packets is experienced.

At $t = 5.5$, a QoS Violation event is detected by QoS Violation Detection due to the increase in average delay experienced by packets being served at the α-queue, as shown in Figure 5.5.2 (b). In this delay graph, D_{req} is drawn as a red horizontal line at 50 ms. From this graph, we observe that the peak average delay experienced by the connections at the α-queue is slightly above 40 ms but less than D_{req} at the time when a QoS Violation event is detected. This is due to two factors. The first is that the algorithm used to calculate delay in QoS Violation Detection is conservative, which over-estimates the average delay experienced by packets being transmitted. The second is that the value shown in the delay graph is the average delay of packets being transmitted in the MAC frame. As such, it takes 0.5 second for the number of packets at the α-queue to build up sufficiently to cause a QoS Violation event to be detected. The time taken for detection depends on the DQ bandwidth and the total offered load of the DQ traffic at the α-queue at that time.

Further, we observe that the MAC throughput of each connection being served

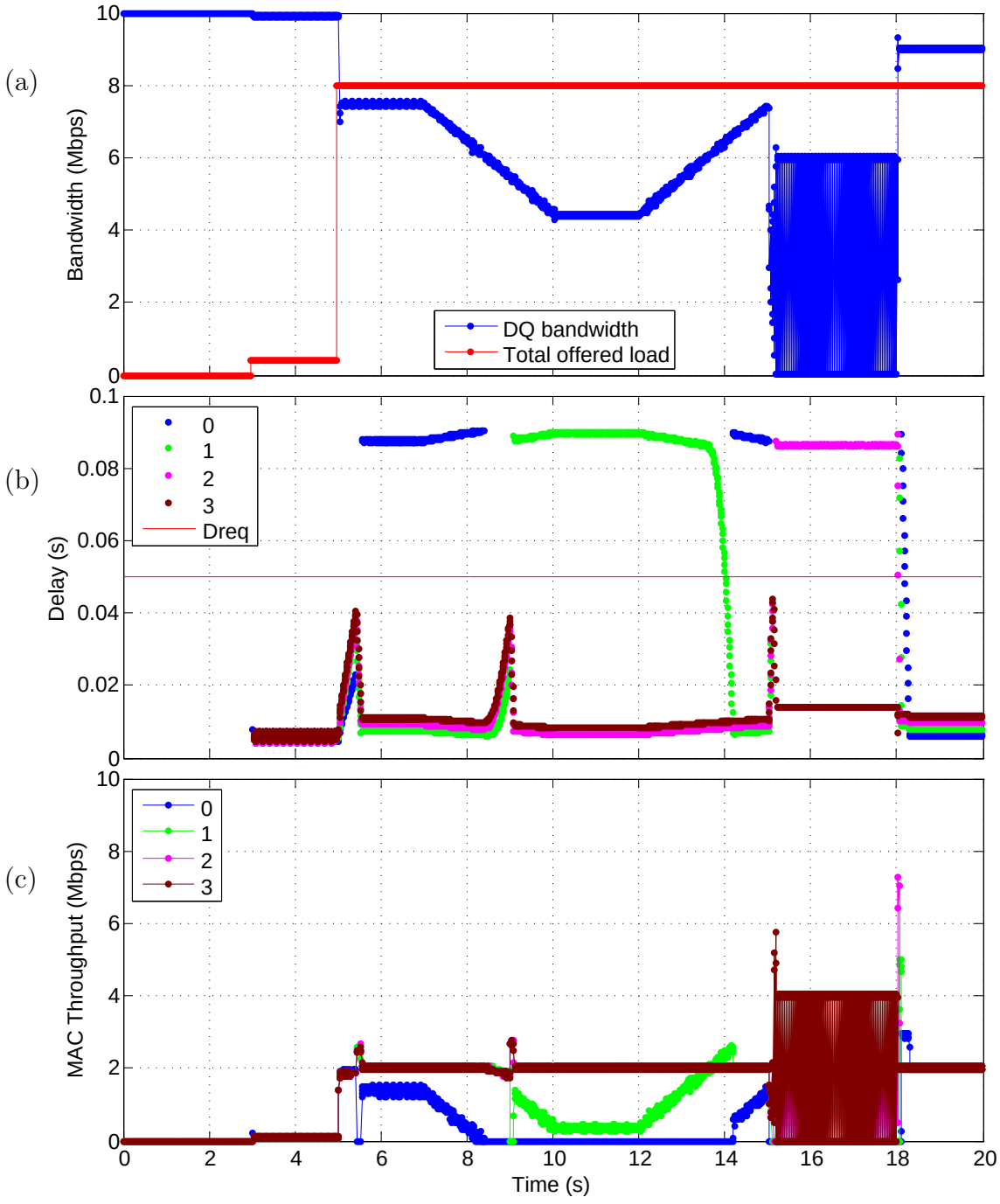

Figure 5.5.2: Experiment 1 (DLDQ): (a) DQ bandwidth, (b) packet delay, and (c) MAC throughput.

at the α-queue is slightly less than the offered load of 2 $Mbps$ at $t = 5$. This is because the DQ bandwidth is only equal to 7.5 $Mbps$. Given a DQ bandwidth of 7.5 $Mbps$, the DLDQ scheduler distributes this bandwidth equally among the four DQ connections at the α-queue according to the WFQ algorithm, giving each connection a MAC throughput of about 1.88 $Mbps$. This is shown in Figure 5.5.2 (c).

When a QoS Violation event is detected, a single connection is sent to the β-queue, as indicated in Figure 5.5.3 (b). Since all connections have the same PHY mode and connection 0 has the lowest index, it is chosen by the connection selection scheme (as detailed in Table 5.5.2) to be sent to the β-queue.

As soon as connection 0 is sent to the β-queue, there are only three remaining connections at the α-queue to share the DQ bandwidth of 7.5 $Mbps$. Hence, each of these connections is allocated with a bandwidth of 2.5 $Mbps$. Because of previous build up in the queues, they initially have enough packets to send and hence they achieve a MAC throughput of 2.5 $Mbps$ as shown in Figure 5.5.2 (c). At the same time, the delay experienced by these connections at the α-queue starts to decrease after connection 0 is sent to the β-queue, as their built up packets are being transmitted.

Until all the packets at the α-queue are transmitted, the β-queue is not served. As the β-queue is not served at all while waiting for the α-queue to transmit all built up packets, the delay experienced by the connection being moved from the α-queue to the β-queue jumps at $t = 5.5$.

When all built up packets at the α-queue are transmitted, the connections at the α-queue stabilise to a MAC throughput that equals their offered load. This is 2 $Mbps$ and connection 0 is allocated the remaining bandwidth after the α-queue is served, as shown in Figure 5.5.2 (c), which equals 1.5 $Mbps$. Since the MAC throughput of connection 0 does not reach its offered load, some packets from this connection are dropped by the EPD mechanism.

Given all the events explained above, the delay experienced by the packets being served at the α-queue is brought down to around 10 ms (D_{mac}). In the case of the

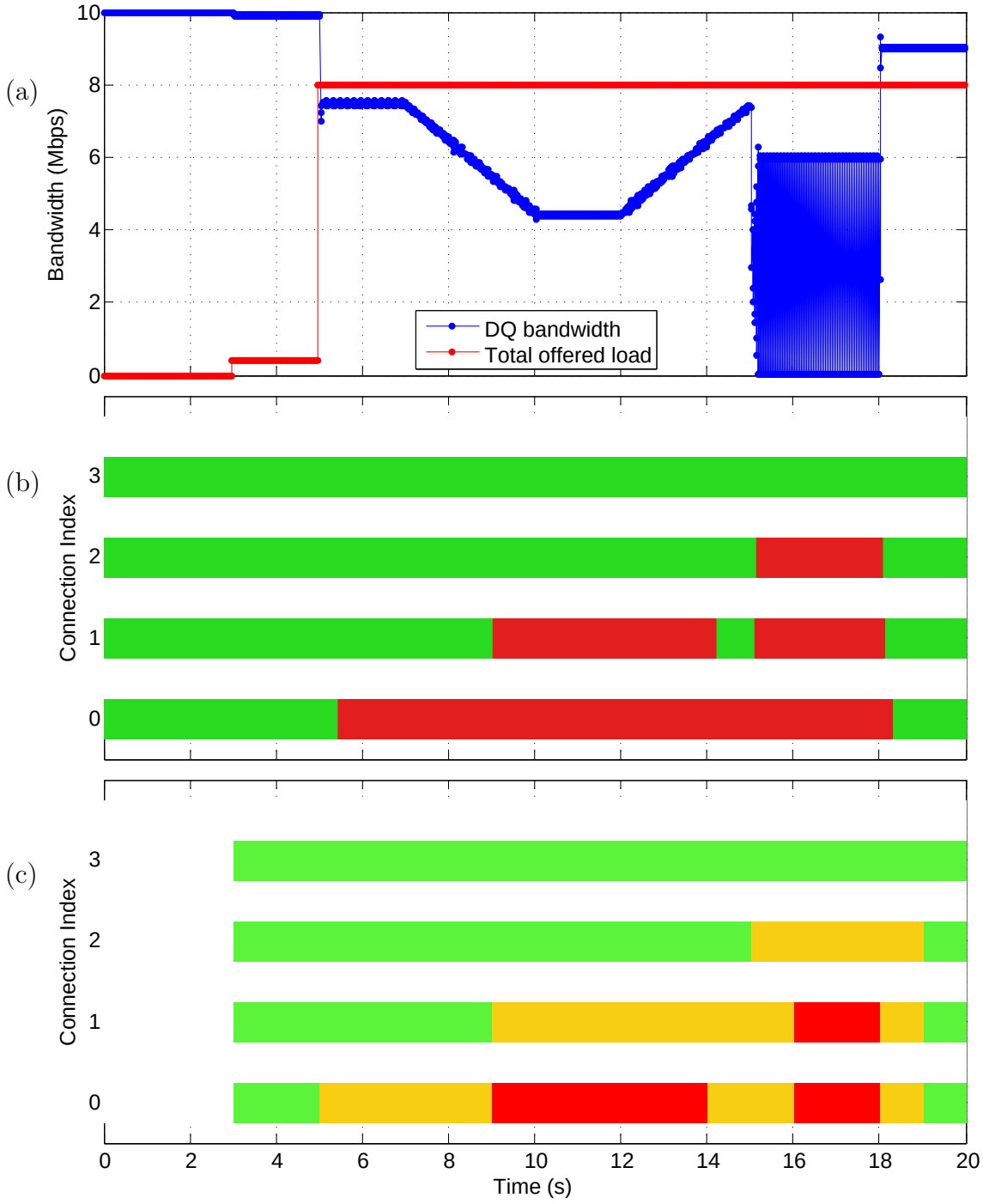

Figure 5.5.3: Experiment 1 (DLDQ): (a) DQ bandwidth, (b) connections at each queue; α-queue (green), β-queue (red), and (c) QoS received; good service (green), degraded service (amber) and no service (red).

connection being served at the β-queue, the delay is determined by D_{drop}. From Figure 5.5.2 (b), the maximum delay on the transmitted packets experienced by the connections being served at the β-queue is about 90 ms. Note that D_{drop} is, in fact, 100 ms, and the delay experienced is only 90 ms. This is due to the same reasons that the peak delay experienced by the packets transmitted is 40 ms instead of 50 ms when a QoS Violation event is detected, as mentioned earlier.

Due to the way the transmission ordering is specified in Table 5.5.2, a connection with a lower index is always scheduled ahead of a connection with a higher index if they belong to the same PHY burst. Since we are not considering different PHY modes in this experiment, there is only one PHY mode burst for data transmission. Therefore, we observe that the delay of a connection from the α-queue with a lower index is always slightly lower than a connection with a higher index.

In Figure 5.5.3 (c), we attempt to describe the QoS experienced by a customer. Hence, it is more suitable to show the QoS experienced by each connection in every second, rather than in every MAC frame. Furthermore, time intervals like 10 or 20 seconds are not chosen because we would like to investigate the agility of the DQ scheduler at managing the QoS of each connection. We refer to a connection as receiving good service during a one-second interval if there is no packet loss and none of the average packet delays calculated per MAC frame within the one-second interval exceeds D_{req}. If a connection experiences some packet loss (offered load $>$ MAC throughput > 0) or the average packet delay of at least one of the MAC frames within the one-second interval is greater than D_{req}, it is deemed to be receiving degraded service. Finally, if a connection is not being served at all during a one-second interval even though it has packets waiting for transmission (MAC throughput equals 0), we regard the connection as receiving "no service". We use green, amber and red to represent the times where a connection experiences good service, degraded service and 'no service respectively.

From this QoS graph, we observe that the connections being served at the α-queue receive good service. For the connection that is sent to the β-queue (connec-

tion 0), it receives degraded service as it experiences a delay which is more than the delay requirement (D_{req}) and some packet loss. Note that even though connection 0 is sent to the β-queue at $t = 5.5$, it is considered to be receiving degraded service during the time interval between $t = 5$ and $t = 6$ because its QoS is measured for every one-second interval.

Region 3 ($6 \leq t < 11$)

At $t = 6$, there are three connections in the α-queue, and each of these connections achieves a MAC throughput of 2 $Mbps$ and experiences a delay of about 10 ms, which is less than D_{req}. In the β-queue, one connection is served with a MAC throughput of 1.5 $Mbps$ and the average delay for the transmitted packets of this connection is 90 ms.

At $t = 7$, the DQ bandwidth starts to reduce linearly. This reduction in bandwidth is detected by the DLDQ scheduler. Hence, the bandwidth left to serve the β-queue is decreased, which causes a reduction in the MAC throughput of the connection being served at the β-queue, connection 0. The reduction in the MAC throughput of connection 0 matches exactly the reduction in the DQ bandwidth, as shown in Figures 5.5.2 (a) and 5.5.2 (c). However, the delay and the MAC throughput of the connections being served at the α-queue are not affected by this decrease in the DQ bandwidth, as long as the DQ bandwidth is sufficient to serve all the connections in the α-queue.

When the DQ bandwidth decreases to the stage where there is no bandwidth left to serve the β-queue (that is at $t = 8.4$, the MAC throughput of the connection served at the β-queue reaches zero), the same chain of events that began at $t = 5$ is repeated. That is, another QoS Violation event is detected, which is indicated by the delay peak at $t = 9$. Hence, connection 1 is sent to the β-queue as it is the connection with the lowest index at the α-queue.

After connection 1 is sent to the β-queue, there are two connections left in the β-queue. However, only connection 1 will be served if there is bandwidth left to

serve the β-queue. Connection 1 is chosen instead of connection 0 by the connection selection scheme because connection 1 has a higher index. The reduction in the MAC throughput of connection 1 also matches exactly the reduction in the DQ bandwidth until $t = 10$, where the DQ bandwidth remains constant.

Similar to the previous region, we observe that the connections being served at the α-queue receive good service, shown as green in Figure 5.5.3 (c). For the connections at the β-queue, they experience either degraded service or no service. Connections in the β-queue experience degraded service with only some packets served, shown in amber on Figure 5.5.3 (c). Connections in the β-queue that are not allocated any transmission slots, experience no service and this is depicted in red.

Region 4 ($11 \leq t < 15$)

At $t = 11$, there are two connections in the α-queue, and both connections achieve a MAC throughput of 2 $Mbps$ and experience a delay of about 10 ms, which is less than D_{req}. At the β-queue, one connection, connection 1, is served with a MAC throughput of 0.3 $Mbps$ and the average delay for the transmitted packets of this connection is 90 ms.

At $t = 12$, the DQ bandwidth begins to increase linearly. This increase in bandwidth is detected by the DLDQ scheduler. Hence, the bandwidth left to serve the β-queue is increased, which causes an increase in the MAC throughput of the single connection being served at the β-queue, which is connection 1. The increase in the MAC throughput of connection 1 matches exactly with the increase in the DQ bandwidth, as shown in Figures 5.5.2 (a) and 5.5.2 (c).

At $t = 13.7$, the MAC throughput of connection 1 begins to exceed its offered load since this connection has extra packets built up at the β-queue. The average delay experienced by the packets of this connection then starts to drop below 90 ms after its MAC throughput begins to exceed its offered load.

At $t = 14.2$, connection 1 is redirected back to the α-queue by QoS Recovery

Detection, as the queue for connection 1 is emptied. This event is evident in Figure 5.5.3 (b). As soon as this happens, another connection (connection 0) is chosen to be served at the β-queue in order to use the bandwidth left after serving the α-queue. Since the DQ bandwidth is still increasing after $t = 14.2$, the MAC throughput of connection 0 matches the increase in the DQ bandwidth.

Even though connection 1 is redirected back to the α-queue at $t = 14.2$, it is regarded as receiving degraded service from $t = 14$ to $t = 15$. In contrast, connection 0 is no longer regarded as receiving no service during the time interval between $t = 14$ to $t = 15$ as it starts to transmit at $t = 14.2$.

Region 5 ($15 \leq t < 18$)

Notice that the DQ bandwidth is expected to be constant at 3 $Mbps$, but it fluctuates between 6 and 0 $Mbps$ in this region. A closer look at the beginning of this region is shown in Figure 5.5.4.

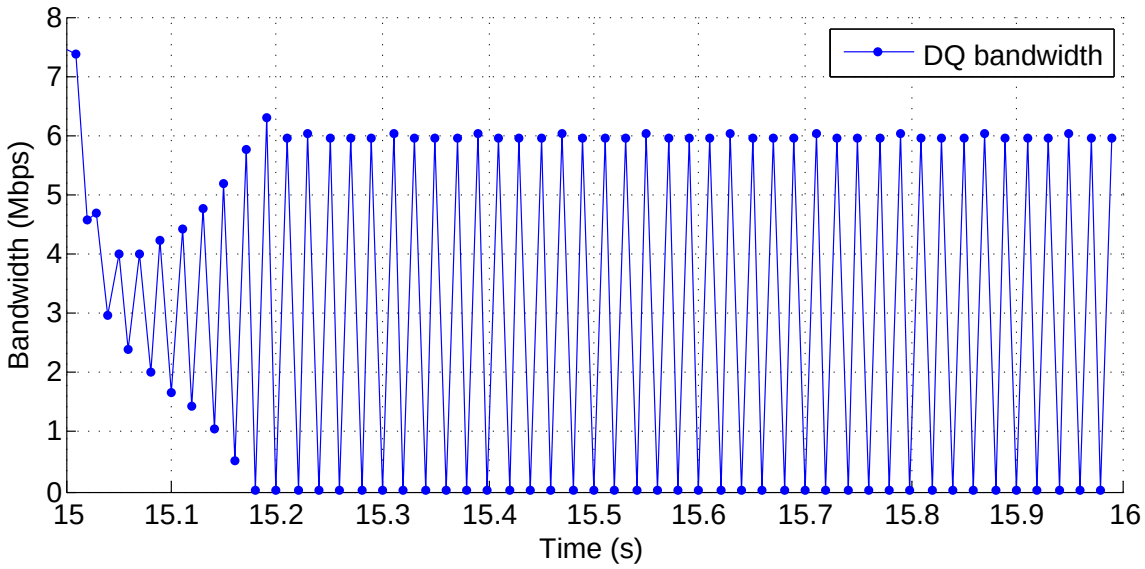

Figure 5.5.4: The DQ bandwidth from $t = 15$ to $t = 16$ in experiment 1 with the DLDQ scheduler.

This fluctuating behaviour of the DQ bandwidth is due to the UGS traffic. At

$t = 15$, the UGS traffic on the UL increases from 2.5 to 7 $Mbps$. Let us assume that the last MAC frame in the system that experiences a UGS rate of 2.5 $Mbps$ is MAC frame $n-1$. Therefore, at MAC frame n, the system experiences a new UGS rate of 7 $Mbps$.

Now, in Figure 5.5.5, we show the Bandwidth Request event in every MAC frame, which is carried out at the beginning of every UL sub-frame. In MAC frame $n-1$, there is more DL traffic than UL traffic, hence the DL sub-frame is longer than the UL sub-frame, and so the BR event in MAC frame $n-1$ is closer to the end of MAC frame $n-1$. Since the BR event in MAC frame $n-1$ is for scheduling the UL transmissions in MAC frame n, the proportion of the DL and the UL sub-frames as well as the time when the BR event is carried out in MAC frame n is similar to the one in MAC frame $n-1$.

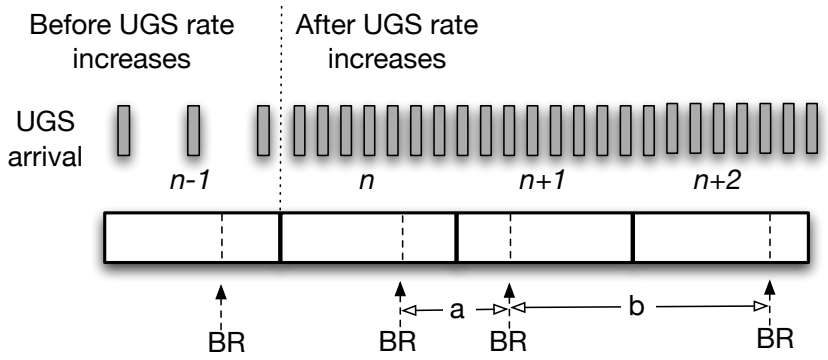

Figure 5.5.5: Diagram showing the uneven intervals between the BR events.

In MAC frame n, the system starts to experience an increase in the UL UGS traffic rate since more UGS traffic arrives at the queue. Hence, a lot of packets arrive at the UL queue between the start of MAC frame n and the BR event in MAC frame n, and so, the UL sub-frame in MAC frame $n+1$ is expected to be longer than the DL sub-frame. This pushes the BR event in MAC frame $n+1$ towards the start of MAC frame $n+1$. As a result, the BR event in MAC frame $n+1$ would report a smaller request than the average number of packets arriving in a MAC frame, since $a < D_{mac}$, where a is the time between the BR events in MAC frames n and $n+1$.

In contrast, due to the smaller BR request in MAC frame $n+1$, the BR event in MAC frame $n+2$ occurs at a time closer to the end of MAC frame $n+2$ and this results in a larger request than the average number of packets arriving in a MAC frame, since $b > D_{mac}$, where b is the time between BR events in MAC frames $n+1$ and $n+2$.

This behaviour builds up like a "snow ball" effect, which eventually causes all slots in a MAC frame being allocated to the UGS traffic, whilst in the next MAC frame, a relatively much smaller number of slots or even none are allocated to the UGS traffic. Therefore, this behaviour causes the DQ bandwidth being dropped to zero in a MAC frame, whilst in the next MAC frame, it increases to a relatively much larger value, as shown in Figure 5.5.4.

This behaviour causes a slight increase in the average delay of the UGS traffic, as shown in Figure 5.5.6. Further, it also causes a slight increase in the average delay experienced by the DQ (rtPS) traffic, as shown in Figure 5.5.2 (b). For the MAC throughput achieved by the DQ traffic, fluctuations similar to those observed in the MAC throughput of the UGS traffic occur, but in an opposite sense as the DQ traffic is served after serving the UGS traffic.

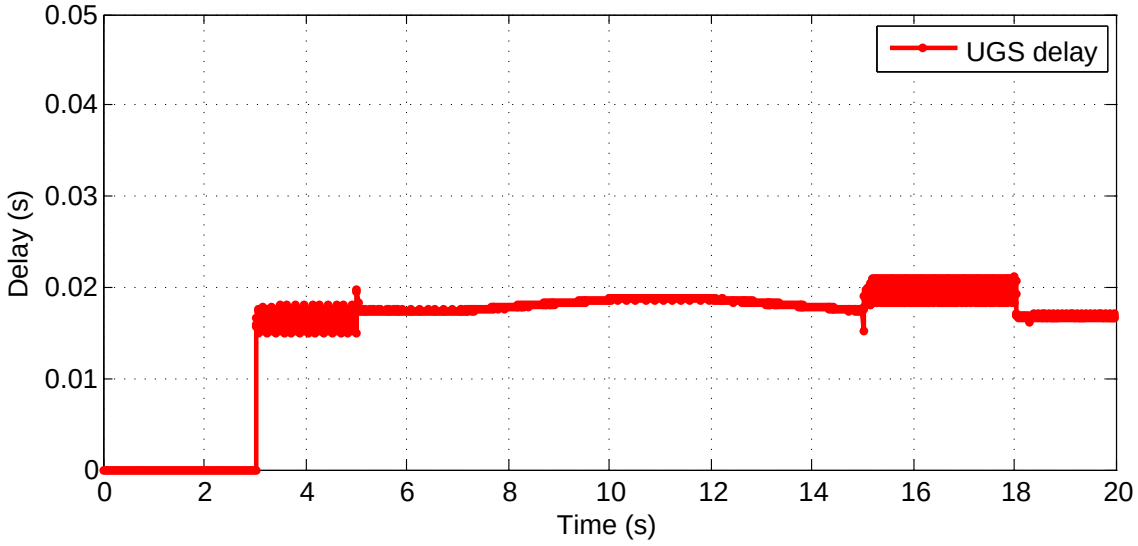

Figure 5.5.6: Delay experienced by the UGS traffic on the UL.

In fact, we plotted the MAC throughput of each connection over every MAC frame to show the way that the scheduler allocates slots in every MAC frame, which reveals the fluctuation artifact described above. In real applications, it may not be required to analyse the MAC throughput of an 802.16 system in every 10 ms interval. If we plot the MAC throughput over every two MAC frame intervals, the artifact described above would not be observed. We re-plot the DQ bandwidth and the MAC throughput of each connection measured over every two MAC frames in Figures 5.5.7 and 5.5.8. The artifact described above is no longer observed in these figures.

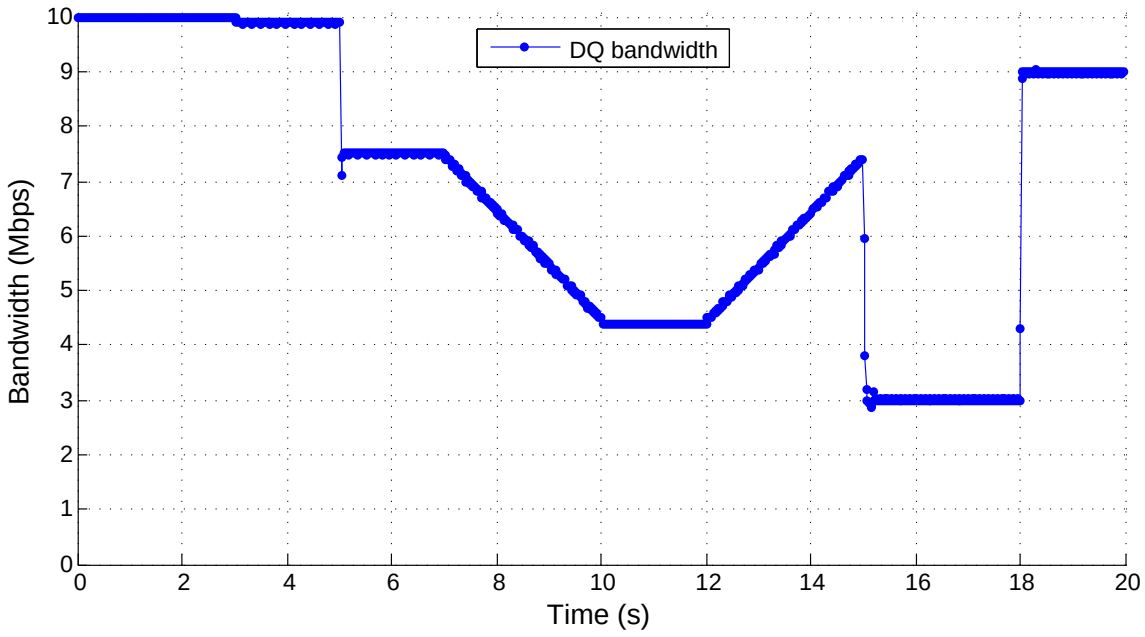

Figure 5.5.7: DQ bandwidth measured over intervals of two MAC frames in length.

Note that this fluctuation artifact happens when there is a sudden increase in the rate of the UL UGS traffic, which affects the time to carry out the BR mechanism in every MAC frame. During the time interval between $t = 7$ and $t = 10$, where the UGS rate is increased gradually, this artifact does not occur. This artifact is also due to the way we handle UGS traffic, where a strict priority rule applies, which allows UGS traffic to be allocated all the slots in the network. Since this artifact

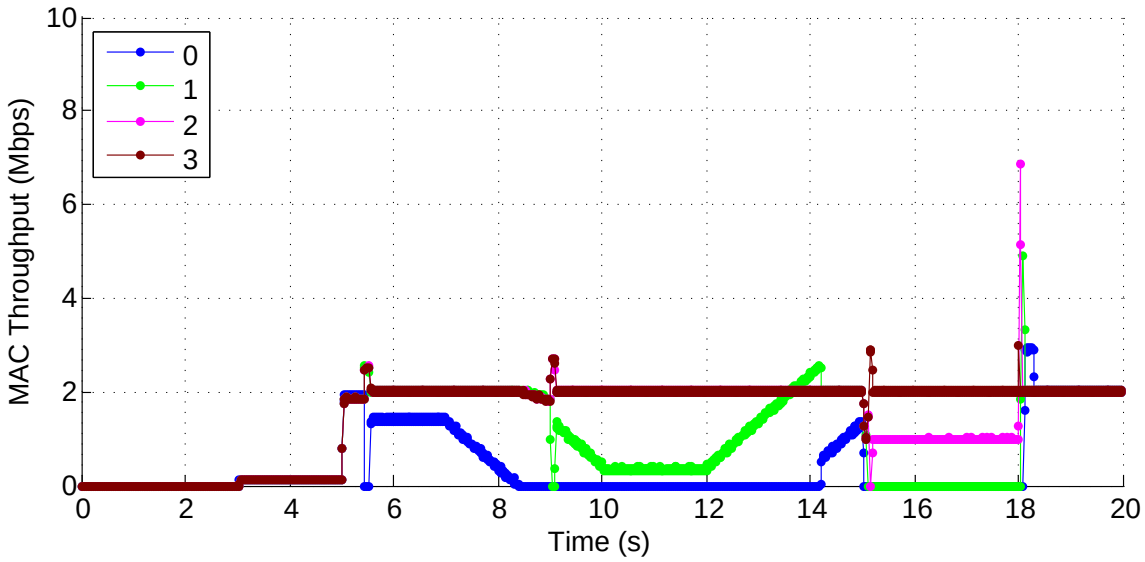

Figure 5.5.8: MAC throughput experienced by each DQ connection measured over intervals of two MAC frames in length.

stems from the UGS traffic, which is not handled by the DQ scheduler, we do not discuss it further. We now return to explaining the events that happen in the DQ scheduler.

At a time just before $t = 15$, there are three connections at the α-queue and one connection in the β-queue. At $t = 15$, the DQ bandwidth suddenly decreases to about 3 $Mbps$. This decrease in DQ bandwidth is detected by the DLDQ scheduler and therefore, the same chain of events that began at times 5 and 8.5 seconds are repeated (see Figure 5.5.8 for the MAC throughput of each connection). The only difference is the shorter time taken for QoS Violation Detection to detect a QoS Violation event. This is because the number of packets at the α-queue builds up faster than the previous cases due to the smaller DQ bandwidth, which causes a steeper increase in the delay experienced by the connections at the α-queue. Since a single connection is moved to the β-queue at any time, it takes some time for the DLDQ scheduler to move the correct number of connections from the α-queue to the β-queue, as we have explained in Section 5.3.3.

From the actual data, we can extract the MAC frame number when the QoS Violation event is detected, $F_{qv} = 1508$, which corresponds to $t = 15.08$. Due to this QoS Violation event, connection 1 is sent to the β-queue. Hence, the result of QoS Violation Detection is ignored after MAC frame 1508. Next, we can extract the MAC frame number where the result of QoS Violation Detection is no longer ignored, that is $F_{on} = 1512$. This agrees with equation (5.3.3) given in Section 5.3.3, where the result of QoS Violation Detection is ignored from MAC frame 1509 until 1511. At MAC frame $F_{on} = 1512$, another QoS Violation event is detected, which sends connection 2 to the β-queue.

From Figure 5.5.3 (c), connection 3 experiences good service at all times in this region. In fact, due to the artifact explained above, the QoS experienced by connection 3 would alternate between good service and no service if we considered the connection's QoS in every MAC frame as in every alternate frame it receives no transmissions at all. We contend that this has no impact on the end user because the delay experienced by connection 3 is still maintained below D_{req} at all times.

Region 6 ($18 \leq t \leq 20$)

At $t = 18$, the DQ bandwidth instantly increases to 9 $Mbps$. As a result, all the connections are redirected back to the α-queue (one at a time). Due to this sudden increase in bandwidth available to serve the DQ traffic, we observe some spikes in the MAC throughput of the connections, which is caused by the transmission of all the packets waiting at the queue. After these queued packets are transmitted, each DQ connection stabilises to a MAC throughput that equals its offered load, giving a total MAC throughput of 8 $Mbps$. Clearly, all these connections experience good service at this time. Notice that all connections redirected back to the α-queue are considered to have received degraded service for one second during the time interval between $t = 18$ and $t = 19$ because they experience degraded service in the first set of MAC frames after $t = 18$.

5.5.2 Experiment 2: Changing PHY mode

In this second experiment, we investigate the performance of the DLDQ scheduler in response to changing DQ bandwidth due to changes in PHY mode of the SSs. Similar to experiment 1, we have four SSs connected to a BS in an 802.16 network. Initially, each SS is assigned a particular PHY mode: SS 0 (22.5 $Mbps$), SS 1 (20.0 $Mbps$), SS 2 (15 $Mbps$) and SS 3 (10 $Mbps$). The PHY rate of each SS changes in this experiment as shown in Figure 5.5.9 (a).

Each SS carries a single DL rtPS connection generated by a CBR traffic source at a rate of 1 $Mbps$. The packet size of the CBR traffic source is 100 B (including 20 B of IP header). The total number of slots allocated to serve the DQ traffic in a MAC frame is held constant and is equal to the number of data slots in a MAC frame ($S_{mac} = 218$), as in Table 5.5.1.

The number of bytes of MAC layer data that can be transmitted in a slot for each PHY mode is shown in Table 5.4.1. From this table, we know that the average number of slots required to maintain a connection at a lower PHY mode is higher than one transmitted at a higher PHY mode. In this experiment, the average number of slots required in a MAC frame to transmit an offered load of 1 $Mbps$ for different PHY modes is presented in Table 5.5.3.

We note that the PHY rates 22.5 and 5 $Mbps$ have a ratio of 4.5, however the ratio of the number of slots required in a MAC frame to transmit an offered load of 1 $Mbps$ between the two PHY rates is about 7. Hence, the number of slots required to maintain a connection is not linearly proportional to the actual PHY rate of transmissions. This is due to the higher proportion of overheads in a slot for a lower PHY mode. Note also that the information in Table 5.5.3 is used for explanatory purposes only and is not available to the DLDQ scheduler as the offered load of a rtPS connection is not known to the DQ scheduler.

In this experiment, we use the number of slots to quantify DQ bandwidth in order to highlight the different amount of data carried in each slot using different

Table 5.5.3: Average number of slots required per MAC frame by a 1 *Mbps* connection under different PHY modes.

PHY rate (*Mbps*)	22.5	20.0	15.0	10.0	5.0
Slots required	22.2	25.2	35.2	57.5	156

PHY modes. For higher PHY modes, more data can be carried in each slot and hence, a lower number of slots is required to maintain a connection with a certain offered load. Further, we assume that a PHY mode change takes effect immediately before the scheduling process of every MAC frame. Hence, there is no issue with changing PHY mode during transmission periods within a MAC frame.

Experimental Regions

Similar to experiment 1, experiment 2 is carried out for 20 seconds and we divide this experiment into 4 regions in order to show the different DLDQ scheduling features with respect to changing PHY mode of the SSs.

- Region 1: $0 \leq t < 5$, where t is time in seconds. Similar to experiment 1, this region is used for the initialisation process in NS-2.

- Region 2: $5 \leq t < 6$. The offered load of each connection is increased to 1 *Mbps* and no congestion is experienced in this region because the DQ bandwidth is greater than the total offered load of the DQ traffic. That is, S_{mac} is greater that the total average number of slots required for all DQ connections.

- Region 3: $6 \leq t < 12$. Starting from $t = 7$, all connections drop their initial PHY rate to 5 *Mbps*, one at a time each second. This region is designed to investigate how the scheduler reacts to congestion due to PHY mode changes in the network.

- Region 4: $12 \leq t \leq 20$. Starting from $t = 14$, all connections change their PHY mode back to their initial PHY rate, one at a time each second. This region is designed to investigate how the scheduler recovers from congestion due to PHY mode changes.

Region 1 $(0 \leq t < 5)$

Similar to experiment 1, the first 5 seconds is reserved for the initialisation process. The offered load of each connection is 0.1 $Mbps$.

Region 2 $(5 \leq t < 6)$

At $t = 5$, the offered load of each connection is increased to 1 $Mbps$ and it is maintained at this constant rate throughout the remainder of the experiment. Initially, all the SSs operate at their respective PHY modes and all four connections are being served at the α-queue. From Figure 5.5.9 (b), we observe that each connection achieves an average delay that is less than the delay requirement (D_{req}). Further, each connection achieves a MAC throughput that equals its offered load of 1 $Mbps$ as shown in Figure 5.5.9 (c). Therefore, all connections experience good service when they are operating at their initial PHY mode.

Using Table 5.5.3, we calculate the total average number of slots required in a MAC frame by all connections, which is 140, and it is less than S_{mac}. Therefore, the DLDQ scheduler manages to accommodate all four connections at the α-queue.

Due to the way that the 802.16 system is specified in the standard, a connection with a lower PHY mode is always scheduled to transmit ahead of a connection with a higher PHY mode. When there are two or more connections with the same PHY mode, the connection with the lowest index will be transmitted ahead of connections with a higher index. From Figure 5.5.9 (b), we observe that the average delay of a connection with a lower PHY mode is always less than a connection with a higher PHY mode.

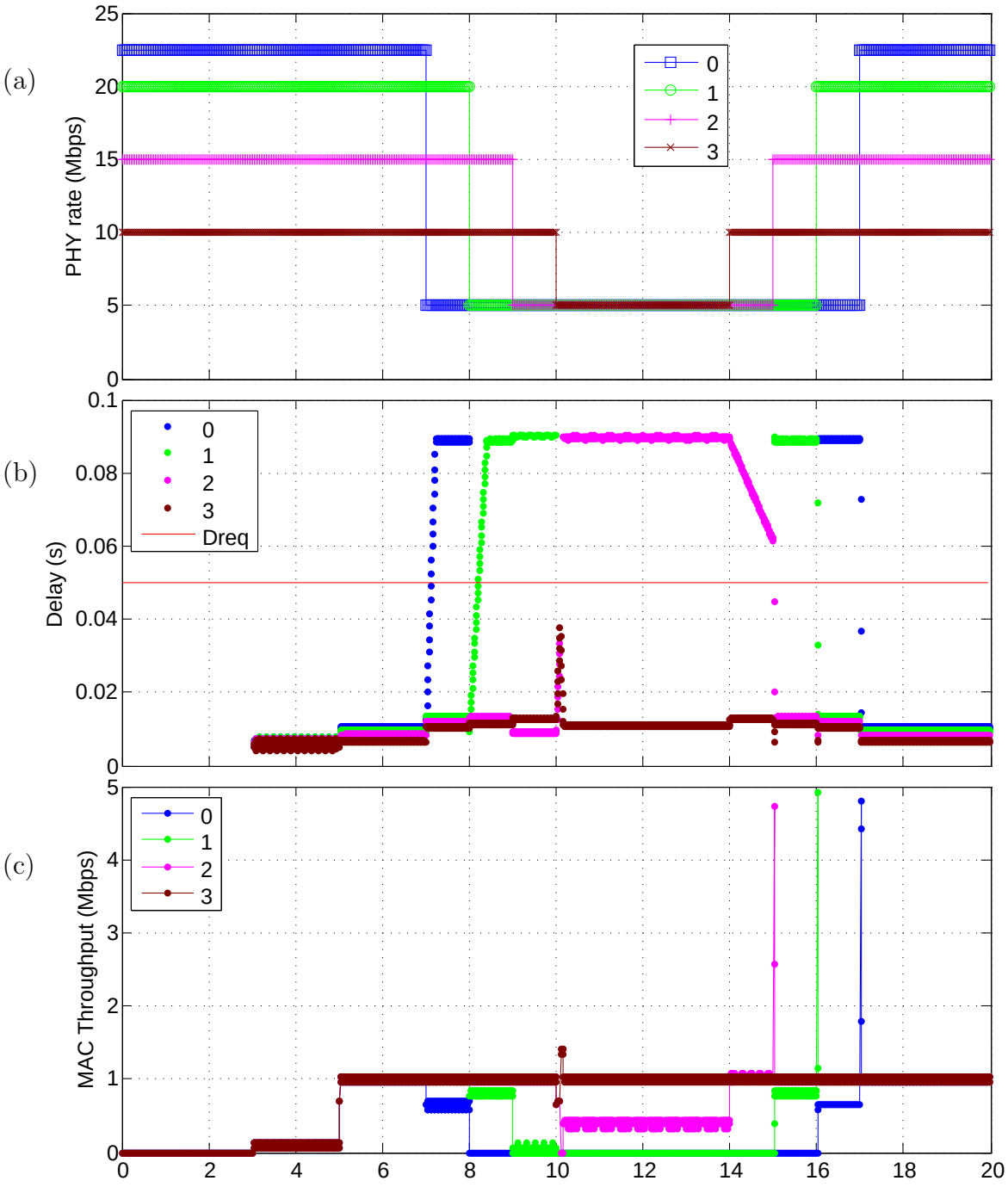

Figure 5.5.9: Experiment 2 (DLDQ): (a) PHY rate, (b) packet delay, and (c) MAC throughput.

Region 3 ($6 \leq t \leq 12$)

At $t = 6$, all four connections are served at the α-queue and experience good service. Each of these connections achieves a MAC throughput of 1 $Mbps$ and experiences an average delay between 6 and 10 ms, depending on its PHY mode.

At $t = 7$, connection 0 changes PHY mode from 22.5 to 5 $Mbps$. When this happens, an increase in the average delay experienced by all the connections is observed at the α-queue. This causes a QoS Violation event soon after connection 0 changes PHY mode. In fact, the increase in delay is markedly seen in connection 0 because of the objective that maximises the MAC throughput of the network. A connection with a higher PHY mode is given more transmission opportunities than a connection with a lower PHY mode. Therefore, connection 0 is allocated slots based on the number of slots left after serving the other connections at the α-queue as it has the lowest PHY mode. From Figure 5.5.9 (c), the MAC throughput of connection 0 drops below its offered load as soon as it changes PHY mode; the MAC throughput of the other connections at the α-queue is not affected. When the QoS Violation event is detected, connection 0 is sent to the β-queue as it has the lowest PHY mode at that time, as shown in Figures 5.5.10 (a) and 5.5.10 (b). At the β-queue, connection 0 is served based on the number of slots left after serving the α-queue.

Let us consider the total average number of slots required in a MAC frame by all the connections at the α-queue and the β-queue after connection 0 changes PHY mode. Since connection 0 changes PHY mode from a high to a low transmission rate, the average number of slots required to support connection 0 has increased from 22.2 to 156. Therefore, the total average number of slots required in a MAC frame to support all the connections equals, $156 + 25.2 + 35.2 + 57.5 = 274$, which is greater than S_{mac}. This explains why the QoS Violation event was detected by the DLDQ scheduler so soon after connection 0 changes PHY mode.

At $t = 8$, connection 1 changes PHY mode from 20 to 5 $Mbps$. The same chain

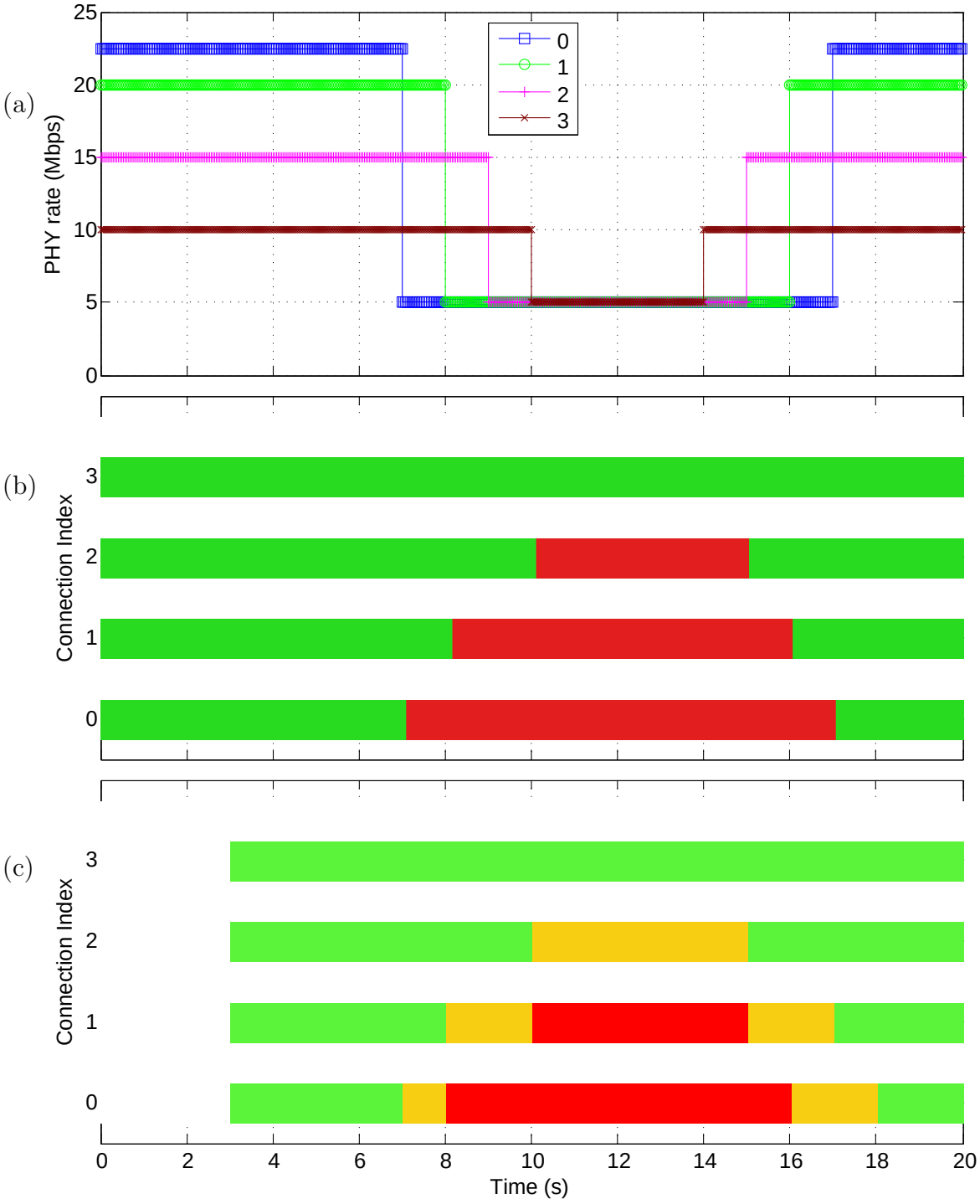

Figure 5.5.10: Experiment 2 (DLDQ): (a) PHY rate, (b) connections at each queue; α-queue (green), β-queue (red), and (c) QoS received; good service (green), degraded service (amber) and no service (red).

of events that began at $t = 7$ are repeated. As a result, a QoS Violation event is detected because there are insufficient slots to accommodate all the connections at the α-queue, which causes connection 1 to be sent to the β-queue.

At this stage, two connections remain at the α-queue and two connections have been sent to the β-queue, where the connection with the highest index (connection 1) at the β-queue is served first, receiving the number of slots left after serving the α-queue.

At $t = 9$, connection 2 changes PHY mode from 15 to 5 $Mbps$. Even though there is an increase in the number of slots required by connection 2, there is no increase in average delay experienced by the connections at the α-queue, as observed from Figure 5.5.9 (b) due to the fact that there are sufficient slots to serve the α-queue. However, the MAC throughput of the connection being served at the β-queue (connection 1) has been reduced, as shown in Figure 5.5.9 (c). This reflects the fact that the increase in the average number of slots required by connection 2 has caused a reduction in the number of slots left to serve the β-queue. Since the available slots is sufficient to accommodate the two connections at the α-queue, a QoS Violation event does not occur due to this PHY mode change.

At $t = 10$, connection 3 changes PHY mode from 10 to 5 $Mbps$. At this time, all the connections at the α-queue and the β-queue have the same PHY mode. An increase in the average delay experienced by both connections at the α-queue is observed, which eventually causes a QoS Violation event to be detected. Note that as soon as connection 3 changes PHY mode, the MAC throughput of both connections being served at the α-queue drops below their offered load. This is due to the fact that there are insufficient slots to serve the two connections with same PHY mode at the α-queue. If these two connections were operating at different PHY modes, only the MAC throughput of the lower PHY mode connection would be affected due to the same explanation for the events that happened at $t = 7$ and $t = 8$.

After connection 2 is sent to the β-queue following the detection of a QoS Vio-

lation event, the MAC throughput of the connection being served at the α-queue, connection 3, temporarily achieves a rate of ($1 \times S_{mac}/156 = 1.4\ Mbps$) which is above its offered load due to the transmission of all the queued packets, before settling back to its offered load of 1 $Mbps$. For the connections at the β-queue, only one connection, connection 2, is served after all the packets at the α-queue are transmitted.

From Figure 5.5.9 (c), we observe that the connections being served at the α-queue receive good service at all times, unless it has just been redirected back to the α-queue. For the connections being served at the β-queue, they either receive degraded service or no service.

Region 4 ($12 \leq t \leq 20$)

At $t = 12$, there is a single connection being served at the α-queue which stabilises to a MAC throughput of 1 $Mbps$. At the β-queue, one of the connections, connection 2, is being served with a MAC throughput of 0.4 $Mbps$.

At $t = 14$, connection 3 changes its PHY mode back to its initial PHY mode of 10 $Mbps$. When this happens, a reduction in the average delay and an increase in MAC throughput of the connection being served at the β-queue, connection 2, are observed because there are more available slots to serve the β-queue. During the time interval between $t = 9$ and $t = 10$, we know that the α-queue was able to accommodate both connection 2 (PHY rate = 5 $Mbps$) and connection 3 (PHY rate = 10 $Mbps$). However, due to the queued packets of connection 2 at the β-queue, it takes some seconds for connection 2 to transmit all these packets before it can be redirected back to the α-queue.

From Figures 5.5.9 (b) and 5.5.9 (c), we observe that the delay experienced by connection 2 begins to decrease after connection 3 changes PHY mode, and Connection 2 achieves a MAC throughput which is very slightly above its offered load.

At $t = 15$, connection 2 changes its PHY mode back to its initial PHY mode

of 15 *Mbps*. As a result, there are more available slots to transmit all the queued packets of connection 2 at the β-queue. Note that there is a spike in the MAC throughput of connection 2 and as soon as all the queued packets of this connection are transmitted, it is redirected back to the α-queue, as shown in Figure 5.5.10 (b). After connection 2 is redirected back to the α-queue, and its MAC throughput has stabilised at 1 *Mbps*, another connection from the β-queue, connection 1, is chosen to be served based on the number of slots left after serving the α-queue.

At $t = 16$ and $t = 17$, the same chain of events that began at both $t = 14$ and $t = 15$ is repeated. Eventually, all connections are redirected back to the α-queue. The same delay characteristics observed during the time interval between 5 and 7 seconds are observed again after $t = 17$, where the delay of a connection with a lower PHY mode is always less than that of a connection with a higher PHY mode.

5.5.3 Summary

From the results of experiments 1 and 2, we have demonstrated that our proposed DLDQ scheduler effectively manages congestion using the QoS Violation Detection and QoS Recovery Detection mechanisms. These were triggered by the changes in both the number of slots available to serve the DQ traffic and the PHY mode of the SSs. The QoS Violation Detection and QoS Recovery Detection mechanisms are able to achieve their objective to maximise the number of connections at the α-queue, or equivalently the number of connections that receive good service at all times.

From Figures 5.5.3 (c) and 5.5.10 (c), we observe that connection 3 experiences good service throughout the experiment because it has the highest priority. In fact, a connection with a higher priority experiences more good service intervals than connections with lower priorities due to the connection selection schemes chosen in our experiments. When a QoS Violation event is detected, the connection with the lowest priority from the α-queue is sent to the β-queue, which gives more transmission

opportunities to connections with a higher priority. Similarly, when there is bandwidth or slots left after serving the α-queue, the connection with the highest priority at the β-queue will be served, which again gives more transmission opportunities to connections with a higher priority.

Therefore, with the deployed connection selection schemes, the DLDQ scheduler manages to achieve the goal of these schemes, which gives more transmission opportunities to connections with higher priorities. As a result, the DLDQ scheduler is able to maximise the number of connections who never receive anything other than good service or receive good service as much as possible. Furthermore, the secondary objective to maximise throughput is also achieved.

5.6 Core Uplink Dual-Queue Mechanisms

In this section, we propose the core DQ mechanisms for the ULDQ scheduler for handling real-time services in 802.16 systems. As discussed earlier, the ULDQ scheduling is more complicated than the DLDQ scheduling, however, the proposal for the DLDQ scheduler has solved some common issues for the ULDQ scheduler.

5.6.1 Connection Prioritisation

Similar to the DL, the connection prioritisation mechanism is based on the objectives defined for the 802.16 system and tie-breaker algorithms are needed to assign a unique priority to each connection in the system.

5.6.2 QoS Violation Detection

Since the time-stamping or frame-stamping information for UL packets is not available to the Base Station, we can not apply the same DL technique to UL QoS Violation Detection. Therefore, we propose the following technique for UL QoS Violation Detection.

Information Available

First of all, we investigate the BR mechanism for the BS to collect UL information from the SSs. As discussed in Chapter 2, the 802.16 MAC frame can be divided into DL and UL sub-frames, where the DL sub-frame is scheduled prior to the UL sub-frame, and the BR mechanism provides the request for bandwidth at the beginning of every UL sub-frame, as shown in Figure 5.6.1. We assume that the BS obtains and updates the SSs UL transmission request information on a per MAC frame basis.

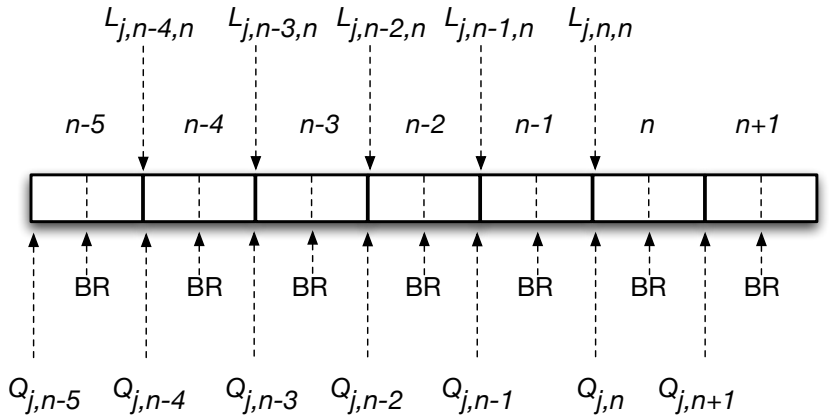

Figure 5.6.1: The timing diagram of events across multiple MAC frames.

We now define some terms used to describe the QoS Violation Detection algorithm for ULDQ scheduling.

- $Q_{j,n}$: The number of slots of data waiting for UL transmission at SS j in MAC frame n, where this information is obtained from the BR mechanism in MAC frame $n-1$. Clearly, $Q_{j,n} \geq 0$.

- $T_{j,n}$: The number of slots of data scheduled for UL transmission for SS j in MAC frame n, where $T_{j,n} \geq 0$. Note that $T_{j,n} \leq Q_{j,n}$, $\forall j, n$.

- $L_{j,k,n}$: The number of slots of data for SS j that have arrived in the FIFO queue before the start of MAC frame k, but are still not transmitted by the

end of MAC frame n, where $k \leq n$ and $L_{j,k,n} \geq 0$.

In Chapter 2, we explained that the BS schedules the DL and the UL transmissions at the beginning of a MAC frame. Furthermore, it transmits the DL-MAP and the UL-MAP that describe the bandwidth allocation and usage of the current MAC frame to all the SSs before the beginning of the DL sub-frame. Therefore, the UL request information obtained from the BR mechanism of the current MAC frame is used for scheduling the UL transmissions in the next MAC frame.

From the 802.16 standard, the SSs UL request information is transmitted to the BS in terms of the number of bytes of data waiting for UL transmission. When BR information arrives at the BS, the information in terms of the number of bytes is converted to a number of slots. From Figure 5.6.1, the UL information for SS j obtained from the BR mechanism in MAC frame $n-1$ is represented as $Q_{j,n}$, as this information is used for the UL scheduling of MAC frame n.

Scheduling Relationship

We assume that the real-time service queues in each of the SSs are served in FIFO order. Hence, the number of slots of data at a SS that existed before MAC frame k but are still waiting for transmission at the end of MAC frame n, is equal to the number of slots of data waiting for scheduling in MAC frame k, minus the number of slots of data scheduled for UL transmission from MAC frame k up to MAC frame n. Therefore, we have that

$$L_{j,k,n} = \Big[Q_{j,k} - \sum_{h=k}^{n} T_{j,h}\Big]^{+}, \qquad \forall k \leq n, \forall j, \tag{5.6.1}$$

where

$$[A]^{+} = \begin{cases} A, & \text{if } A \geq 0, \\ 0, & \text{if } A \leq 0. \end{cases}$$

In Figure 5.6.2, we show the actual size of the queue for SS j at the end of the MAC frame $n-1$, where $L_{j,n-i,n} = 0$ for some $i > 0$.

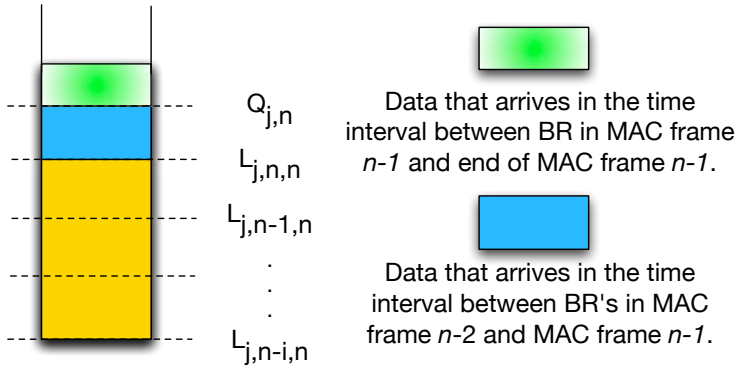

Figure 5.6.2: Snapshot of the queue at the end of MAC frame *n-1*.

Assume that the current MAC frame is n as shown in Figure 5.6.2. Based on equation (5.6.1), we note the obvious property,

$$L_{j,k',n} \leq L_{j,k,n}, \qquad \forall k' \leq k. \tag{5.6.2}$$

Delay Indication from $L_{j,k,n}$

Let us consider an example where the current MAC frame is MAC frame n. Assume $L_{j,n-i,n} = P > 0$ and $L_{j,n-(i+1),n} = 0$, hence, $L_{j,k',n} = 0$, $\forall k' < n-(i+1)$. In other words, there are P slots of data for SS j that have arrived in the queue before the start of MAC frame $n-i$, but have not been transmitted by the end of MAC frame n. This data is also the oldest data in the queue waiting for UL transmission and the earliest that these P slots of data will be transmitted would be in the UL sub-frame of the next MAC frame $(n+1)$.

We define $D_{j,n+1}$ as the delay of the oldest data at the queue being transmitted for SS j in MAC frame $n+1$, which consists of the P slots of data. Since $L_{j,n-i,n}$ is calculated from $Q_{j,n-i}$, which is obtained from the BR mechanism carried out in MAC frame $n-(i+1)$, the P slots of data must have arrived at the queue in the time interval between the BR events carried out in MAC frames $n-(i+2)$ and $n-(i+1)$. Therefore, we can calculate the maximum and minimum of $D_{j,n+1}$. The

maximum of $D_{j,n+1}$ will be obtained if the P slots of data arrived at the start of MAC frame $n-(i+2)$, which is equal to the sum of the duration of MAC frame $n-(i+2)$ up to MAC frame $n+1$. The minimum of $D_{j,n+1}$ will be obtained if the P slots of data arrived at the end of MAC frame $n-(i+1)$, which is equal to the sum of duration of MAC frame $n-i$ up to MAC frame $n+1$. Note that the explanation above is only possible assuming a DL or UL sub-frame can take up a full MAC frame in Adaptive TDD mode. Hence,

$$D_{j,n+1} = d_{n,i} \times D_{mac}. \tag{5.6.3}$$

where

$$i+2 \leq d_{n,i} \leq i+4.$$

Therefore, by monitoring $L_{j,k,n}$ in every MAC frame, we can calculate the delay of the data served by the queues within the SSs.

Algorithm for Detection of QoS Violations

Based on equations (5.6.2) and (5.6.3), we can construct a QoS Violation Detection algorithm for the UL. A QoS Violation event is detected when,

$$L_{j,n-i,n} > 0, \tag{5.6.4}$$

where

$$D_{j,n+1} > D_{req}.$$

In this thesis, we choose $d_{n,i} = i+4$ as this choice is more conservative. With this choice, it is possible that a QoS Violation event is detected early, but it may also detect a false QoS Violation event when there is only a short traffic burst in the network. If we chose $d_{n,i} = i+2$, it would reduce the possibility of false detection at the risk of missing QoS Violation events until after they have actually occurred. Therefore, there is a trade-off between the speed of detection and the rates of false detection. For example, if $D_{req} = 50\ ms$ and $D_{mac} = 10\ ms$, a QoS Violation event is detected when $L_{j,n-i,n} > 0$, where $i = 2$, based on $d_{n,i} = i+4$.

When a QoS Violation event is detected in MAC frame n, a single connection will be sent to the β-queue and the scheduler will recalculate $T_{j,n}$, for all remaining connections in the α-queue and re-allocate slots to the α-queue based on these values. This approach allows the movement of a connection to take effect as soon as a QoS Violation event is detected.

5.6.3 Response to QoS Violation Detection

Moving a connection to the β-queue

Similar to the DL, a single active connection from the α-queue will be chosen and sent to the β-queue based on certain connection selection criteria when a QoS Violation event is detected in a MAC frame. These connection selection criteria are also formulated based on the objectives defined for the system. However, since the α-queue and the β-queue are not storing actual packets, we simply mark the connection to be sent to the β-queue. Hence, it is not served starting from the MAC frame that the QoS Violation event was detected. The bandwidth request information from this connection is now stored in the β-queue.

How long should we wait before moving another connection?

Similar to the DLDQ scheduler, the ULDQ scheduler needs to allow enough time for the removal of a connection from the α-queue to take effect without moving extra connections to the β-queue due to the same QoS Violation event. The same algorithm to calculate the amount of time to wait before another connection is implemented to the ULDQ scheduler.

Note that the processes for QoS Violation Detection must be carried out every MAC frame even though the response to QoS Violation Detection has been turned off after a QoS Violation event is detected. This is because QoS Violation Detection on the UL is relying on transmission information in every MAC frame to update the QoS Violation Detection algorithms. In contrast, QoS Violation Detection in the

DL only relies on time-stamping information, and hence it can be totally turned off and resumed at anytime.

Since there are no actual packets for marking in the BS for the UL traffic when a QoS Violation event is detected in MAC frame F_{qv}, the ULDQ scheduler stores the total amount of bandwidth requested in terms of the number of slots of all UL connections at the α-queue. If this number of slots is completely allocated at a MAC frame before F_{on}, where $[(F_{on} - F_{qv}) + 1] \times D_{mac} = D_{req}$, the result of QoS Violation Detection is no longer ignored. Hence, this process also incurs a minimum waiting time between moving another connection to the β-queue after the previous connection is moved to the β-queue, equal to D_{req}, since the result of QoS Violation Detection is ignored until F_{on}.

5.6.4 QoS Recovery Detection

Similar to the DL, QoS Recovery Detection is carried out to check if there is sufficient bandwidth available for the α-queue to accommodate a connection from the β-queue. We also propose to allocate all available slots to serve the β-queue to a specifically chosen UL connection from the β-queue in every MAC frame. This connection is also chosen based on certain connection selection criteria in order to achieve the objectives defined for the system.

If there are slots left over from serving this connection, the remaining slots will be allocated to another specifically chosen connection based on the same connection selection criteria. This process is repeated until all the slots left to serve the β-queue are assigned to connections at the β-queue.

Similar to the DL, QoS Recovery Detection for the UL detects whether a connection being served from the β-queue exhausts its queue, or in other words, has no packets to send. Furthermore, we only apply QoS Recovery Detection to connections that are allocated transmission slots at the β-queue in that MAC frame to prevent moving a connection that has had all its packets removed by the EPD mechanism.

5.6.5 Response to QoS Recovery Detection

Similar to the DL, when QoS Recovery Detection indicates that the α-queue can accommodate another connection, the connection being served at the β-queue that has exhausted its queue will be redirected back to the α-queue.

5.6.6 Explicit Packet Dropping

Similar to the DLDQ scheduler, we apply an EPD mechanism to the α-queue and the β-queue of the UL to remove packets that have been waiting for more than D_{drop} at the queue. However, since the BS only stores the UL bandwidth request information of the SSs, this process is not carried out in the BS, but in the SSs.

We propose the same EPD mechanism be carried out on the UL as is on the DL. Therefore, each packet that arrives at the MAC layer of the SSs is tagged with the current MAC frame number. A packet is dropped if it has been waiting at the queue for more than D_{drop}.

However, there is a concern about the impact of the EPD mechanism on the UL QoS Violation Detection because it may introduce errors to the UL QoS Violation Detection algorithms. In an error-free environment, the UL QoS Violation Detection algorithm manages to keep track of the number of slots requested and the number of slots allocated in every MAC frame. When the EPD mechanism drops a packet, from the head of the queue, this causes a drop in the number of slots requested. This may also cause the ULDQ scheduler at the BS to falsely assume that less data has arrived at the tail of the queue. This would introduce errors into UL QoS Violation Detection.

Since UL QoS Violation Detection records the number of slots with different delays at the α-queue in terms of multiples of D_{mac}, the BS is able to anticipate the dropping of packets from a SS based on the delay information obtained from QoS Violation Detection. Therefore, the BS is able to correctly account for the effects of the EPD mechanism.

Let us consider an example as shown in Figure 5.6.3, where each bar represents a data slot waiting for transmission and $L_{j,n-8,n} = 2 > 0$. From the QoS Violation Detection algorithm, we can calculate the waiting time of the $L_{j,n-8,n} = 2$ slots of data, say $d \times D_{mac}$. Hence, when $d \times D_{mac} > D_{drop}$, the BS knows that SS j is dropping the packets that experience delay more than D_{drop}. Since packet dropping is expected in SS j, the BS can explicitly set $L_{j,n-8,n} = 0$ to correctly account for the effects of the EPD mechanism. Only after this correction is carried out does the BS perform UL QoS Violation Detection.

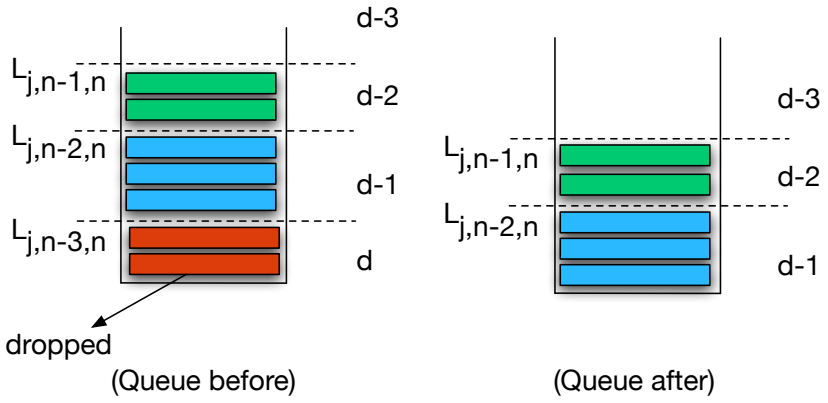

Figure 5.6.3: A snapshot of a queue when the EPD mechanism drops packets.

5.7 Uplink Experiments

We carry out the same experiments on the UL as were performed on the DL described in Section 5.5. Now, the DQ traffic is the UL traffic and the UGS traffic is a DL connection. We assume that the BR information of all UL connections is sent to the BS on a per MAC frame basis, which corresponds to a BR polling interval that equals D_{mac}.

Based on equation (5.6.4) of Section 5.6 for UL QoS Violation Detection, a QoS Violation event is detected when $L_{j,n-i,n} > 0$, where $i = 2$, given $D_{mac} = 10\ ms$ and $D_{req} = 50\ ms$. As mentioned in Section 5.4, we set the transmission ordering

for the UL to start from connections with the lowest index to connections with the highest index.

We compare the following experiments to experiments 1 and 2 carried out in Section 5.5 to investigate the DLDQ scheduler.

5.7.1 Experiment 1: Changing number of slots to serve DQ traffic

From Figure 5.7.1 (b), we observe that QoS Violation events are detected due to the same chains of events as those described in experiment 1 of Section 5.5. The maximum average delay experienced by the connections being served at the α-queue before the detection of a QoS Violation event is also similar to the one described in experiment 1 of Section 5.5, even though the QoS Violation Detection algorithms used are not the same.

However, the average delay experienced by the UL connections at the α-queue during uncongested periods is D_{mac} higher than the average delay experienced by the DL connections at the α-queue described in experiment 1 of Section 5.5. This is due to the BR mechanism carried out by the UL connections, which incurs an additional delay equal to the BR polling interval, which is D_{mac}.

In the case of the UL connections being served at the β-queue, their average delay is slightly more than the average delay experienced by the DL connections being served at the β-queue described in experiment 1 of Section 5.5. This can be explained by referring to Figure 5.7.2. In Figure 5.7.2, the MAC frame arrangement of this experiment and experiment 1 described in Section 5.5 is shown, where yellow represents the overheads of a MAC frame, blue represents the UGS traffic and white represents the DQ traffic.

As we have seen earlier, the transmission ordering of connections within a MAC frame has some impact on the delay experienced by the connections. That is, the average delay of the connections being consistently transmitted near the end of every

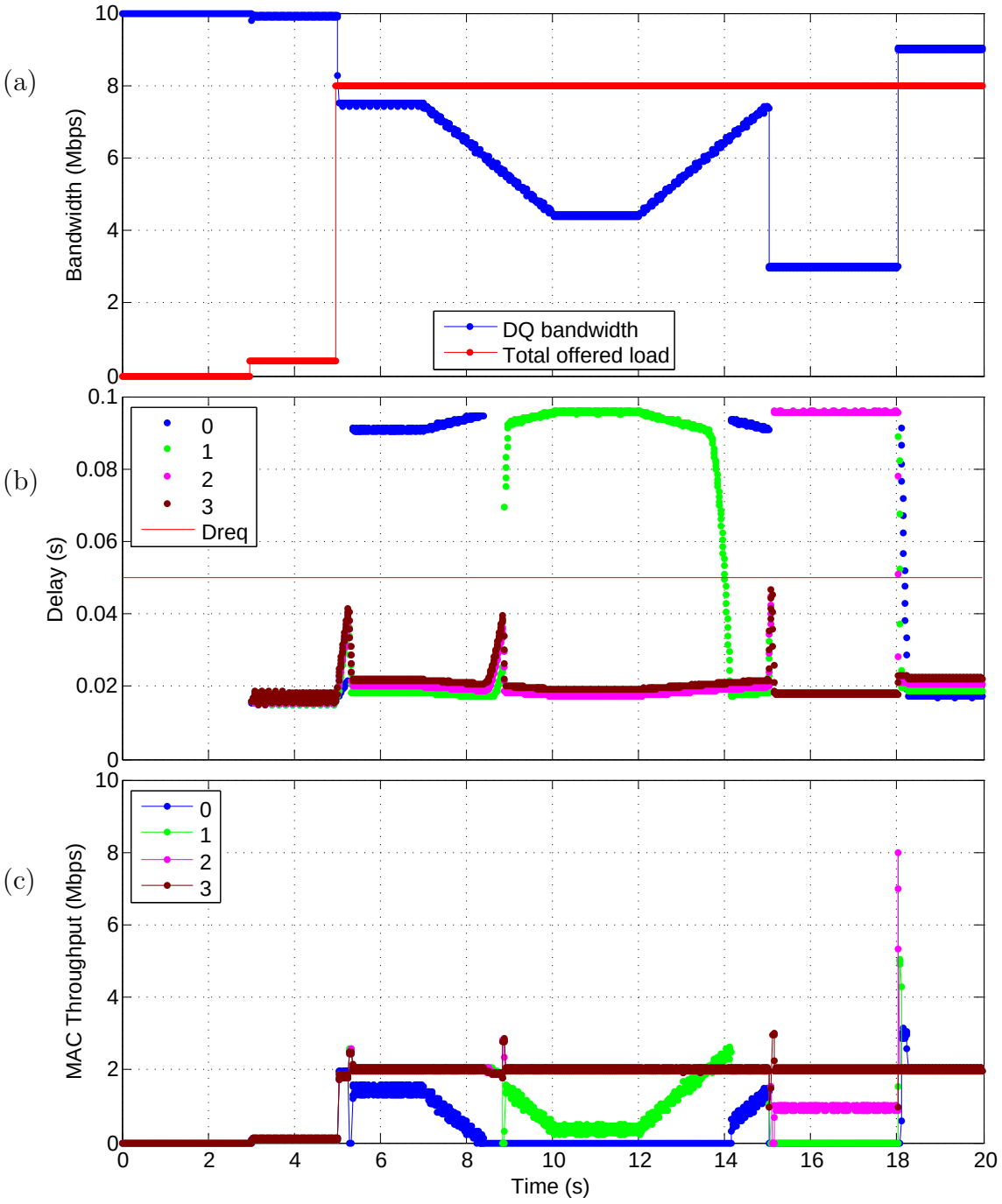

Figure 5.7.1: Experiment 1 (ULDQ): (a) DQ bandwidth, (b) packet delay, and (c) MAC throughput.

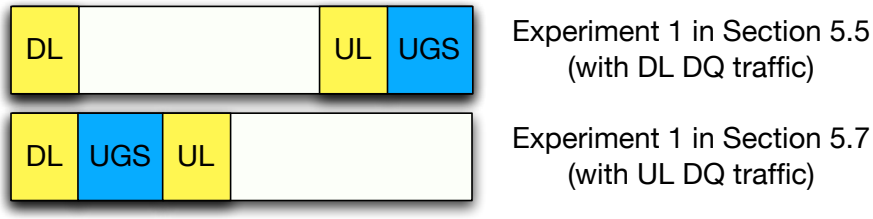

Figure 5.7.2: The MAC frame arrangement for different experiments.

MAC frame is higher than the average delay of the connections being consistently transmitted at the start of every MAC frame. Therefore, it is expected that the average delay experienced by the UL connections at the β-queue in this experiment be slightly greater than the average delay experienced by the DL connections at the β-queue described in experiment 1 of Section 5.5. For the UL connections at the α-queue, this impact is not obvious because their delay has been increased by D_{mac} due to the BR mechanism.

Next, we consider times when a connection is being served at the α-queue or the β-queue. From the actual data, the first QoS Violation event in experiment 1 described in Section 5.5 is detected at $t = 5.28$; for this experiment, the first QoS Violation event is detected at $t = 5.26$. There are two mainfactors that contribute to early detection on the UL. Firstly, the average delay of the UL connections at the α-queue is D_{mac} higher than the average delay of the DL connections at the α-queue. Secondly, the QoS Violation Detection algorithm used on the UL is more conservative than the one used on the DL. Therefore, it takes less time for the calculated delay of the UL connections at the α-queue to exceed D_{req}.

Note that the time difference between detections of QoS Violation event on the DL and the UL is dependent on the variability of the offered load and the DQ bandwidth of the network. In fact, it is hard to justify the claim that a QoS Violation event can be detected earlier on the UL based on two separate experiments. However, this claim is affirmed in Chapter 7.

Figure 5.7.1 (c) looks very similar to Figure 5.5.8 corresponding to experiment

1 of Section 5.5, except the actual times for connections being moved between the α-queue and the β-queue are not identical. This is expected as different transmission directions and QoS Violation Detection algorithms should not affect the MAC throughput of the network. In addition, we include Figure 5.7.3 (c) to show the QoS of each connection in the system, and this figure also looks very similar to Figure 5.5.3 (c) corresponding to experiment 1 of Section 5.5.

5.7.2 Experiment 2: Changing PHY mode

By comparing Figure 5.5.9 of Section 5.5 and Figure 5.7.5, we observe that there is not much difference except during uncongested periods, the average delay for the UL connections at the α-queue in this experiment is higher by D_{mac} than the average delay for the DL connections at the α-queue in experiment 2 described in Section 5.5.

Further, the average delay for the UL connections being served at the β-queue in this experiment is marginally higher than the average delay for the DL connections being served at the β-queue in experiment 2 of Section 5.5. This is due to the difference in MAC frame arrangement between the two experiments, as shown in Figure 5.7.4. We observe less impact on the average delay of the DL connections being served at the β-queue as those described in experiment 1 because there is no UGS traffic in experiment 2. Therefore, the impact only comes from the DL and the UL overheads.

We also observe that the average delay of the UL connections with the lowest index being served at the α-queue is always lower than those connections with a higher index due to the transmission ordering chosen for this experiment. That is, within a MAC frame, connections with the lowest index are transmitted ahead of connections with a higher index. For the DL connections in experiment 2 of Section 5.5, the transmission ordering is based on the connections' PHY mode, that is, connections with the lowest PHY mode are transmitted ahead of connections with

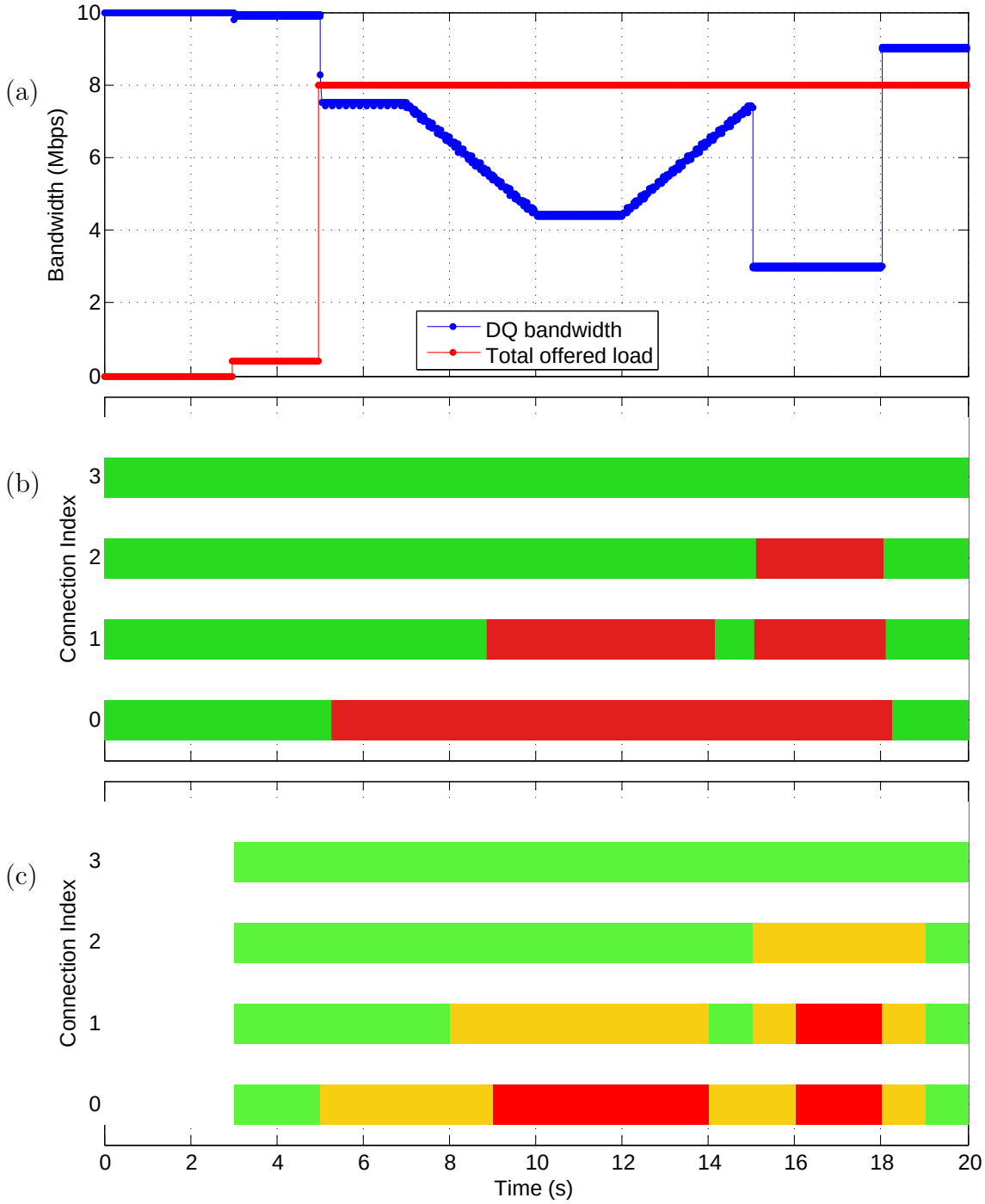

Figure 5.7.3: Experiment 1 (ULDQ): (a) DQ bandwidth, (b) connections at each queue; α-queue (green), β-queue (red), and (c) QoS received; good service (green), degraded service (amber) and no service (red).

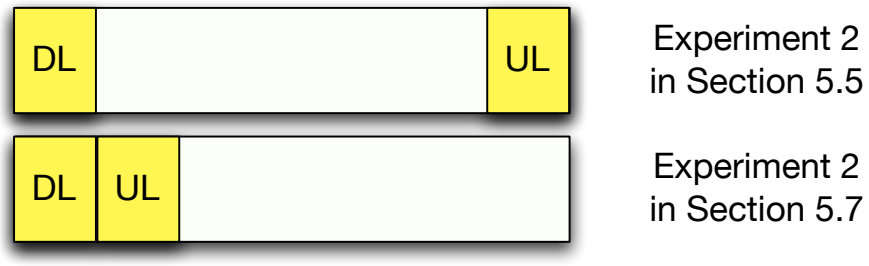

Figure 5.7.4: The MAC frame arrangement for experiment 2.

a higher PHY mode. From Figures 5.7.6 (c) and 5.5.10 (c), we observe similar QoS performance between this experience and experiment 2 of Section 5.5.

5.7.3 Summary

From the results discussed in experiments 1 and 2, we have demonstrated that our proposed ULDQ scheduler effectively manages congestion using the QoS Violation Detection and QoS Recovery Detection mechanisms. These were triggered by the changes in both the number of slots to serve the DQ traffic and the PHY mode of the SSs. Similar to the DL, the QoS Violation Detection and QoS Recovery Detection mechanisms are able to achieve their objective while maximising the number of connections at the α-queue, or equivalently the number of connections that receive good service at all times. Further, the secondary objective to maximise throughput is achieved.

The average delay of the UL connections at the α-queue has been shown to be higher than the average delay of the DL connections at the α-queue at all times by D_{mac}, which equals the Bandwidth Request polling interval. Due to this, and the conservative design of the QoS Violation Detection algorithm for the UL, the ULDQ scheduler is claimed to detect a QoS Violation event earlier, given the same network conditions.

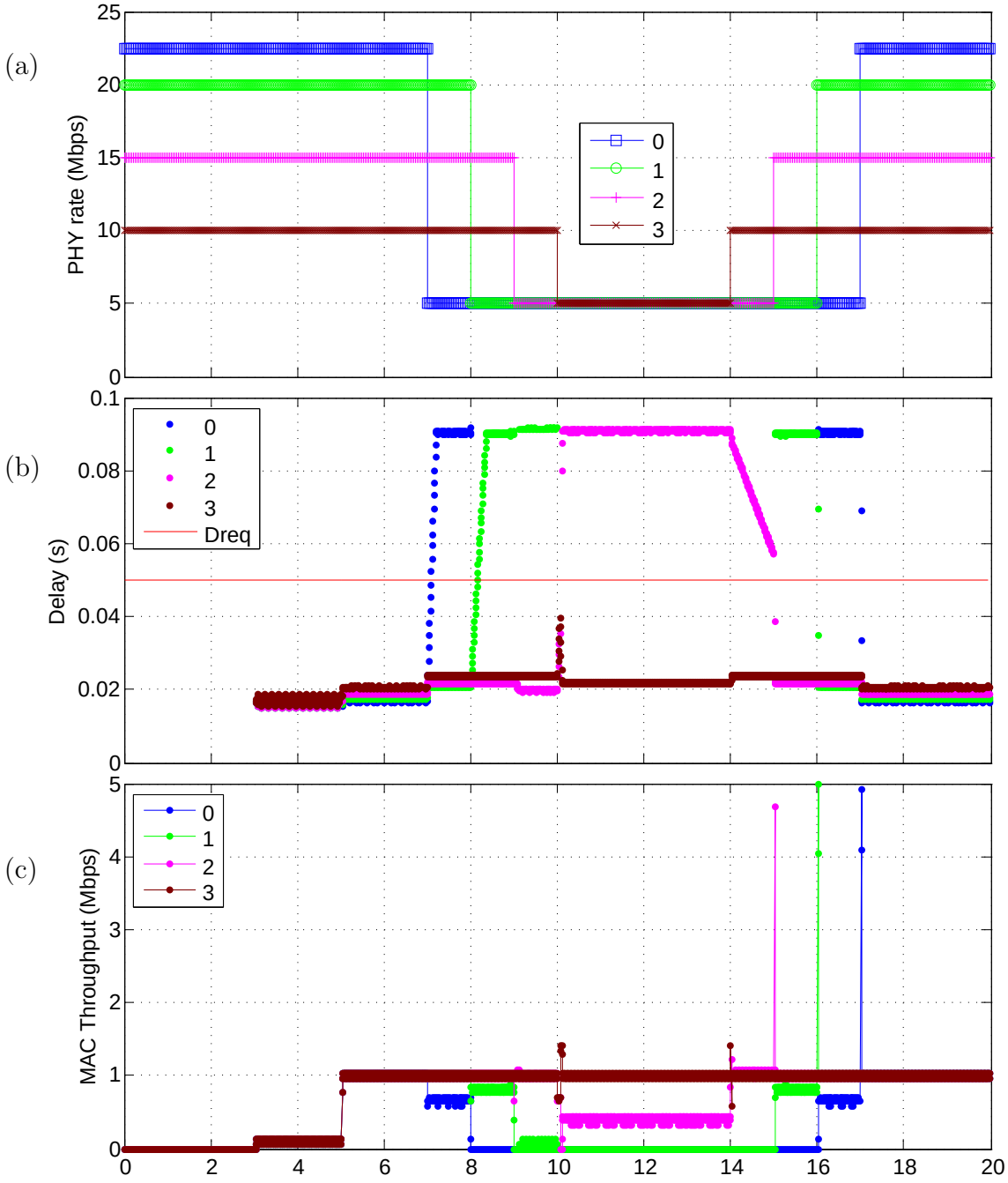

Figure 5.7.5: Experiment 2 (ULDQ): (a) PHY mode, (b) packet delay, and (c) MAC throughput.

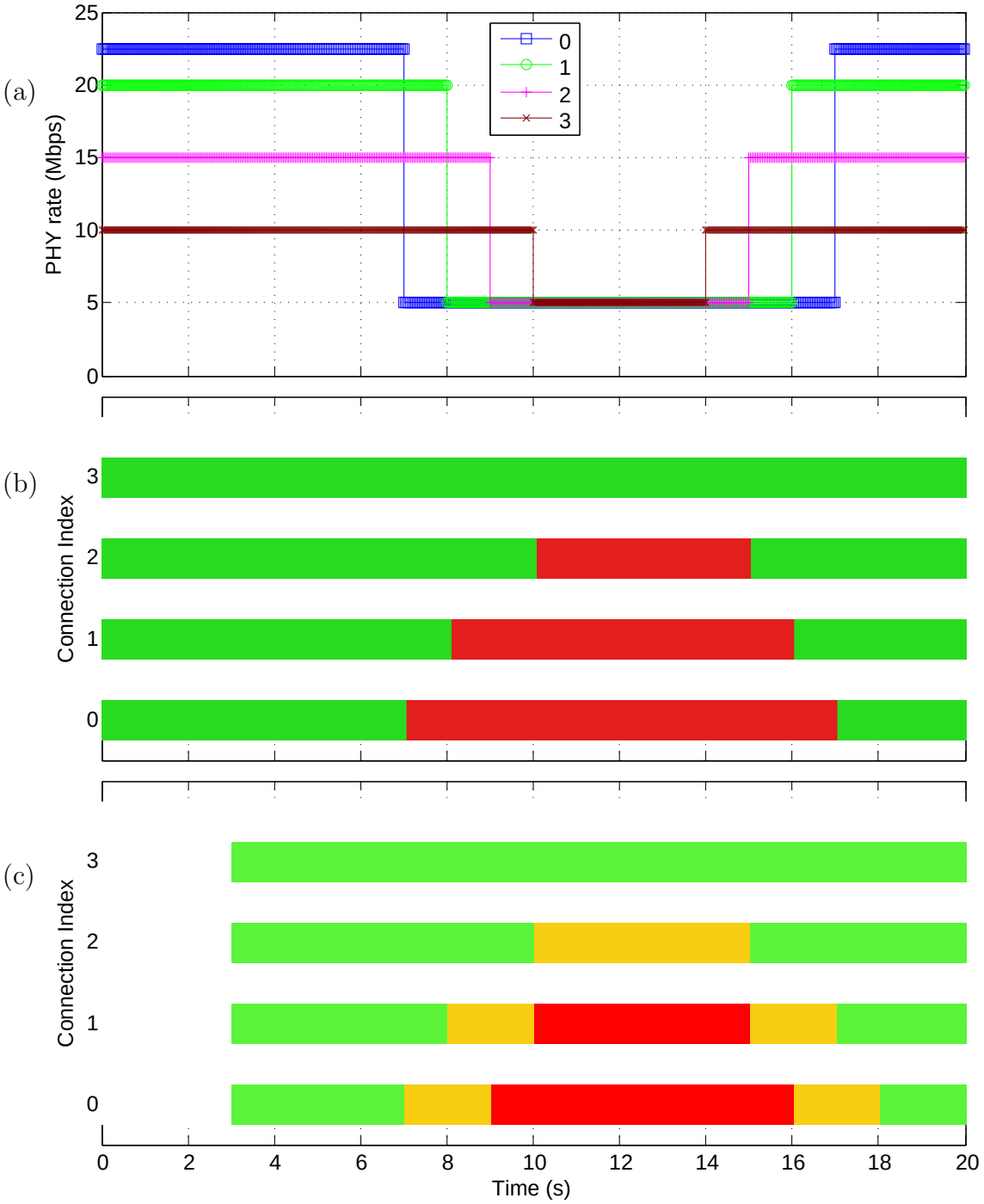

Figure 5.7.6: Experiment 2 (ULDQ): (a) PHY rate, (b) connections at each queue; α-queue (green), β-queue (red), and (c) QoS received; good service (green), degraded service (amber) and no service (red).

5.8 The Dual-Queue

Having investigated the DLDQ and the ULDQ schedulers separately in two experiments as described in Sections 5.5 and 5.7, we now investigate the performance of the entire DQ scheduler. In addition, the experiments conducted in this section include the impact of bit errors and the use of ARQ.

We compare the performance of the DQ scheduler to a WFQ scheduler. The reason for choosing the WFQ algorithm is because it is one of the common scheduling algorithms investigated in the 802.16 literature (see Chapter 2). At the same time, we investigate the impact of enhancing the WFQ scheduler with the EPD mechanism since the DQ scheduler explicitly incorporates the EPD mechanism.

We evaluate the performance of these schedulers in terms of the proportion of good service experienced by each connection. We also investigate the total good service MAC throughput achieved by these schedulers.

5.8.1 Experiment Description

We examine 3 schemes in this experiment, which aim to achieve the objectives as detailed in Table 5.5.2. That is, the primary objective of the system is to maximise the number of connections that experience good service and the secondary objective is to maximise throughput.

- Scheme 1 (Dual-Queue): We employ the DQ scheduler, which uses the DQ parameters shown in Table 5.5.2. The connection prioritisation mechanism assigns the highest priority to the connection with the highest PHY mode and the highest index, followed by the connection with the highest PHY mode and the next highest index and so on.

- Scheme 2 (WFQ): We employ a WFQ scheduler to distribute slots to all rtPS connections in a system. There is no connection prioritisation considered in this scheme, and so the weight assigned to each connection in the system is

equal.

- Scheme 3 (EPD enhanced WFQ): We employ a WFQ scheduler enhanced by the Explicit Packet Dropping (EPD) mechanism to distribute slots to all rtPS connections in a system. Similar to scheme 2, the weight assigned to each connection is equal. Further, the EPD mechanism is carried out to remove packets that have been in the system for more than D_{drop}.

We consider a network with four SSs connected to a BS. This experiment is carried out for 100 seconds. Similar to the experiments described in Sections 5.5 and 5.7, the first 5 seconds of this experiment is reserved for the initialisation process.

Having investigated CBR traffic source in Sections 5.5 and 5.7, we now investigate the behaviour of variable-bit-rate (VBR) traffic in our DQ system in this section. We introduce a simple VBR traffic source in this experiment by using an exponential traffic generator. Note that this experiment is designed to investigate an arbitrary VBR model. In Section 5.9, we investigate the behaviour of some real-time traffic models, such as video and VoIP, in our DQ system.

Each of these SSs carries one rtPS connection. Of the 4 connections, two of these connections are DL connections (connections DL1 and DL2) and the other two are UL connections (connections UL1 and UL2). The DL connections are handled by the DLDQ scheduler, while the UL connections are handled by the ULDQ scheduler. The packet size of these connections is 100 B (including IP header). Packets are generated in exponentially distributed bursts, with exponential interburst times. During "on" periods, packets are generated at constant bit rate. The average "on" time and "off" time of the exponential generator are arbitrarily set to 650 and 350 ms respectively. The burst rate of the exponential traffic generator is set to 2 $Mbps$.

To create a changing number of slots available to serve the rtPS traffic, we introduce a UGS connection, which is generated using a CBR traffic source with a packet size of 100 B (including IP header). The offered load of the UGS traffic is changed to a new rate every 10 seconds.

We consider a physical environment where the DL connections are operating at a PHY rate of 22.5 $Mbps$, while the UL connections are operating at a lower PHY rate of 15 $Mbps$. The experiment parameters are predominantly the same as those in Table 5.5.1, except that the BER is chosen to be 10^{-5}. In fact, the BER value does not directly affect our DQ scheduling decision because it is well taken care by the physical layer. For example, if the BER value is high, the physical layer may switch to a lower PHY mode to reduce retransmissions. Hence, our DQ system reacts according to a change in PHY mode as a result of a change in BER, rather than directly to a change in BER. However, we intend to show that our DQ system works well under conditions where the BER is not zero. More experiments under various non-zero BER values are carried out in Section 5.9.

We define the following notation. The total number of slots allocated for both DL and UL traffic is S, and the number of requested slots, including ARQ traffic, for the DL and the UL is X_{dl} and X_{ul} respectively. The number of slots allocated to the DL (S_{dl}) and the UL (S_{ul}) are given by

$$S_{dl} = \begin{cases} r_{dl} \times S, & \text{if } X_{dl} \geq r_{dl} \times S \text{ and } X_{ul} \geq r_{ul} \times S, \\ X_{dl}, & \text{if } X_{dl} \leq r_{dl} \times S, \\ S - X_{ul}, & \text{if } X_{ul} \leq r_{ul} \times S, \end{cases}$$

and

$$S_{ul} = S - S_{dl},$$

where $r_{dl} = r_{ul} = 0.5$.

We refer to the above allocation rule for the DL and the UL schedulers as the "DL and UL allocation rule". Other allocation rules can be implemented as required. Note that any ARQ traffic will be transmitted ahead of the un-transmitted traffic in the α-queue and the β-queue.

5.8.2 Experimental Results

In Figure 5.8.1 (a), we present the offered load of the UGS traffic and the PHY mode of each connection throughout the experiment. We observe a noisy offered load of the UGS traffic, resulting from retransmissions, as the BER is set to 10^{-5}. On Figures 5.8.1 (b) to 5.8.1 (d), we show the QoS of each connection in every one-second interval for each scheme.

With the Dual-Queue scheme, more connections experience good service when the offered load of the UGS traffic is low. When the offered load of the UGS traffic is high, which results in a fewer number of slots to serve the DQ traffic, more connections are sent to the β-queue, and hence they either experience degraded service or no service.

Given the independence of the DLDQ and the ULDQ schedulers, both the DL and the UL are allocated the same number of slots when both the DL and the UL have enough data to transmit. In the DLDQ scheduler, connection DL2 is given higher priority than connection DL1 because the tie breaker rule is based on connection indices, where a higher index corresponds to a higher priority. Similarly in the ULDQ scheduler, connection UL2 is given higher priority than connection UL1 due to the same tie breaker rule.

Therefore, in general, connections DL2 and UL2 have the same priority because of the even split of slots between the DL and UL. However, connection DL2 has a higher chance to be fully served due to its higher PHY mode and hence connection DL2 is expected to experience a better QoS than connection UL2. Similar arguments apply to connections DL1 and UL1. Based on the discussion above, in effect connection DL2 has the highest priority, followed by connection UL2, then connection DL1 and finally connection UL1.

With the EPD enhanced WFQ scheme, we observe that both DL connections experience a better QoS performance than the UL connections. Given the same offered load, connections UL1 and UL2 require more transmission slots because

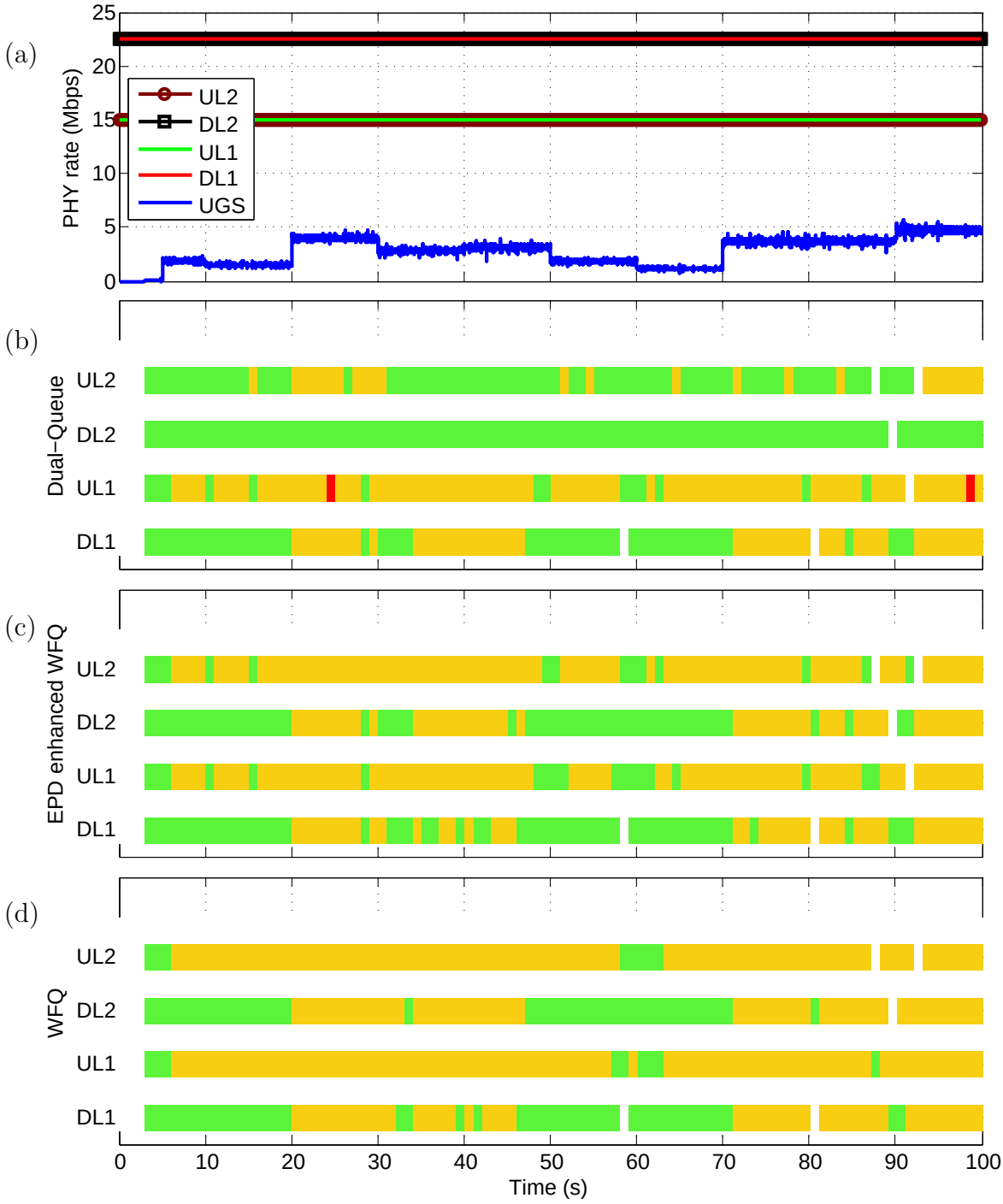

Figure 5.8.1: (a) UGS rate and PHY rate, (b) QoS received under DQ, (c) QoS received under EPD enhanced WFQ, and (d) QoS received under WFQ.

they have a lower PHY mode. Hence during congested periods, the DL connections are expected to experience a better QoS performance than the UL connections. The EPD mechanism drops packets that experience a delay of more than D_{drop}, giving a higher chance for other packets to meet the delay requirement of the system.

With the WFQ scheme, we observe one major difference compared with the other schemes, that is, longer blocks of degraded service being experienced by all connections, especially the UL connections. This is because the white area in Figures 5.8.1 (b) to 5.8.1 (d) is defined as times when no packets are waiting in the queue for transmission. Without the EPD mechanism, all packets can only leave the system either by transmission or by being dropped when the queues in the system overflow. The size of the buffers is set to a sufficiently large value in order to differentiate the WFQ scheme from the EPD enhanced WFQ scheme. With a small buffer size, we would obtain a similar effect to deploying the EPD mechanism but at the expense of greater packet loss and variability in delay when the PHY mode of a connection changes. Further, the UL connections have a lower PHY mode and hence they suffer even longer blocks of degraded service.

Notice that connections in the WFQ and the EPD enhanced WFQ schemes never experience no service. This is because the WFQ scheduling algorithm makes sure all connections are allocated at least some slots for transmission at all times, unless the number of slots available to serve the rtPS connections is less than the number of rtPS connections.

In order to quantify the improvement in the connections' QoS by using the DQ scheme, we calculate the overall proportion of good service of each connection under the different schemes. This is equal to the number of seconds experiencing good service divided by the total number of seconds experiencing either good service, degraded service or no service. In other words, 100 seconds minus the number of seconds that the connection has no packets in its queue. This is shown in Figure 5.8.2.

We observe that connections DL2 and UL2 experience an overall proportion of

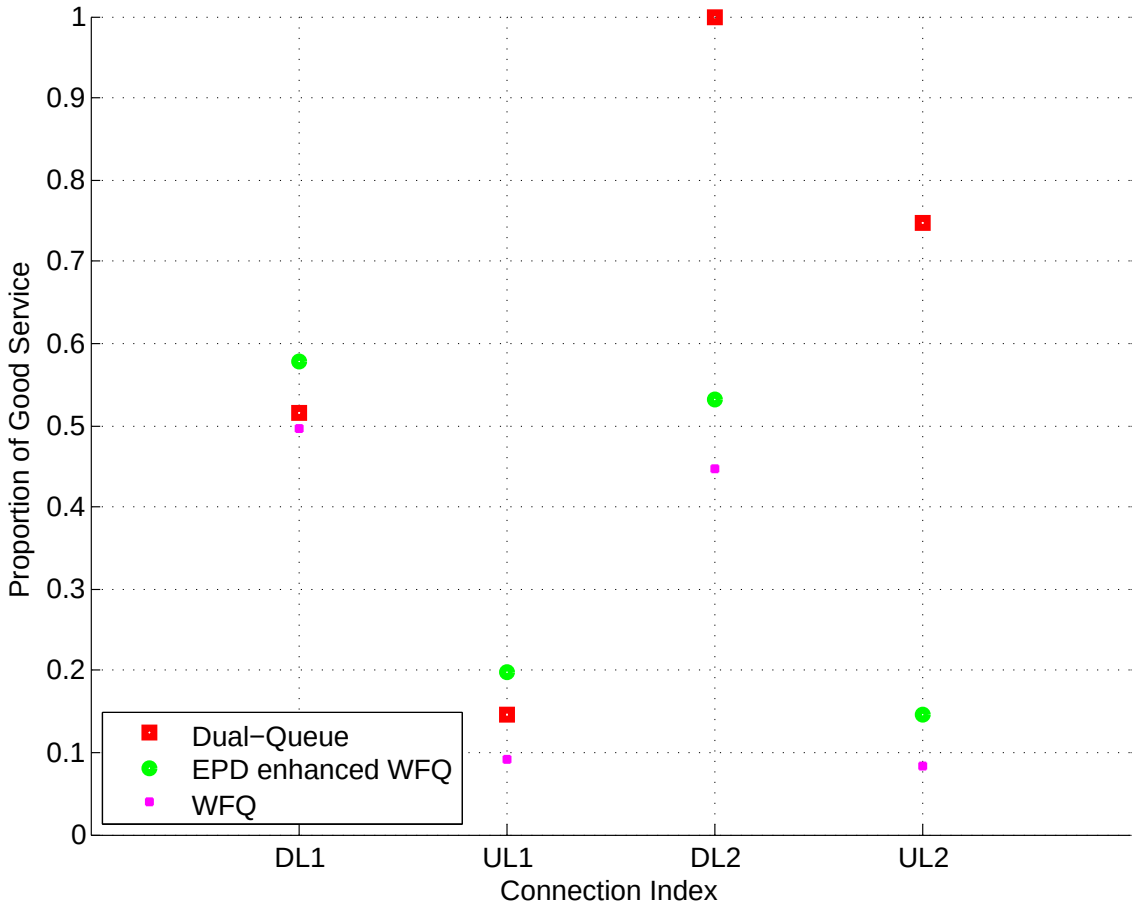

Figure 5.8.2: Overall proportion of good service of all connections under the different schemes.

good service of 1 and 0.75 respectively with the DQ scheduler. The trade off in maximising the number of connections that experience good service with the DQ scheduler is shown by a slightly worse QoS achieved by connections DL1 and UL1 compared to the EPD enhanced WFQ scheme. However, these connections still achieve a better QoS than the WFQ scheme. Furthermore, the improvement in proportion of good service experienced by connections DL2 and UL2 with the DQ scheduler is much greater than the QoS lost by connections DL1 and UL1.

We also observe that the overall proportion of good service of all connections is higher in the EPD enhanced WFQ scheme than in the WFQ scheme. This shows

the importance of deploying the EPD mechanism to remove packets that have been waiting for too long in the queues. In both the EPD enhanced WFQ and the WFQ schemes, we observe that the DL connections achieve a higher overall proportion of good service than the UL connections, due to the higher PHY mode of the DL connections.

In Figure 5.8.3 (a), we plot the cumulative number of bits transferred by all connections under the different schemes. The highest number of bits transferred is achieved by the WFQ scheme because the system always has packets to send due to the backlog of packets in the queues since this scheme does not employ the EPD mechanism. Hence, the system is more utilised under the WFQ scheme.

In order to show the system being more utilised under the WFQ scheme, we plot the cumulative number of unused slots throughout the experiment under the different scheduling schemes in Figure 5.8.4. These cumulative unused slots represent the cumulative free bandwidth after serving the DQ traffic throughout the experiment. Under the WFQ scheme, the least number of unused slots is accumulated throughout the experiment. Notice that between time 20 to 50 seconds, the system is fully utilised at all times, as there are no unused slots in the system during this time interval in the WFQ scheme. This time period also corresponds to the long time blocks of degraded service being experienced by the UL connections in the WFQ scheme, as shown in Figure 5.8.1 (d).

In Figure 5.8.5, we plot the cumulative number of packets dropped in the system under the different schemes. The packets are dropped either by the EPD mechanism or by the system itself when the queue overflows. Since there is no EPD mechanism employed in the WFQ scheme, the only factor that contributes to packet dropping is queues overflowing. Hence, there is a relatively smaller number of packets being dropped under the WFQ scheme. Until the queue in the system is full at $t = 24$, there are no packets dropped under the WFQ scheme.

In contrast, the system is less utilised under the DQ and the EPD enhanced WFQ schemes due to the EPD mechanism, which drops packets that experience a

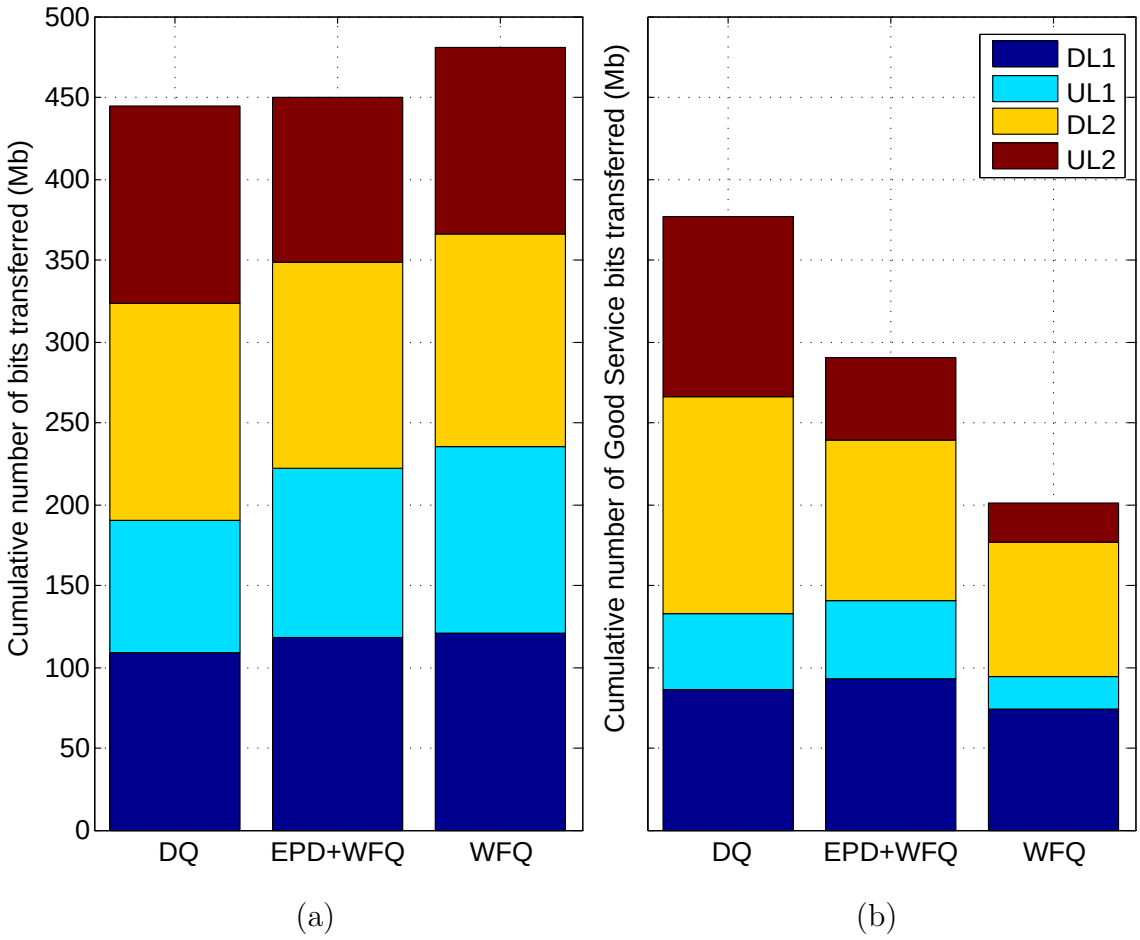

Figure 5.8.3: (a) Cumulative number of bits transferred, and (b) cumulative number of good service bits transferred under the different scheduling schemes.

delay of more than the specified delay requirement. Therefore, we would expect a lower cumulative number of bits transferred to be achieved by the DQ and the EPD enhanced WFQ schemes. Furthermore, we observe that between the DQ and the EPD enhanced WFQ schemes, there are more unused slots in the system under the DQ scheme, due to more packets being dropped by the EPD mechanism in the DQ scheme.

We can explain why more packets are being dropped by the EPD mechanism in the DQ scheme than in the EPD enhanced WFQ scheme. If the system is congested, the number of slots allocated to each connection by the DQ scheduler is dynamic

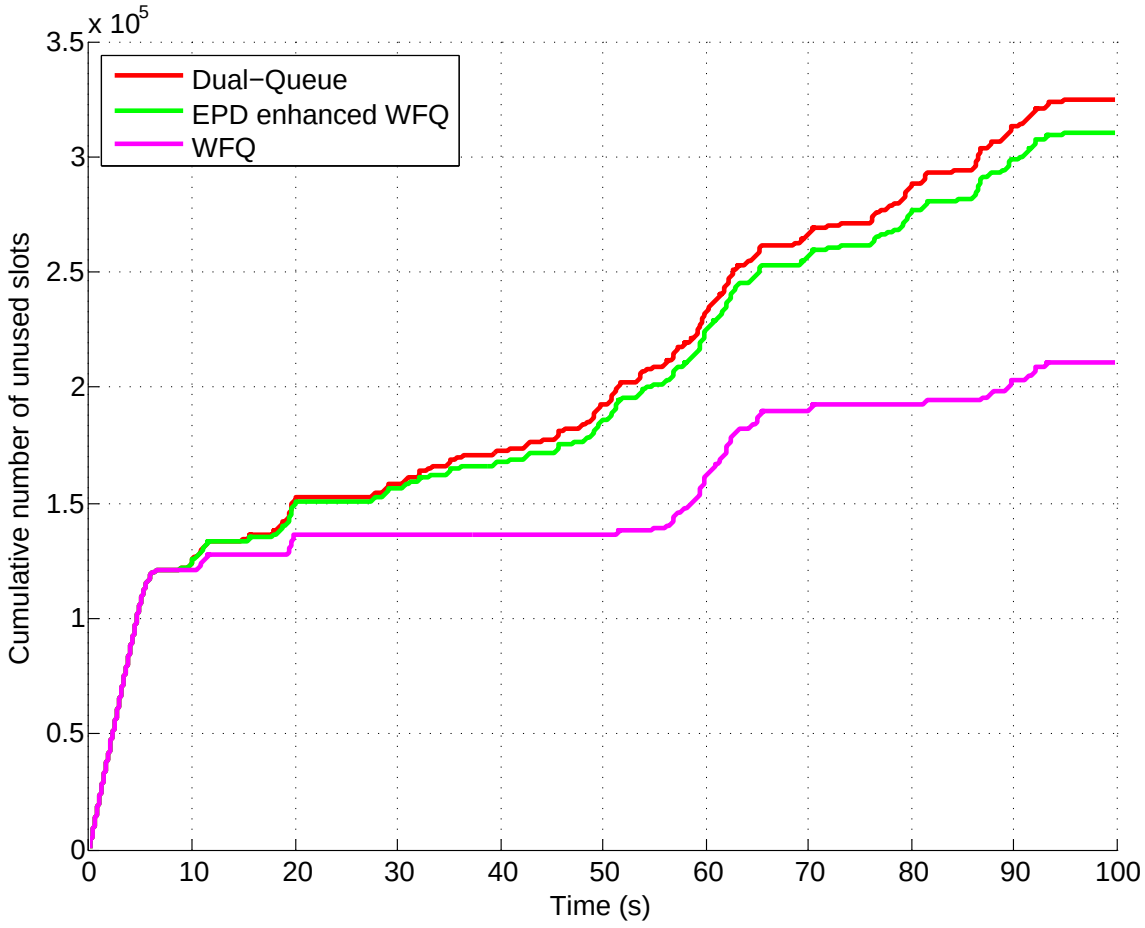

Figure 5.8.4: Cumulative number of unused slots under the different scheduling schemes.

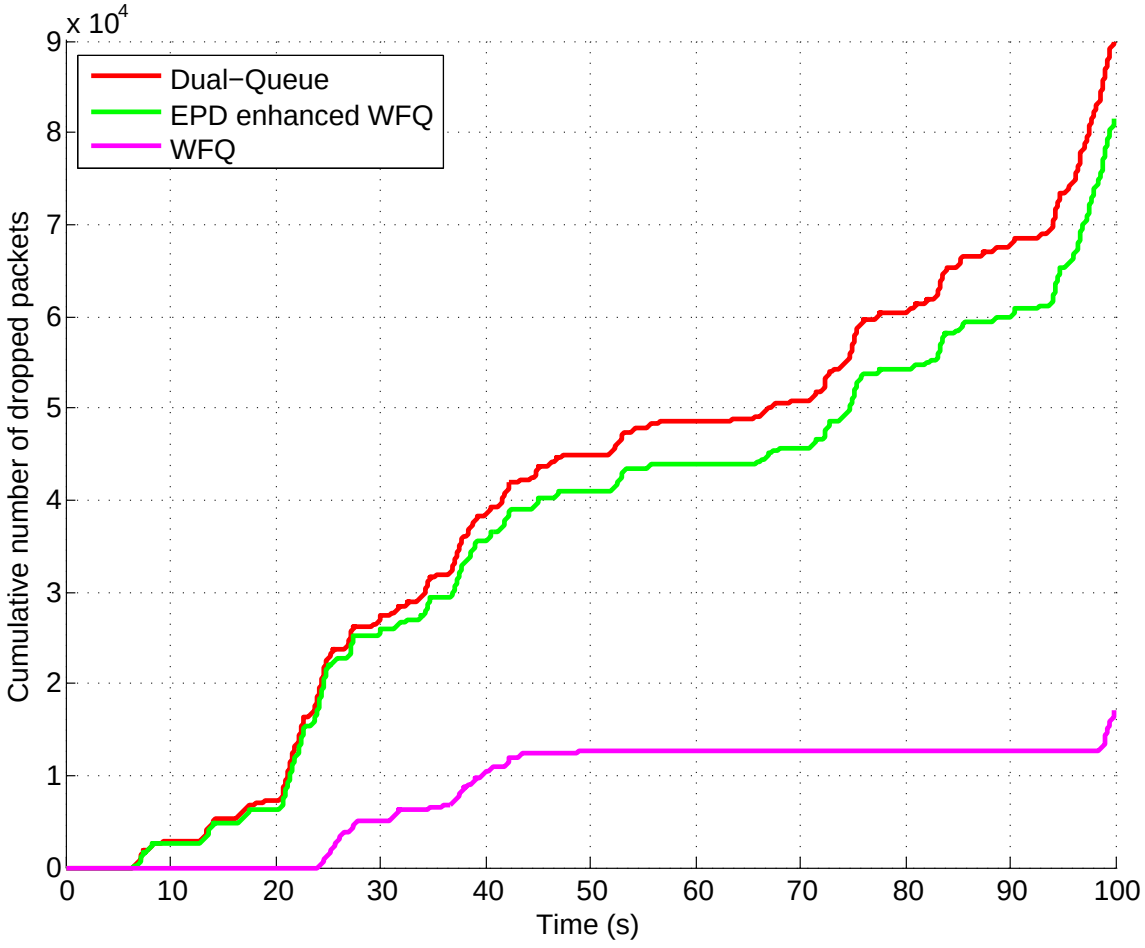

Figure 5.8.5: Cumulative number of packets dropped under the different scheduling schemes.

and it depends on the priority of each connection. In contrast, the number of slots allocated to each connection under the EPD enhanced WFQ scheme is equal under congested periods.

Let us consider a scenario where there is one connection in the α-queue and the β-queue respectively, and there is no α or β differentiation for these two connections in the EPD enhanced WFQ scheme. If the number of slots required by each of these connections is P and the total number of slots available to serve these connections is also P, the number of slots allocated to the connection in the α-queue and β-queue is P and 0 respectively in the DQ scheme; while in the EPD enhanced WFQ scheme the number of slots allocated to each connection is $P/2$. Therefore, the number of packets of the connection at the β-queue in the DQ scheme is definitely more than the number of packets at the queue for any connections in the EPD enhanced WFQ scheme. In contrast, the number of packets of the connection at the α-queue in the DQ scheme is definitely less than the number of packets at the queue for any connections in the EPD enhanced WFQ scheme. If this situation remains for sufficiently long, all these packets already in the β-queue would eventually be dropped. Due to the higher backlog of packets of the connection at the β-queue in the DQ scheme, more packets are dropped by the EPD mechanism in the DQ scheme than in the EPD enhanced WFQ scheme. This argument is evident in Figure 5.8.5, where the cumulative number of packets dropped in the DQ scheme is the highest at all times.

As there are more packets being dropped by the EPD scheme under the DQ scheme, it is no surprise that there are more unused slots in the system under the DQ scheme than under the EPD enhanced WFQ scheme. Due to these reasons, the cumulative number of bits transferred by the DQ scheme is lower than that achieved by the EPD enhanced WFQ scheme.

Having shown that the highest cumulative number of bits transferred is achieved by the WFQ scheme, we need to consider the usefulness of these bits transferred. If some of these packets fail to meet the QoS requirements, then they would be

considered as having less value to the system and their bandwidth may have been better allocated to other services, such as the best effort traffic. Due to this, we are only interested in the bits transferred from packets that experience good service, that is, those packets that experience a delay of less than D_{req}, which we call the "good service" bits transferred.

We show the cumulative number of "good service" bits transferred of each scheme in Figure 5.8.3 (b). In this graph, we observe that the highest cumulative number of "good service" bit transferred is achieved by the DQ scheme, followed by the EPD enhanced WFQ scheme, and finally the WFQ scheme. In the DQ scheme, connection DL2 achieves the highest cumulative "good service" bits transferred as it has the highest priority given by the DQ scheduler.

5.8.3 Summary

In this section, we showed that the DLDQ and the ULDQ schedulers are able to operate independently in the system. We also showed that our proposed DQ scheduler is able to operate properly under the existence of ARQ traffic. Further, we showed that our proposed DQ scheme performs better than the EPD enhanced WFQ and the WFQ scheduling schemes in an 802.16 system, in terms of maximising the number of connections that experience good service.

In order to quantify the improvement of our DQ scheduler, we compared the overall proportion of good service experienced by each connection in the system. With the DQ scheduler, the highest overall proportion of "good service" is achieved by the connections assigned the highest priority by the DQ scheduler. We also discussed the trade off in deploying the DQ scheduler, which is sacrificing the QoS of lower priority connections. However, we showed that the degraded QoS experienced by the lower priority connections is small, compared to the improved QoS experienced by the higher priority connections. Therefore, the DQ scheduler is able to provide good service to more connections simultaneously than the WFQ and the

EPD enhanced WFQ scheduling disciplines.

Furthermore, we showed the important role of the EPD mechanism in the DQ and the EPD enhanced WFQ schedulers. By employing the EPD mechanism, transmission slots are not wasted on packets that have exceeded their delay requirement. Thus, we may give more transmission opportunities to packets that have the potential to meet their delay requirement. This results in a lower use of network resources in the DQ and the EPD enhanced schemes, which is a desired outcome as unused bandwidth could be allocated to other services.

Lastly, we investigated the impact of employing the DQ scheduler in terms of cumulative number of bits transferred. We first showed that the least total cumulative number of bits transferred is achieved by the DQ scheduler, which is one of the consequences of more packets being dropped by the EPD mechanism. We contended that a better measure is bits transferred that meet the QoS requirements. Hence, we compared the cumulative number of bits transferred that represents packets experiencing good service and showed that the highest cumulative "good service" bits transferred is achieved by the DQ scheme.

5.9 Mixed Traffic Profiles Experiments

5.9.1 Experiment Description

In this experiment, we investigate the robustness of our DQ system at handling real-time traffic. We consider two kinds of traffic: VoIP and MPEG (Moving Picture Expert Group) video.

VoIP can either be like a CBR source or a VBR source, depending on the codec used. An example of CBR based VoIP is G.711, which generates 160 B of payload every 20 ms, giving a payload rate of 64 $Kbps$. However, human conversation consists of talk-spurts and silence periods, also known as on-off patterns. Therefore, during silence periods, a VoIP encoder may choose not to generate a payload in

order to save transmission resources. This is carried out by Voice Activity Detection (VAD). In general, VAD can be applied to any of the CBR-like VoIP codecs. Alternatively, there are codecs that include an integrated VAD function, such as G.729B and G.723.1A.

Given that VoIP with VAD has become common, we use a G.729B model in our experiment. According to [114], G.729B is identical to G.729 (CBR based) with VAD and Comfort Noise Generation (CNG). CNG is used to differentiate between a disconnected call and a silence period. During silence periods, CNG locally generates "white noise" so that a call appears connected. G.729 generates 20 B of payload every 20 ms, giving a payload rate of 8 $Kbps$.

There are different models used to describe the characteristics of the VAD process, such as exponential [115]–[117], Weibull [118]–[120] and Gamma [118], [121]. Most of the earlier work that investigates the on-off patterns of voice, uses exponential models. However, Jiang and Schulzrinne [122] claim that exponential models no longer fit well to modern voice codecs and silence detectors. They, however, note explicitly that exponential models are still useful for first-hand performance estimates. On the other hand, Bellalta *et al.* [118] claim that the Gamma and Weibull models should be used to describe the on and off periods respectively. Given that our research focus is not on VoIP codecs, but on performance of our QoS scheme, we use a simple exponential model to describe the on-off patterns of our G.729B implementation.

To complete our G.729B implementation, we need reasonable values for the mean on and mean off periods. Before we discuss the choices that we have, we introduce the term "hangover". Hangover is a technique used to avoid sudden end-clipping of speeches and to bridge short silence periods [122]. "Hangover time" is hence defined as the amount of time added to the end of an active speech interval that is transmitted. G.729B is one of the codecs that include hangover time. Now, we discuss the result from some work on the on-off pattern of human speech. Sriram and Whitt quote a mean talk-spurt of 352 ms and a mean silence period of 650 ms

with zero hangover time [123]. While International Telecommunication Union (ITU) P.59 recommendation specifies an artificial on-off model for human speech with a mean on period of 1.004 s and a mean off period of 1.587 s with hangover [124]. The ITU P.59 recommendation also has a model without hangover, which gives a mean on time of 227 ms and a mean off time of 596 ms. On the other hand, Brady's result [117] gives 1.2 s and 1.8 s for the mean on and mean off periods respectively after applying hangover. Since G.729B uses hangover, we choose the ITU P.59 recommendation because its result is derived from multiple sources, including some of Brady's work. Therefore, we choose a mean on period of 1.004 s and a mean off period of 1.587 s for our G.729B implementation.

For the video traffic, we obtained a MPEG-4 traffic model from [125] that uses the Transform Expand Sample (TES) methodology. TES is a methodology to generate data that closely matches the marginal distribution and auto-correlation function [125]. There are two input values for this model: $rateFactor$ and $initialSeed$. $rateFactor$ is a scaling factor for changing the size of each frame while preserving their marginal distribution and auto-correlation function. For example, the average data rate of the TES traffic model that corresponds to a $rateFactor$ of 1 is 0.25 $Mbps$ and a $rateFactor$ of 10 is 2.5 $Mbps$. The $initialSeed$ is used to generate a random first frame of a video connection.

MPEG-4 is designed to support low bit rate video connections, such as video streaming and video conferencing. However, according to [126], MPEG-4 supports a large number of data rates, ranging from 16 $Kbps$ to at least 20 $Mbps$, depending on audio and video encoding techniques, as well as quality requirements.

We obtained NS-2 source code from [127] that generates video traffic based on real video traces. We downloaded various sample MPEG-4 movie traces from [128], such as *Jurassic Park I, Silence of the Lambs and Mr. Bean.* The information contained in these movie traces includes frame type, frame number, frame size and sending time of frame. The downloaded traces from [128] have an average data rate range of 200 $Kbps$ to 700 $Kbps$.

Having discussed the VoIP and video models, we now describe our experiment. We apply SS differentiation, as described in Chapter 3, to this experiment to show the flexibility of our DQ system. We apply a gold/silver SS differentiation scheme, such that, the gold class SSs will always be served ahead of the silver class SSs. Further, we assign a higher priority to VoIP connections because they have a lower data rate and they are more delay sensitive. A summary of the traffic profiles used in this experiment is shown in Table 5.9.1.

Each of the video connections based on the TES model is generated by a random $initialSeed$ value and a random $rateFactor$, giving an average payload rate that ranges from 250 $Kbps$ up to 3 $Mbps$. For each of the video connections based on real movie traces, we randomly select 1 out of 10 movie traces available to be transmitted.

This experiment is carried out for 100 seconds and the first 5 seconds of this experiment is reserved for the initialisation process. All connections are started at random times after $t = 5$ and they last until the end of the experiment. Note that VoIP connections that belong to the same session are started at the same time. Furthermore, all connections operate at the same PHY rate of 22.5 $Mbps$. To create a high utilisation environment, we introduce a UGS connection, which has a constant offered load of 2.5 $Mbps$ from $t = 5$ until the end of the experiment.

To obtain confidence intervals for our results, we replicate this experiment 20 times with random starting times for all connections, random $initialSeed$ and $rateFactor$ values for video connections based on TES model and random selection of real movie traces to be started in the experiment. We also set BER to be zero.

Finally, we rerun the same set of experiments with different non-zero BER values. Given that the PHY rate is set to 22.5 $Mbps$ for all connections, the BER values must be small, ranging from 10^{-4} to 10^{-6}.

Table 5.9.1: A summary of mixed traffic profiles.

Index	Description
TG.D1	(Gold) First downlink MPEG-4 video based on TES model.
TG.D2	(Gold) Second downlink MPEG-4 video based on TES model.
TS.D1	(Silver) First downlink MPEG-4 video based on TES model.
TS.D2	(Silver) Second downlink MPEG-4 video based on TES model.
TS.D3	(Silver) Third downlink MPEG-4 video based on TES model.
MG.D1	(Gold) First downlink MPEG-4 video based on movie traces.
MG.D2	(Gold) Second downlink MPEG-4 video based on movie traces.
MS.D1	(Silver) First downlink MPEG-4 video based on movie traces.
MS.D2	(Silver) Second downlink MPEG-4 video based on movie traces.
MS.D3	(Silver) Third downlink MPEG-4 video based on movie traces.
VG.D1	(Gold) Downlink VoIP connection for user 1 (G.729B)
VG.D2	(Gold) Downlink VoIP connection for user 2 (G.729B)
VS.D3	(Silver) Downlink VoIP connection for user 3 (G.729B)
VS.D4	(Silver) Downlink VoIP connection for user 4 (G.729B)
VS.D5	(Silver) Downlink VoIP connection for user 5 (G.729B)
VS.U1	(Gold) Uplink VoIP connection for user 1 (G.729B)
VG.U2	(Gold) Uplink VoIP connection for user 2 (G.729B)
VS.U3	(Silver) Uplink VoIP connection for user 3 (G.729B)
VS.U4	(Silver) Uplink VoIP connection for user 4 (G.729B)
VS.U5	(Silver) Uplink VoIP connection for user 5 (G.729B)

5.9.2 Experimental Results

In Figure 5.9.1, we plot the MAC throughput of all connections for one of the 20 experiments carried out. From Figures 5.9.1 (a) and (b), we notice that the peak video MAC throughput can go up to 4.4 $Mbps$. In Figure 5.9.1 (c), the MAC throughput of G.729B during most of the on periods is 24 $Kbps$, due to the extra Real-time Transport Protocol (RTP), User Datagram Protocol (UDP) and IP overheads of 40 B for every 20 B of payload. In certain times, the MAC throughput of some G.729B connections can go up to 32 $Kbps$, due to queueing of packets in buffers, followed by transmission bursts.

In Figure 5.9.2, we plot the delay of the connections shown in Figure 5.9.1. We notice in Figures 5.9.2 (a) and (b) that the silver class video connections experience a higher average delay than the gold class video connections. Most of the video packets that exceed the delay requirement belong to the silver class connections. For the G.729B connections, most of them experience a delay between 5 ms to 25 ms, except for 3 silver class connections (VS.D3, VS.D4 and VS.D5), as shown in Figure 5.9.2 (c). Note that these downlink silver class G.729B connections experience a higher average delay than those silver class connections on the uplink because the downlink is more congested, due to the high-data-rate downlink video connections. Hence, QoS Violation events are detected at the downlink, which send downlink connections to the β-queue.

Next, we plot the QoS of each connection in Figure 5.9.3. From Figure 5.9.3, we notice that all the gold class video and VoIP connections experience good service, except for connection TG.D1, which suffers one second of degraded service. This is a transient effect due to a sudden increase in payload rate of the video connections, as explained in Sections 5.5 and 5.7.

In Figure 5.9.4, we plot a 95% confidence interval of the mean proportion of good service for each connection, based on the 20 experiments. From Figure 5.9.4, we note that all the gold class connections experience a proportion of good service

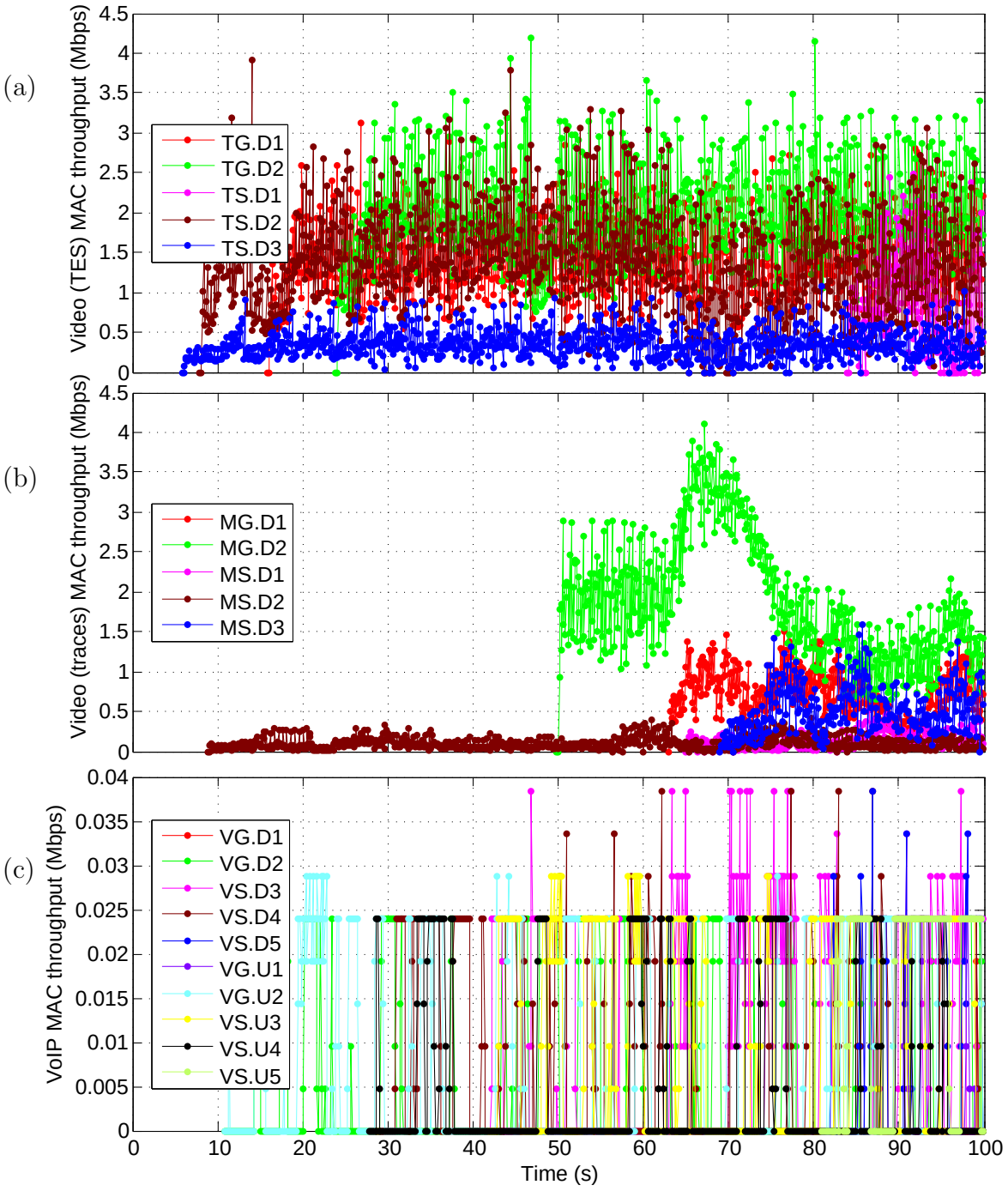

Figure 5.9.1: (a) Video (TES) MAC throughput, (b) Video (traces) MAC throughput, (c) VoIP MAC throughput.

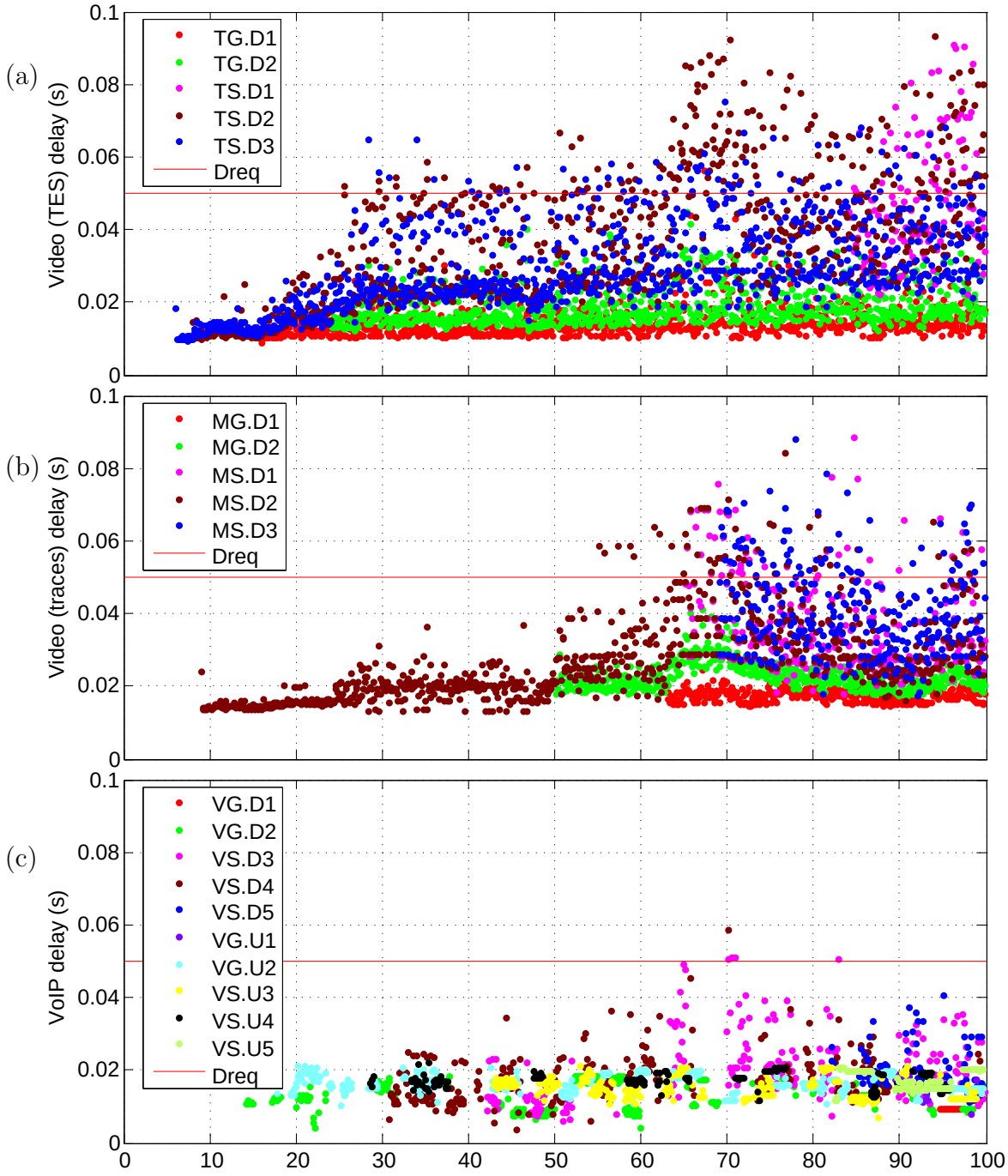

Figure 5.9.2: (a) Video (TES) delay, (b) Video (traces) delay, (c) VoIP delay.

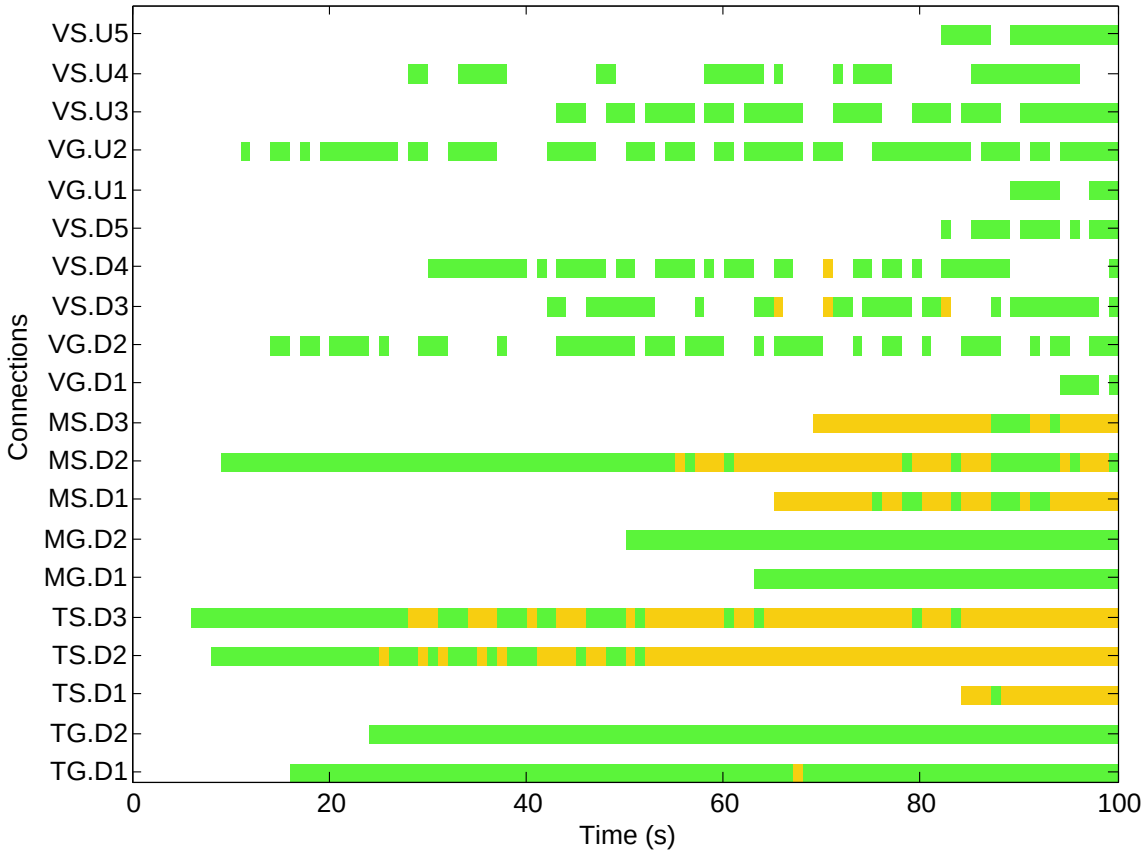

Figure 5.9.3: QoS experienced by each connection for one of the 20 experiments; good service (green), degraded service (amber) and no service (red).

very close to 1. For the silver class connections, the G.729B connections experience a higher proportion of good service than the video connections, due to the higher priority assigned to G.729B connections.

Finally in Figure 5.9.5, we plot a 95% confidence interval of the mean proportion of good service for each connection for our experiments under different BER values. It is noticed from Figure 5.9.5 that in general, a higher BER value results in a lower proportion of good service. This is due to the fact that a higher BER value causes a higher retransmission rate, which eventually results in less transmission opportunities for the DQ traffic. This effect is greater for the silver class connections because they are served based on the transmission opportunities left after serving the gold class connections. Further, we also notice that the uplink gold class G.729B connections suffer a lower proportion of good service than the downlink gold class G.729B connections. This is due to the fact that uplink connections experience a higher delay due to the extra waiting time for sending bandwidth requests to the BS. Hence, an increase in BER value has a larger impact on the QoS of the uplink connections.

5.10 Summary

In this chapter, we have proposed a DQ implementation for 802.16 systems for the purpose of handling real-time services in both the DL and the UL. We have developed the required framework for the implementation, which addressed issues that include the structure of the α-queue and the β-queue, handling ARQ traffic and the service scheme deployed by the DQ scheduler. We have also proposed the core DQ mechanisms for the DL and the UL separately. Next, we have conducted the same set of experiments for the DL and the UL and showed that our ULDQ proposal has similar performance to our DLDQ proposal.

We have compared the performance of our DQ scheduler to a WFQ scheduler under the existence of Automatic Repeat Request traffic. We have also conducted

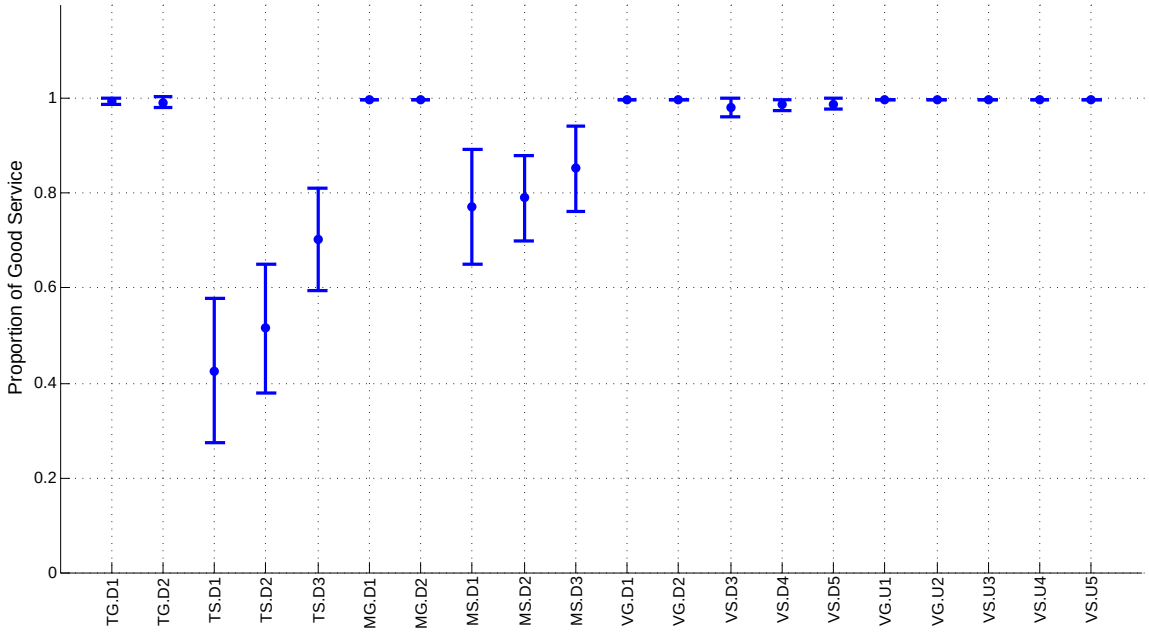

Figure 5.9.4: 95% confidence interval of the mean proportion of good service for each connection, with $BER = 0$.

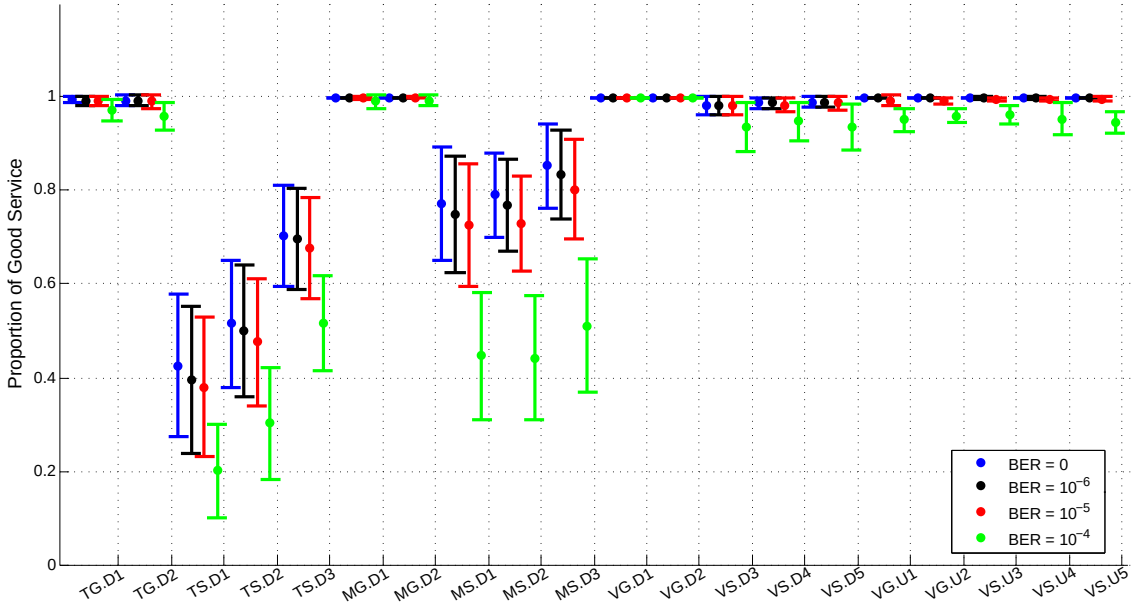

Figure 5.9.5: 95% confidence interval of the mean proportion of good service for each connection under different BER values.

a comparison between our DQ scheduler and an EPD enhanced WFQ scheduler. From the experimental results, we have shown that our proposed DQ has much better QoS performance not only in terms of maximising the number of connections that experience "good service", but also in terms of maximising "good service" throughput.

We have also discussed the trade off in employing the DQ scheduler, which sacrifices the QoS of lower priority connections. However, we have shown that the degradation in QoS experienced by the lower priority connections is small, compared to the improvement in QoS experienced by the higher priority connections. On the other hand, we have shown the importance of the EPD mechanism in the DQ and the EPD enhanced WFQ schedulers, which results in a lower use of network resources. This is a desired outcome as any unused bandwidth could be allocated to other services.

Furthermore, we have shown that our DQ system can handle mixed traffic profiles, such as MPEG-4 video and G.729B VoIP. We have also shown that our DQ system is able to work well under different non-zero BER conditions.

Note that the DQ scheduler can operate along with any existing scheduling schemes employed in an 802.16 network. The DQ scheduler could be turned off by using the α-queue only, that is, by choosing D_{req} so that no QoS Violation events can be detected.