CROSS-LAYER RACM DESIGN FOR VERTICALLY INTEGRATED WIRELESS NETWORKS

A dissertation submitted to the Department of Computer Science,
Faculty of Science at the University of Cape Town
in fulfilment of the requirements for the degree of

MASTERS

in

COMPUTER SCIENCE

— Paolo P Pileggi —

Supervisor

Professor Pieter S Kritzinger
University of Cape Town, South Africa

UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Co-supervisor

Professor Giuseppe Iazeolla
University of Rome 'Tor Vergata', Italy

ABSTRACT

IEEE 802.16 wireless metropolitan area network (WMAN) technology is an improvement on its wireless local area network (WLAN) counterpart, namely IEEE 802.11, in that it provides longer range and higher bandwidth capabilities. More importantly, it specifies a connection-oriented medium access control layer (MAC) and scheduling services to support quality of service (QoS) in IEEE 802.16 networks. However, in the standard, scheduling and connection admission control (CAC) mechanisms are left unspecified, leaving this for network operators to decide. This allows implementers to create market and performance advantages, making it a rich field of research and performance analysis.

Typically, researchers studying scheduling and admission control in such wireless networks consider these resource and connection management (RaCM) algorithms in isolation: They investigate (1) schedulers while fixing the admission control processes or using static connection scenarios and (2) admission controllers while fixing the scheduling processes.

We hypothesize that there exists an interdependent relationship between RaCM components which is an essential aspect to cross-layer inter-RaCM algorithm design.

In order to prove our hypothesis that you cannot consider the scheduler and the CAC in isolation, where it involves the performance of IEEE 802.16 networks, we require a performance model: Analytic modelling is an ideal solution but the system is far too complex. Experimental test beds are expensive, making hardware experimentation another impractical solution. The only other feasible solution is simulation.

General simulation environments, such as NS2 and OMNeT++, offer IEEE 802.16 libraries and some degree of development community support. However, for several reasons, as we shall discuss, we developed our own *deep* simulator – a discrete-event simulation model of an IEEE vertically integrated wireless Internet scenario. In particular, we concentrate our effort on the fixed IEEE 802.16 WMAN (802.16-2004), simulating admission control and scheduling processes exactly.

Both the machine model and workload model play an integral part in obtaining useful performance data: Our machine model includes particular MAC and physical layer (PHY) functions of the standard, such as framing, adaptive modulation and coding, fragmentation, and so on, as well as the admission control and scheduling algorithms. For the workload model, we developed a Markov Modulated Arrival Process (MMAP) by combining existing traffic models of different Internet applications, such as VoIP, P2P, etc. Each application is associated with

one of the IEEE 802.16 traffic categories (TCs). The MMAP generates both connection– and packet level data, maintaining traffic volume ratios, as reported by previous studies of Internet application traffic volumes.

Performance metrics of delay and jitter are calculated per TC connection. This allows a comparison of the quality of experience (QoE) of an individual user for the duration of a connection. At the connection level, we report the blocking probability.

By simulating the RaCM with various admission control and scheduling configurations, we were able to show that there is a significant difference in performance when using different CAC and scheduler combinations. Although hardly surprising, it is still proof that one cannot simply consider either in isolation, as is done in various performance studies reported in the literature. This interdependent relationship should be considered when designing complementary admission control and scheduling algorithms.

ACKNOWLEDGEMENTS

Professor Kritzinger Thank you for your words of wisdom and for being there when the drama escalated (and when it didn't) – for simply being a friend.

Professor Iazeolla You've been on board with me and Prof Kritzinger since the start of this MSc. What I have learnt from you is most certainly priceless. Grazie mille!

Professor Tim Dunne, thank you for the informative and insightful discussion on statistical analysis of the results.

To those of you close to me, thank you for the support.

Dar es Salaam
2 December 2009

Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

Wireless local and metropolitan area network (WLAN/WMAN) technologies, more specifically IEEE 802.11 (or wireless fidelity, WiFi) and IEEE 802.16 (or wireless interoperability for microwave access, WiMAX), are well-suited to enterprise networking since wireless offers the advantages of rapid deployment in places that are difficult to wire. However, these networking standards are relatively young with respect to their traditional mature high-speed low-latency fixed-line networking counterparts. It is more challenging for the network provider to supply the necessary quality of service (QoS) to support the variety of existing multimedia services over wireless technology. Wireless communication is also unreliable in nature, making the provisioning of agreed QoS even more challenging.

Considering the advantages and disadvantages, wireless networks prove well-suited to connecting rural areas to the Internet or as a networking solution for areas that are difficult to wire. The focus of this study specifically pertains to IEEE 802.16 and the part it plays in an IEEE vertically integrated wireless Internet (WIN): IEEE 802.16 is a wireless broadband backhaul technology, capable of connecting local area networks (LANs), wireless or fixed-line, to the Internet via a high-speed fixed-line link. Figure 1.1 shows the WIN application where the IEEE 802.16 backhaul has a base station (BS) which connects multiple subscriber stations (SSs). Each SS acts as a gateway that provides backhaul access to a user environment (ENV) and ultimately, the Internet. In this study, the ENV is an IEEE 802.11 wireless network.

Resource and connection management (RaCM) is important in the IEEE 802.16 backhaul network because it controls the resources between the ENVs and the Internet. It therefore has a significant impact on the QoS of the services supported. Amongst providing other functions, the BS provides Internet gateway functionality for its SSs and implements two interfaces, namely the wireless IEEE 802.16 interface, connecting BS and SSs and, in our case, the fixed-line Internet interface that connects the BS to the Internet. The BS manages QoS by sharing resources between the multimedia connections requested by the users, which originate in the ENVs and the Internet. The network must enforce connection admission control (CAC) and scheduling protocols to maintain acceptable QoS for admitted connections. These connections carry typical Internet data, such as hypertext transfer protocol (HTTP) and Voice over Internet Protocol (VoIP) data, which are mapped onto one of the traffic categories (TCs) specified in the IEEE 802.16 standard. Only once a connection has been granted service may it start generating and sending its content data to its destination. Each content datum of a granted connection is associated with the TC that the connection subscribed to and receives service accordingly.

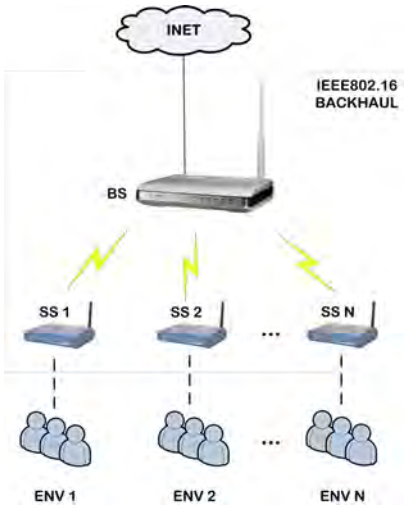

Figure 1.1: The vertically integrated wireless Internet scenario with the IEEE 802.16 backhaul

1.1 PROBLEM STATEMENT

IEEE 802.16 specifies a connection-oriented MAC and scheduling services that connections subscribe to in order to promote better QoS management within and across IEEE 802.16 networks. However, the standard does not specify any scheduling or CAC algorithms, leaving this up to the network operator to decide. This gives implementers the opportunity to create performance advantages and provides a rich field of research and performance analysis.

As already pointed out, admission control and scheduling have a significant impact on the QoS of the network, where both are responsible for sharing the available resource amongst user connections. While the scheduler manages resource at the packet-level, the CAC manages connection-level resource sharing. From this, the question arises:

> *What are the roles of, and the relationship between, the CAC and scheduler in the IEEE 802.16 fixed network?*

IEEE 802.16 is still relatively recent and receives a great deal of attention from both industry and research communities. Hence, in this dissertation, we investigate how admission control and scheduling jointly influence the QoS in the IEEE 802.16 Internet backhaul application. Understanding the synergism between the CAC and scheduler will aid the design of better admission control and scheduling algorithms and ultimately lead to more effective QoS control within the network.

An appropriate traffic model is needed to avoid under- or over-estimation of system performance. Very little data exist of these networks and therefore, the next question is

> *How can IEEE 802.16 connection-oriented workload be modelled?*

The dissertation will show by means of a simulation study, that there exists a synergy between the IEEE 802.16 CAC and Scheduler when it comes to maintaining QoS. This relationship should be exploited for the design of

joint CAC and scheduling mechanisms, resulting in better RaCM design.

1.2 DISSERTATION CONTRIBUTION

This dissertation makes three contributions to the state of the art of RaCM and IEEE 802.16 performance modelling for QoS analysis.

- **Scheduling and CAC Analytical Framework**
  The analytical framework [26], presented in Chapter 3, describes the explicit roles of, and the relationships between, the IEEE 802.16 Scheduler and CAC. These are described in context of the protocol layering of the standard. Data flow through these layers and components are described. The framework can be used to develop IEEE 802.16 network performance models for each of the operation modes, described in Section 2.1.1, and is used to develop the simulation model in this study.
- **IEEE vertically integrated wireless Internet discrete-event deep simulator**
  IEEE 802.16 is a very complex technology. Therefore, to be absolutely sure that it is modelled correctly in every detail, we developed a discrete-event simulator where we were sure of the implementation, settings of the various parameter values, and much else. Our simulator investigates the performance of the scenario shown in Figure 1.1. The analytical framework, described in Chapter 3, is used to develop the system performance model, described in Chapter 4. Documented in Chapter 5, are the specific CAC, schedulers and workload model implemented. The simulator development, described in Appendix A, generates both connection– and packet-level performance data used to analyse the QoS of the IEEE 802.16 backhaul.
- **Connection-oriented Workload Model**
  Even though workload modelling is not the focus of this study, a connection– and packet-level workload model was developed. Traffic models, based on related work, such as by Walters [64] and He [23] for HTML and VoIP traffic, respectively, abstract the connection request process and the packet-level behaviour of specific applications that should subscribe to one of the TCs, described in Section 2.1.2. The packet-level model is a one-dimensional Markov chain that determines which underlying traffic model is generating traffic, while connection-request generation streams generate admission and deletion requests in parallel for the different traffic models.

1.3 DISSERTATION OUTLINE

This dissertation is organised as follows.

Chapter 2: Background

In this chapter, we provide a detailed description of the IEEE 802.16 standard. Since the standard specifies different network modes, our configuration is identified; only the relevant parts of the standard are detailed in Section 2.1. In Section 2.2, the existing literature in scheduling and admission control for IEEE 802.16 networks are surveyed and the *state of the art* is presented. In Section 2.3, performance modelling techniques are highlighted and, in Section 2.3.2, efforts in workload modelling for wireless and Internet workloads are surveyed.

Chapter 3: Analytical Framework

In the analytical framework [26], we consider both the CAC and scheduler separately; we identify the re-

spective roles to describe the explicit relationship between these components. In Section 3.3, we describe data flow through both components and the IEEE 802.16 protocol stack. BS and SS model architectures are presented in Section 3.4, after which the extensions of the framework to all other modes of operation, identified in Chapter 2, are described.

Chapter 4: Performance Model

This chapter presents the software engineering artefacts. The abstraction of the actual system (given in terms of requirements) is translated into software artefacts (the various high-level models), continually being refined. By ensuring that all these models capture the features of the system, the simulator should be valid, provided the system implementation is correct.

In particular, the analytical framework is first extended to include the fixed-line Internet interface at the BS. The system performance model (SPM) is consequently derived and the abstracted time division duplexing frame is then discussed. The SPM is then used to develop separate network of queues (NoQ) models for the various parts of the SPM, where after these NoQ models are combined to form one NoQ model to base the simulation model development upon. Lastly, in Section 4.4, we discuss and motivate the development of a *deep simulator* for performance analysis of this study. Section 4.5 identifies the performance metrics to be observed in the context of the NoQ.

Chapter 5: System Components

In this chapter, we present the design, implementation and testing of the admission control and scheduling processes. Each component implements the respective component interface(s) specified by our simulator. Our simulator development is reported in Appendix A.

Chapter 6: Steady-state Analysis

This chapter presents a mathematical analysis of system utilisation of the uplink and downlink wireless channel servers. The analysis is used to select parameters that should ensure the system is operating in steady-state conditions and both uplink and downlink utilisation is unity. The analysis is used to maximise of both link utilisations.

Chapter 7: Experimentation

This chapter first highlights the hypothesis and proposes how we aim to prove it. Thereafter, we describe our approach to data analysis. Finally, the results are presented and discussed.

Chapter 8: Conclusion and Future Work

This chapter concludes this study and mentions possible future work.

CHAPTER 2

BACKGROUND

IEEE 802.16-2004, also referred to as fixed WiMAX, is the WirelessMAN™ standard for broadband wireless communication capable of spanning metropolitan areas. IEEE 802.16e-2005, mobile WiMAX, specifies the medium access control (MAC) and physical (PHY) layer amendment for combined fixed and mobile operation in licensed bands. Since mobility is not in the scope of this project, IEEE 802.16-2004 will be the focal point of this work.

2.1 IEEE 802.16 PRELIMINARIES

The standard specifies two air-interface protocol layers, namely the medium access control layer (MAC) and the physical layer (PHY). Figure 2.1 shows the data/control plane of the protocol stack, showing the various MAC sublayers and service access points (SAP), where data protocol units are communicated between adjoining layers through the SAPs.

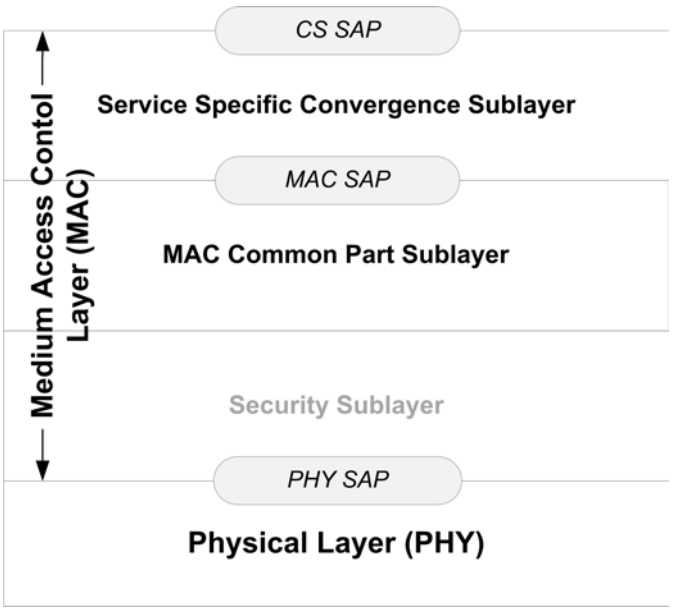

Figure 2.1: IEEE 802.16 air-interface protocol stack, taken from the 2004 standard [27]

The MAC, detailed in Section 2.1.2, consists of the service-specific convergence sublayer (SS CS), the MAC common part sublayer (MAC CPS) and the security sublayer, where the security sublayer is out of the scope of this work and is therefore ignored. The standard-defined scheduling services and the various data units that pass through the MAC, are then described.

The IEEE 802.16 standard specifies five different air interfaces. These are summarized in Table 2.1. In Table 2.1, adaptive antenna system (AAS) exploits more than one antenna to improve the coverage and the system capacity. Similarly, spacetime coding (STC) is a method employed to improve the reliability of data transmission in wireless communication systems using multiple transmit antennas. STCs rely on transmitting multiple, redundant copies of a data stream to the receiver in the hope that at least some of them may survive the physical path between transmission and reception in a 'good enough' state to allow reliable decoding. *Mesh* refers to the network architecture and automatic repeat request (ARQ) has its usual meaning. Line of sight (*LOS*) and non-LOS (*NLOS*) mean just that. The WirelessMAN-SC™ interface PHY specification, described in Section 2.1.3, is suitable for our application and is the PHY[1] we choose to model.

Designation	**Applicability**	**Options**	**Duplexing options**	**Operation**
WirelessMAN-SC™	10 – 66GHz		TDD/FDD	LOS
WirelessMAN-SCa™	Below 11GHz licensed bands	AAS/ARQ/ STC	TDD/FDD	NLOS
WirelessMAN-ODFM™	Below 11GHz licensed bands	AAS/ARQ/ Mesh/STC	TDD/FDD	NLOS
WirelessMAN-OFDMA™	Below 11GHz licensed bands	AAS/ARQ/ STC	TDD/FDD	NLOS
WirelessHUMAN™	Below 11GHz licensed exempt bands	AAS/ARQ/ Mesh/STC	TDD	NLOS

Table 2.1: Air interface alternatives

Four modes of operation, called *operation modes*, are identified. These modes are a combination of the topology that the network is operating in and the connection admission grant paradigm implemented.

2.1.1 Operation Modes

A network can operate either in a point-to-multipoint (PMP) or a mesh (MSH) topology. A PMP network, as shown in Figure 2.2, resembles a single-hop star topology and is managed by one centralised base station (BS), responsible for sharing the wireless medium amongst, and relaying traffic between, its associated subscriber stations (SS). SSs may not communicate directly with one another if the network is operating PMP.

A MSH network is characterised by a multi-hop environment where SSs not only communicate directly with the BS but also directly with each other. The MSH may either be centralised, as shown in Figure 2.3, or distributed. In a centralised MSH, resource management is the sole responsibility of the BS whereas, in a distributed MSH, control is shared between two or more BSs.

Since IEEE 802.16-2004 is connection-oriented, a connection admission controller (CAC) must be implemented at the station(s) managing resource allocation, i.e. the BS(s). Two admission paradigms exist, namely grant-per-

[1]Henceforth PHY is assumed to refer to the WirelessMAN-SC™ interface

Figure 2.2: Point-to-multipoint topology

Figure 2.3: Centralised mesh topology

connection (GPC) and grant-per-subscriber-station (GPSS). GPC admission controllers receive bandwidth (BW) requests (BWR) from SSs for individual connections and resources are allocated to each connection by the system scheduler, located at the BS(s). The scheduler informs each SS of individual connection schedules and the SS adheres strictly to this schedule.

Systems that implement GPSS allow an SS to make BWRs for an aggregated group of connections. Moreover, the BS scheduler informs each SS of its allocated resource and the SS is responsible for sharing its resource amongst the connections it manages. GPSS systems are more scalable than GPC systems. However, in GPSS the SSs are required to implement an additional scheduling algorithm to share its aggregated bandwidth (BW) grant amongst the associated connections.

Considering the topologies and admission paradigms described, four operation modes are therefore identified, each representing a topology-admission paradigm combination. The *operation modes* are

1. PMP topology with GPC admission control
2. PMP topology with GPSS admission control
3. MSH topology with GPC admission control
4. MSH topology with GPSS admission control

The WirelessMAN-SC^{TM} PHY supports both PMP and MSH topologies. GPC is not supported. As Chapter 4 reports, we model the PHY in operation mode 2, i.e. PMP, GPSS. In Chapter 4 the BS CAC, the uplink (UL) and downlink (DL) BS schedulers and the UL SS scheduler are discussed.

2.1.2 Medium Access Control Layer

Service-Specific Convergence Sublayer

The SS CS communicates network protocol data units (NPDUs) with the upper network layer and MAC service data units (SDUs) with the lower MAC CPS through the CS SAP and the MAC SAP, respectively. NPDUs are mapped onto particular connections and delivered to the MAC CPS as SDUs by the sending entity, as shown in

Figure 2.4. At the receiving entity, SDUs are received from the MAC CPS by the SS CS and the original NPDU is extracted and delivered to the network layer again.

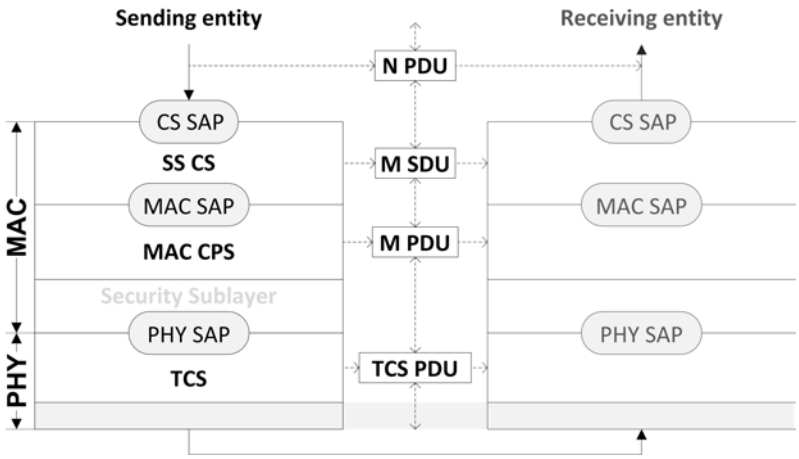

Figure 2.4: Data units showing their flow and relationship in context of the IEEE 802.16 MAC and PHY

Two SS CSs are specified by the standard, namely asynchronous transfer mode (ATM) CS and the packet CS. The relevant CS here is the packet CS since it is used for all packet-based protocols, particularly Internet Protocol (IP) which is the protocol our model is concerned with.

MAC Common Part Sublayer

The MAC CPS communicates SDUs with the upper SS CS and MAC Protocol Data Units (PDUs) with the lower-lying PHY (after passing through the security sublayer) through the MAC SAP and PHY SAP, respectively.

The MAC CPS defines fragmentation and concatenation features, shown in Figures 2.5 and 2.6, respectively. Fragmentation is where, at the sending entity, an SDU larger than a certain length is divided (fragmented) into smaller parts (called fragments), where each fragment becomes the payload of a separate MAC PDU. Once the receiving entity has received all fragments of an entire SDU, the fragments are joined (defragmented) and the original SDU is recovered. Concatenation, on the contrary, is where, at the sending entity, multiple SDUs are be joined together (concatenated/packed) to form one MAC PDU. This MAC PDU, on entering the MAC CPS of the receiving entity, is broken into its individual SDUs and these recovered SDUs are separately delivered to the SS CS. The PHY profile specifies that both fragmentation and concatenation is mandatory but that these features may be turned off on a per-connection basis. In this study, fragmentation will be used for all connections to limit the maximum size of data PDU. Concatenation will be ignored.

Scheduling Services

IEEE 802.16-2005 defines five scheduling services, called *traffic categories* (TC), to which connections may subscribe. Internet traffic, typically hypertext transfer protocol (HTTP), Voice over Internet Protocol (VoIP), Video Streaming, peer-to-peer (P2P) and file transfer protocol (FTP) in wireless networks, must be categorised into these five TC. The four TCs, defined in IEEE 802.16-2004, are extended by including the extended real-time Polling Service (ertPS) for the mobile system. Mention of ertPS is included here for completeness only but it is not considered for the system in this study. The five TCs are the following:

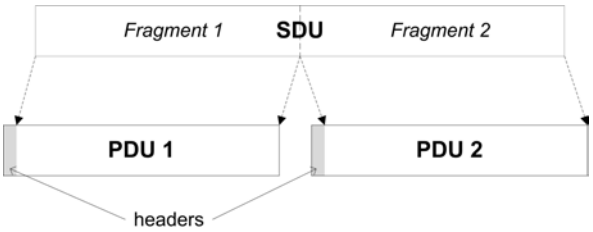

Figure 2.5: Fragmentation of one large MAC SDU into multiple smaller MAC PDUs

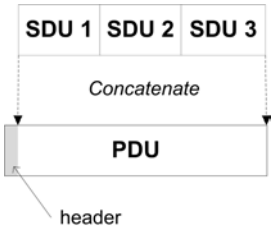

Figure 2.6: Concatenation of multiple small MAC SDUs into one large MAC PDU

- *Unsolicited Grant Service (UGS)* is designed to support real-time applications with strict delay requirements. These are applications that generate fixed-size data packets on a periodic basis, such as T1/E1 and VoIP.

- *Real-time Polling Service (rtPS)* is designed to support real-time applications with less stringent delay requirements. These applications generate variable-size data packets on a periodic basis, such as moving pictures experts group (MPEG) streaming video.

- *Non-real-time Polling Service (nrtPS)* is designed to support delay-tolerant data streams with minimum rate requirements. Different from rtPS, nrtPS connections have to utilize random access transmit opportunities for sending bandwidth requests. The nrtPS is suitable for Internet access with a minimum guaranteed rate and for ATM GFR connections.

- *Best Effort (BE)* is designed to support data streams for which no minimum transmission rate is required and therefore may be handled on a space-available basis, such as HTTP. Neither throughput nor delay guarantees are provided. The SS sends requests for bandwidth in either random access slots or dedicated transmission opportunities.

- *Extended real-time Polling Service (ertPS)*, added in 802.16e-2005 (or Mobile-WMAN), supports real-time applications where the applications require guaranteed data rate and delay. This service is for applications that would typically, in 802.16-2004, subscribe to the *rtPS* service even though they may behave similarly to *UGS* traffic at times, such as VoIP with silence suppression.

A summary of typical applications subscribing to each of these TCs, taken from the standard, are shown in Table 2.2. Since the MAC is connection-oriented, connections that wish to receive service from the network need

Category	*Typical application*
UGS	E1 transport, VoIP (without silence suppression)
ertPS	VoIP (with silence suppression)
rtPS	MPEG video
nrtPS	FTP with guaranteed minimum throughput
BE	HTTP

Table 2.2: Typical applications of each TC, taken from the 2004 standard [27]

to request subscription to one of the TCs. Should the request for admission be successful, the connection should receive service based on the TC it is subscribed to. Therefore, data transmitted over the network are BWRs, for requesting new or service change per connection, and connection data, the actual connection content.

Data Units

The data units already mentioned in this section are the NPDU, the MAC SDU and the MAC PDU. Additionally there is the optional transmission convergence sublayer (TCS) PDU, situated in the optional TCS of the PHY that accepts variable-length MAC PDUs through the PHY SAP and forms fixed-length TCS PDU. The TCS PDU is included in the data units description for completeness but is not modelled since its purpose is beyond the scope of this work. Figure 2.4 shows both the flow from network to air relationships between the four data units and their position in the IEEE 802.16 protocol stack for both sending and receiving entities. The process description to follow is simply reversed in the case of the receiving entity.

On the sending entity's side, NPDUs are accepted through the CS SAP by the SS CS to form the MAC SDU. As shown in Figure 2.7, the NPDU is packed into an SDU and associated to some connection. The payload header suppression index (PHSI) is shown in the figure.

Figure 2.7: MAC SDU format

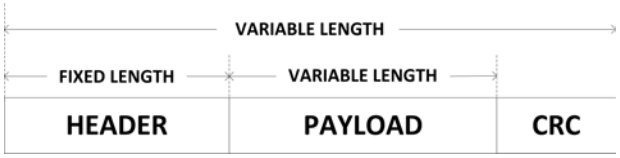

Figure 2.8: MAC PDU format

The variable-length MAC PDU is delivered to the TCS of the PHY through the PHY SAP as the fixed-length TCS PDU. Each TCS PDU consists of a pointer to the start of the first next MAC PDU of the current TCS PDU, as shown in Figure 2.9.

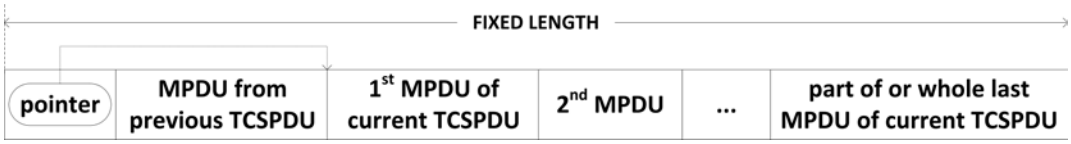

Figure 2.9: TCSPDU format

The scheduler must decide which MAC PDUs to place next into the TCS PDU. Once the MAC PDUs are place in the TCS PDU, they are transmitted. The TCS PDU is not modelled explicitly. In our study we model MAC PDUs transmitted over the medium once they have been selected for transmission.

2.1.3 WirelessMAN-SC™ Physical Layer

The PHY can only support LOS communication and implements either TDD or FDD operation in the 11 to 66 GHz frequency bands. It allows both PMP and MSH topology but only supports GPSS. The duplexing option modelled is TDD, since it appears to be the more popular choice than frequency division duplexing (FDD).

Framing

The IEEE 802.16-2004 standard defines a frame as

> "A structured data sequence of fixed duration used by some PHY specifications."

Transmission over the PHY is framed in both UL and DL according to the TDD frame structure, as shown in Figure 2.10. Each TDD frame consists of n physical slots (PS), where each PS consists of four modulation symbols. The value of n, as shown in Equation 2.1, is fixed for all frames during system operation and is a function of the symbol rate S_R and frame duration chosen by the network operator. The symbol rate $SR = \frac{1}{T}$, where T is the symbol period of the communications system, is in turn a function of the BW BW and the roll-off factor β, as shown in Equation 2.2.

$$n = \frac{SR \times frame\ duration}{4}\ PSs \tag{2.1}$$

$$SR = \frac{BW - 0.088}{1 + \beta} \tag{2.2}$$

The roll-off factor, β, is a measure of the excess BW of the filter, i.e. the BW occupied beyond the Nyquist BW of $\frac{1}{2T}$. If we denote the excess BW as Δf, then β can be calculated using Equation 2.3.

$$\beta = \frac{\Delta f}{(\frac{1}{2T})} = \frac{\Delta f}{\frac{S_R}{2}} = 2T\Delta f \tag{2.3}$$

Table 2.3, taken from the standard, shows baud rates and channel sizes using Nyquist square-root raised, cosine pulse shaping [4], with a roll-off factor of $\beta = 0.25$. The recommended frame duration is 1 ms and n is shown for various baud rate and modulation type combinations. The 3 modulation coding schemes (MCSs) allowed by the standard are QPSK, 16-QAM and 64-QAM, where a symbol represents 2, 4 and 6 bits, respectively.

The modulation type and the symbol rate (symbols per second) therefore play a role in determining system bandwidth. The specified permissible frame durations (0.5, 1 and 2 ms) determine the number of physical slots in a frame. The frame duration thus does not influence the capacity of the channel, only the capacity of a frame.

The TDD frame is logically divided into two sub-frames, for DL and UL transmissions, with an adaptive sub-frame boundary. DL bandwidth is defined with the granularity of one PS while UL bandwidth is defined with the granularity of one minislot (MS), where one MS is 2^m PSs ($0 \leq m \leq 7$). The standard also specifies that, in the DL, TDM bursts may be transmitted with different robustness profiles: QPSK being more robust than 16-QAM, 16-QAM being more robust than 64-QAM. These robustness profiles also apply to the TDMA transmission periods in the UL. For example, data begins with QPSK modulation, followed by 16-QAM, followed by 64-QAM. For simplicity, one burst profile is assumed for the DL TDM and UL TDMA bursts in this study, 16-QAM and QPSK, respectively, since multiple burst profiles imply a variable bandwidth from frame to frame. If so, arbitrarily choosing a channel bandwidth of 25 MHz and a frame duration of 1 ms, an overstated channel capacity of 80 Mbps (see Table 2.3) is calculated.

Note that the standard specifies that the portion of the DL frame used for transmitting the MAPs must be sent using QPSK modulation for maximum likelihood of error-free reception. Moreover, there are preambles that must be sent at the beginning of each new SS transmission on the UL for synchronization. Also, the ULMAP should be larger for a larger number of SSs scheduled to transmit during the frame.

In the model we shall ignore the synchronization effects and simply assume that a frame lasts 1 ms. Assuming the same duration for the DL and UL subframes and that the MAPs sizes in relation to these sub-frames are significantly small, the effective frame size is $\frac{(40+80)}{2} \times 10^6 \times 10^{-3}$ bits. That is, each frame is of maximum size

60 Kb, and will be considered as the "chunks" of data removed (or placed into) the MAC memory buffers by the physical transmitter/receiver at the rate of 10^3 per second.

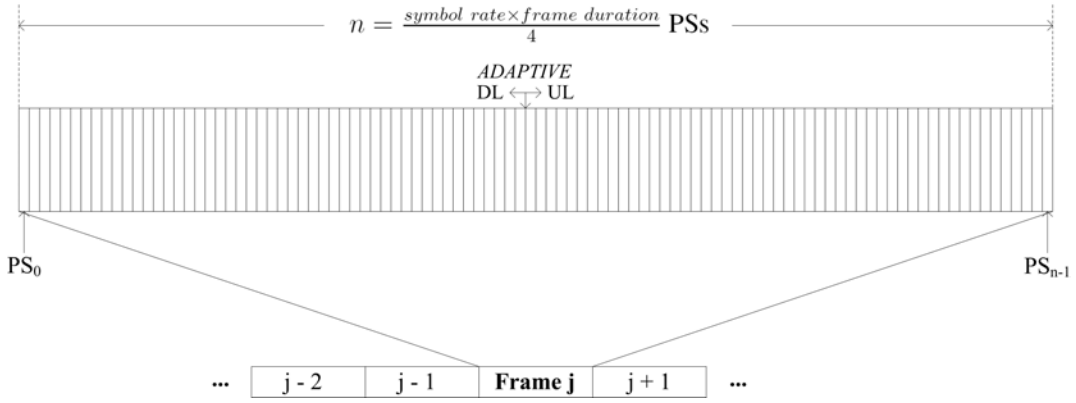

Figure 2.10: TDD frame structure taken from the standard

Channel size (MHz)	Symbol rate (MHz)	Bit rate (Mb/s) QSPK	Bit rate (Mb/s) 16-QAM	Bit rate (Mb/s) 64-QAM	Recommended Frame Duration (ms)	Number of PSs/frame
20	16	32	64	96	1	4000
25	20	40	80	120	1	5000
28	22.4	44.8	89.6	134.4	1	5600

Table 2.3: Baud rates and channel sizes for a roll-off factor of 0.25 taken from the 2004 standard [27]

Functional Frame Periods

As illustrated in Figures 2.11 and 2.12, a variety of periods are identified within the logical structure of the TDD frame. Each period serves a specific function understood by all parties in the network. These functional frame periods of interest are

- DL and UL MAP information,
- DL TDM bursts,
- UL management and bandwidth request contention, and
- UL TDMA bursts.

The transmit/receive transition gap (TTG) and receive/transmit transition gap (RTG) are periods during which no modulated data are transmitted, allowing BS and SSs to switch between transmit and receive states. Additionally, the DL sub-frame has as its first functional period the DL burst preamble, and each UL TDMA burst is prefixed by an SS transition gap (SSTG). The these periods are used for synchronisation of BS and SSs and are not considered in this study. The DL burst preamble, MAP information, DL TDM bursts and TTG make up the DL sub-frame, shown in Figure 2.11, while the UL contention, UL TDMA bursts and RTG constitute the UL sub-frame, shown in Figure 2.12. The maintenance opportunities period is used for general management of network entities, such as initial ranging of an SS, and is also not considered in this study.

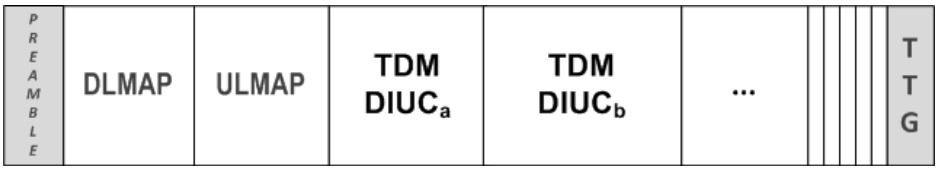

Figure 2.11: DL subframe periods

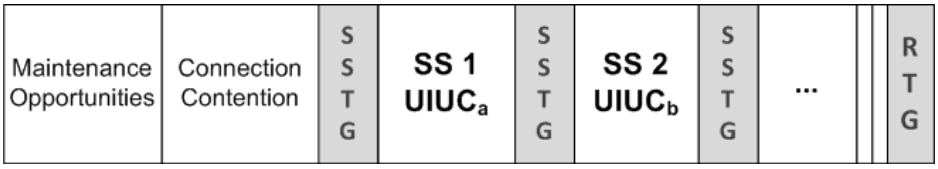

Figure 2.12: UL subframe periods

The DLMAP and ULMAP both consist of information elements (IEs) that, in the case of the DLMAP, describe the DL MCS to be used and when the DL profile is to be in effect. In the case of the ULMAP, IEs indicate the UL TDMA profiles, indicating whether the profile is either a connection contention period or an SS UL transmission opportunity. SS UL transmission opportunities, for purposes of this study, indicate which SS may transmit, the MCS to be used and when the transmission may occur. The DLMAP is approximately 64 bits (excluding 4 padding bits when necessary) plus the the variable length chain of DL IEs, where each IE is 20 bits long. The ULMAP is a approximately 56 bits (excluding 4 padding bits when necessary) plus an additional 32 bits per UL IE.

During the DL TDM burst, the BS transmits data per connection from the BS buffers to the SSs. Each SS evaluates the SSID in the PDU header to determine which belong to itself. During connection contention, SSs transmit BW request PDUs to the BS in contention mode, employing a truncated exponential backoff process to manage collisions. SSs transmit data PDUs, in a polling-based manner, along the UL during UL TDMA bursts assigned to the individually. In this study, it is assumed that no collisions occur when BWRs are transmitted.

Backhaul System Profile

For each air interface in Table 2.1, the standard specifies a set of system profiles. Each profile specifies MAC and PHY parameters separately.

The WirelessMAN-SC^{TM} air interface is the PHY investigated in this study and hence the basic packet MAC profile[2] applies. It specifies that fragmentation and concatenation features are mandatory but are optional per connection. As mentioned before, in this study, fragmentation is used for all connections; it ensures a maximum length for a MAC PDU. However, concatenation is ignored for all connections.

The relevant mandatory PHY parameters for the 25MHz WirelessMAN-SC^{TM} PHY TDD profile[3] are given in Table 2.4.

[2] *profM2* in the standard
[3] *profP1t* in the standard

Parameter	Value
Operation mode	TDD
Frame duration	1 ms
DL modulation	QPSK and 16-QAM
UL modulation	QPSK
Roll-off factor	0.25
Symbol rate	20 MBaud
PS per frame	5000 PSs

Table 2.4: Relevant mandatory profile (*profP1t*) parameters as specified by the standard

2.2 STATE OF THE ART

It is widely accepted that, in the case of traditional fixed-line networks, scheduling and admission control both play an integral role in providing acceptable QoS to users. Arguably, due to the scarce nature of wireless resource and standard-specific details of the technology, these play an even greater part in QoS-management in 802.16 wireless networks. Moreover, these are co-responsible for maintaining QoS. Therefore, recent efforts in scheduling and admission control development are surveyed not only in isolation but also as a singular cooperating toolset: the resource and connection management (RaCM) unit.

2.2.1 Scheduling

Scheduling is a management activity aimed to optimise a system's performance given certain performance criteria. In computer science, specifically in the field of communication networks, a wide variety of schedulers exist [5, 36, 31], each exploiting specific system features and attempting to solve a particular set of optimisation problems. Schedulers are developed with different objectives in mind, such as ensuring that packets are delivered with the minimum delay. Maintaining QoS is integral to the success of wireless networks, especially when the network intends to support the wide variety of multimedia applications used in the Internet – as the backhaul studied here does. It is in part the responsibility of the scheduler to maintain QoS: Many schedulers have been developed for maintaining QoS in similar, such as Asynchronous Transfer Mode (ATM) [51], technologies. As in ATM, a scheduler is designed specifically for the network it is to maintain the QoS for. To adapt previously designed schedulers for IEEE 802.16, such as those found in ATM networks, designers must consider not only similarities between the technologies but also take note of the differences, such as variable PDU length, in the case of IEEE 802.16, versus fixed cell size, in the case of ATM. Such differences, no matter how subtle they may appear, could result in the development of an inadequate scheduler for the IEEE 802.16 network. Similar to ATM, IEEE 802.16 supports multi-class service differentiation and thus, given more than one service category, the scheduling algorithms might aim to, say, minimise the delay and optimise the utilisation of a specific, or multiple, service categories.

Another consideration that must be made regarding IEEE 802.16 scheduling is the subschedulers that make up the system scheduler. In a PMP IEEE 802.16 network, three subschedulers are responsible for resolving BW contention and determining the transmission order of user data amongst all network stations, namely the BS UL scheduler, BS DL scheduler and SS UL scheduler, distinguished by Cicconetti *et al.* [14]. UL BW is shared amongst the admitted connections (GPC) or the individual SS admitted aggregated requests (GPSS) by the BS UL scheduler. In the case of GPC, the SS UL scheduler selects data for transmission from the SS buffers according to the ULMAP (predetermined by the BS). In the case of GPSS however, data per connection must be selected

during the SS aggregated connection grant period (specified in the ULMAP). The latter is achieved by the SS UL scheduler. The BS DL scheduler, selects data from the BS buffers and packs these into the DL frame; the BS DL scheduling algorithm determines the data selected.

Since the SS and BS UL schedulers are either respectively responsible for the order of data arrivals at the BS buffers, the UL scheduler(s) and DL scheduler must manage QoS at the packet-level for the entire network. It is therefore vital that one is not mislead to believe that, since one UL scheduler outperforms another, the UL scheduler(s) being compared are an acceptable choice for QoS management within the network when studied in complete isolation from the DL scheduler. Ali *et al.* [3] reports a comparative study of UL schedulers in PMP WiMAX networks however, even though they mention the significant relationship between CAC and scheduling, no consideration is made for the BS DL scheduler. Consequently, they report the average delay at an SS rather than average delay across the entire network. This is problematic: The UL scheduler may be effective along the UL but it may prove too difficult a task (maybe even impossible) to design a practical solution for the DL scheduler. The UL scheduler(s) may, for example, cause a respectively small average delay at an SS, which may in turn result in an unacceptably large average delay at the BS. Ultimately this combination may lead to poor end-to-end QoS experienced by the user. The UL and DL schedulers should therefore be designed to complement each other since these must cooperatively assist to maintain QoS.

Extending the reach of the taxonomy described by Ali *et al.* [3] by including DL scheduling algorithms, scheduling algorithms can be classed into three categories, namely

- homogeneous,
- hybrid, and
- opportunistic algorithms.

Most traditional schedulers are classified to be homogeneous algorithms. These are the traditional communication schedulers and typically do not change based on dynamic input parameter sets. The combination of homogeneous schedulers, through various arrangements, are then classified as hybrid schedulers. Opportunistic schedulers may be extensions of either homogeneous or hybrid algorithms but need not be. They exploit network conditions (such as channel status) by employing a dynamic scheduler that adapts itself to suit these changing network conditions.

Many of the familiar schedulers found in traditional fixed-line networks are located in the class of homogeneous schedulers. Considering the QoS parameters specific to IEEE 802.16 TCs, latency-rate schedulers [59] prove a popular and sensible starting point for selecting such an algorithm. In fact, Cicconetti *et al.* [14] identify a combination of deficit round robin (DRR) BS DL and SS UL schedulers and a weighted round robin (WRR) BS UL scheduler from this class of schedulers. They found that, in their performance study of a network operating these schedulers under two traffic scenarios, the average delay of UL traffic is higher than that of DL traffic. They attribute this finding to, *inter alia*, overhead introduced by physical preambles since all SSs with admitted connections are scheduled to transmit in each UL subframe. Their main contribution is, however, that it is possible for these traditional fixed-line schedulers to be used to provide QoS guarantees to the service differentiated IEEE 802.16 traffic. Furthermore, with these schedulers employed and by considering additional traffic scenarios which include the four IEEE 802.16-2004 scheduling services, they show that rtPS is very robust and adheres to multimedia delay requirements [15].

Another homogeneous scheduler, as identified by Ruangchaijatupon *et al.* [53], is the earliest deadline first (EDF) scheduler. The work conserving nature of this scheduler makes it an appealing choice since the aim of scheduling is not only to maintain acceptable QoS for all types of connections but also to maximise BW utilisation of the wireless medium. Also, as Ferrari and Verma [20] show, a general multi-class version of EDF is well-suited for scheduling real-time traffic in wide area networks.

Combining a variety of homogeneous algorithms result in hybrid schedulers, as studied by Wongthavarawat and Ganz [67]. By selecting homogeneous algorithms to schedule SS UL per TC and employing a strict priority scheduler to switch between these subschedulers, their study shows that their scheduling and CAC solution provides QoS support for all TCs. They select EDF for rtPS, similarly motivated as before. For nrtPS they employ a weighted fair queueing (WFQ) [18] scheduler that ensures fairness amongst nrtPS connections and maximises BW utilisation since WFQ is work conserving. Since there are no QoS parameters associated with BE traffic, a first in first out (FIFO) scheduler is employed to manage BE scheduling.

Chen *et al.* [11] also employ a hybrid scheduler in their proposed service flow management architecture for TDD operation. In their study, they consider not only admission control and scheduling, but also buffer management – where buffer management is beyond the scope of this study because it is a low-level operating system function and, although important, is very much dependant on the manufacturer. They study the performance of the network with both DL scheduler and CAC cooperating. However, in their study, they primarily focussed on the DL scheduler. Similar to Wongthavarawat and Ganz, they propose a hierarchical scheduler with EDF rtPS, and WFQ nrtPS schedulers for both UL and DL scheduling. For BE traffic, Chen *et al.* employ round robin (RR) UL and DL schedulers. The RR scheduler is more complex than FCFS (employed by Wongthavarawat and Ganz) but should not have any impact on QoS since BE does not make any QoS demands in IEEE 802.16. They select a deficit fair priority queue (DFPQ) to schedule between the multiple TCs and find that their service flow management strategy can meet QoS requirements of the existing TCs in terms of BW and fairness. No mention is made of the delay and jitter QoS experienced by the applicable service classes supported by the network.

Opportunistic schedulers are more aware of the context and dynamic nature of the environment than homogeneous and hybrid schedulers. Attempting to exploit the network conditions may result in better scheduling decisions and ultimately lead to higher BW utilisation and more effective QoS management. Rath *et al.* [52] develop an opportunistic scheduler by enhancing the homogeneous DRR scheduler in an attempt to satisfy multi-class delay requirements, called the opportunistic DRR (O-DRR) scheduler. The algorithm tries to limit the number of SSs scheduled on the UL by specifying a set of SSs that may be scheduled, say $set\ S$, and then schedules SSs in the frame by selecting from $set\ S$. It is their polling algorithm that exploits the present conditions of the network, i.e. determines the set of SS may be scheduled.

Tang *et al.* [61] choose to extend the class of hybrid schedulers by proposing an opportunistic scheduling architecture based on three criteria. The first is service differentiation, achieved by employing various schedulers, such as WFQ and WRR, for the different scheduling services, in the same way hybrid schedulers were described. The second aims to exploit the fact that the DL/UL frame boundary is dynamic, i.e. DL and UL frame sizes may vary on a frame-by-frame basis. If it becomes necessary to transmit more traffic along the UL, which may be the case when there is more priority traffic in need of service along the UL, then more of the frame could be allocated to UL transmission than DL. Thirdly, they propose that the status of the SS should be a criterion for scheduling traffic per station, i.e. SS differentiation based on the SS priority. The authors suggest that allocation of BW at the BS may be dealt with as an optimisation problem and they provide potential objective functions. Their efforts

in proposing an architecture for scheduler design are not investigated further by means of a performance model. Their architecture is included in this survey merely to illustrate how a hybrid scheduler may be used as the basis for developing an opportunistic scheduler.

Liu *et al.* [37] take a cross-layer design approach to developing their opportunistic scheduler. They claim that their scheduler is highly flexible and scalable, offers low implementation complexity, uses BW efficiently and maintains acceptable QoS. Ali *et al.*, in their study of UL schedulers, compare the Liu *et al.* scheduler to a variety of schedulers from all categories and find that this cross-layer scheduler performs poorly. However, taking into account that Ali *et al.* make no consideration of the DL scheduler, it is not entirely clear how well this scheduler will perform with a complementary DL scheduler and CAC configuration. Furthermore, Ali *et al.*represent cross-layer schedulers using this single cross-layer design and therefore no fair conclusion, regarding the performance of cross-layer designed schedulers in general, can be drawn. They conclude that to address the poor performance of this scheduler, a modification is necessary. Their conclusion highlights the tendency of opportunistic schedulers to become more complex than homogeneous and hybrid schedulers.

Considering the frame structure of IEEE 802.16, Sayenko *et al.* [55] propose an opportunistic scheduler based on the RR scheduler, motivated by the simplicity and speed of implementing such a scheduler. Their algorithm considers connection parameters, traffic priority, class type, BW request or queue size and modulation and coding scheme (MCS). Their scheduler is independent of the particular PHY implemented, an appealing feature. It is clear from this work that achieving generality of the scheduler may come at a cost, such as algorithm complexity. However, complexity can be reduced, as Sayenko *et al.* do, by selecting a simple homogeneous scheduler as the platform from which to develop an appealing opportunistic scheduler. Complexity is undesired since it results in a practical problem; where the duration of a frame is 1 ms, it is arguably the case that 1000 scheduling decisions can made per second.

Sayenko *et al.* [55] suggest that IEEE 802.16 should comprise three major stages. First, the scheduler must allocate the minimum number of slots required by each connection to maintain acceptable QoS. Thereafter, should there be unused slots, these may be shared amongst connections and lastly, timing requirements may be managed though the ordering of, and grouping into bursts of, the assigned slots. They claim that, with the primary aim of scheduling in IEEE 802.16 being the provision of basic QoS, the first stage is mandatory. However, in their work, they distinguish between basic and premium QoS, which is related to revenue functions and not the immediate concern of this dissertation. Tang *et al.* [61] similarly incorporate a revenue cost function by suggesting a classification of SSs as having either Bronze, Silver or Gold status and differentiating between these station accordingly.

The variety of aforementioned schedulers describes the progress being made regarding schedulers for UL and DL in IEEE 802.16 networks. By grouping these into three broad classes, their characteristics are better understood. Comparing schedulers belonging to the same class, or across different classes, are also easier to interpret since the benefits and short-comings of these classes are well understood. Furthermore, when designing a new scheduler, understanding the class that the scheduler belongs too will steer the designer away from making design choices that will lead to undesired results.

Considering the IEEE 802.16 network specifically, the three classes of schedulers are useful but still somewhat broad. There is room for improvement: The classification presented can be augmented by identifying the features a network provider may want its scheduler to exhibit. Table 2.5 lists the 5 features identified from the literature studied and gives the objective typically sought after by network providers.

Feature	Objective
Revenue	Maximise
User satisfaction	Fairness through differentiation
QoS	Maintain sufficient QoS
Utilisation	Maximise
Complexity	Low: simplicity, fast execution

Table 2.5: Scheduler features identified, showing the objective typically sought after

Since the network provider operates a business, whose main objective should be to make a profit, *revenue maximisation* is a feature typically expected of a scheduler. At first glance, it may seem that, since the scheduler manages traffic at the packet-level, it depends on the particular business model of the network provider whether this feature is relevant. However, this assumes that the scheduler has no control over connection-level revenue maximisation, which is arguably not the case, as shown by Sayenko *et al.* [55]and Tang *et al.* [61]. Furthermore, the network provider would want users (being either SSs, connections or the human users themselves) to feel like they are being treated fairly – to avoid losing them. Ideally, the scheduler should provide service to users according to their status in the network (as Tang *et al.* propose, where users are grouped per SS) in order to maintain *user satisfaction.*

It goes without saying that an important feature of the scheduler is maintaining *QoS*. However, if the CAC only admits 1 connection service to the network, it would be difficult not to maintain QoS. Therefore, the scheduler should strive to maintain sufficient QoS level when the network must support a workload that places it under stress. The QoS parameters that the scheduler maintains should correspond to the QoS the network provider guarantees. Maintaining QoS may also lead to a better revenue maximisation.

Maximising *utilisation* is not critical if revenue is maximised, QoS is maintained and users are satisfied. However, finding a scheduling solution that does all this is a very difficult (if not impossible) task. It is likely (but may not always be the case) that through increasing network utilisation, users may be more satisfied and revenue might increase. In the case of the WFQ [18] scheduler, the work-conserving behaviour ensures that, while any of the buffers are backlogged, the server may not be idle. If the scheduler was non-work conserving, the BW utilisation would decrease – as would the revenue (in the case of a packet-level business model) and arguably, the users satisfaction.

Lastly, the scheduler cannot be too complex. As mentioned before, time is short and the scheduling algorithm executes as many as 1000 times per second [55], as is the case in this study. Low *complexity* is desired, not only due to its high execution frequency but also for practical implementation purposes and to aid the design of other cooperating components, such as the CAC.

These features are not necessarily only applicable to scheduler design: As you will discover next, they are applicable to CAC design as well. This is not surprising since it has always been maintained that the scheduler and CAC cooperate to maintain QoS.

2.2.2 Connection Admission Control

Since IEEE 802.16 is a connection-oriented technology, a connection management mechanism is incorporated to maintain QoS. In part, the necessity of CAC arises from the general perception that it is more desirable to refuse a connection on attempt of establishment than to terminate the connection during the lifetime of its operation,

where the latter is a likely consequence of a continuation of QoS degradation.

In Section 2.2.1, it was suggested that CAC functions in tandem with scheduling to maintain acceptable QoS for the various TCs. It is however, necessary to distinguish clearly between the influence of the scheduler(s) and CAC in terms of QoS control. The IEEE 802.16 scheduler schedules connection content data passing through the MAC layer, i.e. QoS at a packet-level: It decides when the individual packets (PDUs, in the case of the IEEE 802.16 MAC) should be transmitted over the medium. On the other hand, the CAC evaluates BWRs at the network layer, i.e. QoS at a connection-level: It decides whether a new, or existing, connection may be admitted over, or receive a different BW allocation in, the network.

The rationale behind employing an admission control mechanism is illustrated by the following example: Assume a network with both an UL and DL capacity of 80 Mbps and assume one TC, having each connection with a minimum BW requirement of 1 Mbps. In the case where each connection utilises all of its minimum BW, the network can only support a maximum of 80 simultaneous connections without rendering the service incapable of proper functionality. The CAC therefore restricts resource usage when it foresees that the available resources are insufficient to provide the necessary resource for the connections receiving service at present. The threshold-based CAC that considers the number of connections with QoS demands, analysed by Niyato *et al.* [39], is a basic CAC that aims to regulate the number of QoS-requiring connections based on a threshold value determined and specified by the network operator. The authors present a Markov-model to analyse fairness and QoS performance by considering both CAC and scheduling in differentiated services (DiffServ) [8] wireless networks. Their scheduling mechanism is based on the packetised version of generalised processor sharing (GPS) [48] and they assume a two-state Markov channel error model. They find that, by separating the QoS-sensitive queues from BE traffic, QoS can be assured. Furthermore, they consider the issues and approaches to developing effective CAC for QoS provisioning in 4G wireless networks [40]. In their survey, they describe the different criteria that the CAC may use to make its decision but they primarily focus on cellular networks. An example that could be adapted to suit IEEE 802.16 technology, takes a pricing-based approach: Hou *et al.* [25] propose a cost function that considers maximisation of both service provider revenue and BW utilisation.

A fairness approach is also taken by Msadaa *et al.* [38], targeted at making efficient and fair use of available resources. BE connections are always accepted since there are no QoS requirements associated with this TC. For all non-BE connections, the maximum sustained traffic rate is used as a parameter to determine whether the request is to be granted. The authors seem to neglect the other deadline QoS constraints of connections, such as delay and jitter. However, Chandra and Sahoo [10] take CAC to the extreme case where the CAC first tries to schedule the new connection in the frame (together with the other currently admitted connections) to determine whether to accept the request. This is a complex process and hence, a time consuming approach and may prove impractical to implement. Also, if the scheduler does not schedule connections in the way that the CAC determines whether there is enough resource to schedule the new connection, the CAC may admit connections that will not receive the necessary QoS during its lifetime. The authors neglect to consider whether the accompanying scheduler(s) are a practical solution and to what extent these provide QoS. Furthermore, complications arise when attempting to schedule connection data explicitly: Packet lengths vary and channel error is unpredictable.

A less complex dynamic admission control scheme is proposed by Wang *et al.* [65]. TCs are prioritised, with highest priority given to UGS traffic and, as most CACs do, the BS sets aside a fixed amount of BW per flow. Degradation levels are introduced to manage the QoS provided to nrtPS traffic since the BW of this TC may vary between maximum and minimum BW requirement parameters. Wang *et al.* also neglect deadline QoS constraints.

When developing a CAC and schedulers, it is important to keep in mind the QoS demand parameters required by the various TCs. If a rtPS connection has some jitter QoS requirement, how can an rtPS connection be admitted with any degree of QoS jitter-guarantee?

2.2.3 Scheduler and Admission Controller Co-operation

Developing suitable CACs and schedulers for IEEE 802.16 networks not only demands careful consideration for these components in isolation but also for the cooperative effort made by these for the network to be capable of delivering QoS to the TCs supported. Each component is responsible for sharing the same wireless resources amongst the same users. It would be detrimental to the network's QoS if the QoS policies of these two components contradict each other. In fact, with complementary policies in place, QoS can be maintained to a larger degree.

In the literature surveyed in Sections 2.2.1 and 2.2.2, it is apparent that there is a general tendency towards considering either CAC or scheduling as the primary contributor to maintaining QoS. It is reasonable to claim that, since CAC and scheduling maintain QoS in tandem, development of these RaCM components should occur with equal attention paid to both components. Moreover, UL and DL scheduler(s) are often considered in isolation of one another. Ali *et al.* compares UL schedulers in a network where the CAC remains unchanged. Poorer QoS achieved by a scheduler when compared to another could be as a result of the complementary nature of the specific CAC. Arguments made by the authors about comparisons of the schedulers experimented with may be invalid since the CAC may be the reason for a scheduler under-performing.

RaCM component designers can take a minimalist approach to CAC and schedulers design – where researchers typically initially consider one QoS parameter, cater for service differentiation between two TCs, and so forth. These designs can then be developed further and become better suited to the system it must operate in. This evolutionary development allows for a better selection of features to include in the algorithm design.

Another benefit is that the minimalist approach allows developers to do away with the algorithms that are too complex, or rather, these unwanted features are not included in the algorithm design. A simplistic algorithm should execute faster than a complex one, which is important when some algorithm may need to execute, as explained before, at most 1000 times per second. Additionally, to reduce the complexity of opportunistic algorithms, they should be designed from the beginning using less complex algorithms. Dynamic functions should consider the performance criteria used to exploit present network conditions, where these criteria are determined by network status information variables that affect or describe the QoS parameters specified by the various traffic types.

2.3 PERFORMANCE MODELLING

A system consists of 2 components: the *actual machine* and the *actual workload*. The machine represents the hardware and software configuration of the system while the workload is the traffic load generated and/or supported by the machine. Both workload and machine play important roles in the resulting system performance thus, when developing a system model, both must be modelled with careful consideration. As shown in Figure 2.13, the *system model* comprises a *workload model* and a *machine model*, which are abstractions of the actual workload and actual machine of the *actual system*, respectively.

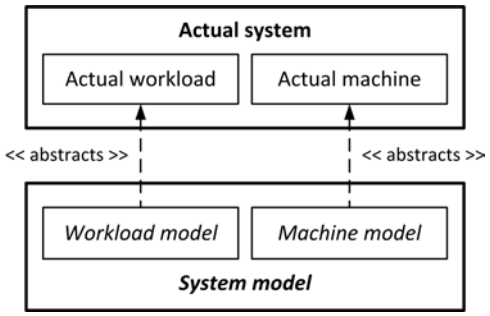

Figure 2.13: Abstraction of the actual system components to form the system model

2.3.1 Machine Model

As mentioned before, the machine model abstracts the hardware and software configuration of the system. There are 3 main techniques used in performance modelling, namely analytic modelling, simulation and hardware experimentation. However, these techniques may be combined when developing a performance model.

Analytic models are fast since they are mathematical formulae that provide instantaneous results. However, these models are usually more difficult to develop, more complex and involve a greater amount of (unrealistic) abstraction than other modelling techniques.

Hardware experimentation is an unrealistic choice because it is expensive, requires particular physical environments and configuration and is prone to hardware error. Additionally, it is difficult (and impossible in some cases) to isolate certain components of a system under study. However it allows for the most realistic model of network performance.

Lastly, simulation modelling involves mimicking the system using a computer program. Various techniques exist to improve the performance of a simulator in terms of memory usage and execution time [34]. Simulation allows one to obtain better performance measurement than analytic models since more realistic assumptions can be made. Also, simulation is a cheaper option than hardware experimentation since, typically, a single powerful machine is necessary to run the program.

2.3.2 Workload Modelling

Representing workload adequately and as accurately as possible is a crucial aspect of network performance modelling. If the workload model does not capture the salient features of the network traffic, one cannot expect useful performance data. It is typically the case that Poisson processes are used to model network arrivals, even though it has been shown on numerous occasions, that workload characteristics, such as inter-arrival time (IAT) and packet length, are not always simply exponentially distributed [49]. Frost *et al.* [21] and Adas [1] outline several traffic models used in broadband networks that capture salient statistical characteristics of the actual network traffic and categorises modelling techniques in light of the characteristics of traffic types or profiles.

Since the 802.16 backhaul application supports traffic carried over the wireless Internet, Internet traffic is relevant to this study. Recently models, by Klemm *et al.* [32] and Salvador *et al.* [54], abstract the IP traffic workload process using a Batch Markovian Arrival Process (BMAP) and a discrete-time BMAP (dBMAP), respectively, and Hernández *et al.* [24] present discrete-time heavy-tailed chains to model Internet traffic as a superposition of

discrete-time on–off sources, capturing the *self-similarity* and *long-range dependence* workload characteristics.

Instead of modelling workloads using analytic techniques, measured traffic datasets (called *traces*), such as the *Community Resource for Archiving Wireless Data At Dartmouth* (CRAWDAD) [69], can be used to for traffic generation. The trace is used as is (after processing of the raw trace) or representative distribution(s) (and their associated parameters) are determined to form part of an analytic model. An advantage of using a trace is that it samples the true workload of the network. However, whether the sample is representative of all traffic depends on various factors, such as trace duration, which time of day the trace represents, etc. A major drawback of a trace is that the data collected is very specific. Not only is it network-specific, but it may be recorded pre– or post-process, i.e., it may be collected either before or after certain system process(es) have taken effect. For example, the *UNC/FORTH Archive of Wireless Traces, Models and Tools* [47] makes available a university-wide wireless network trace – a collection of post-distribution coordination function (DCF) data, where the DCF is the medium access control protocol for the IEEE 802.11 wireless local area network technology. Additional drawbacks of a trace include the need for a large amount of post-processing, insufficient information available and an immense volume of data in the raw trace.

Since no IEEE 802.16 backhaul-specific traces are available at present, models of the different Internet services expected in this network are surveyed. In a study by Schulze and Mochalski [56], the volumes of different Internet services are surveyed: P2P is by far the largest constituent of Internet traffic. In 2007, Schulze and Mochalski [56] found that 69.25 %, 83.46 %, 63.94 %, 48.97 % and 57.19 % of Internet traffic in Germany, Eastern Europe, Southern Europe, the Middle East and Australia, respectively, was P2P traffic. A small percentage of Internet multimedia traffic was VoIP calls. Grouped together with Skype (P2P voice), only 0.92 %, 0.57 % and 0.51 % of Internet traffic in Germany, the Middle East and Australia, respectively, were voice calls. In the Middle East and Australia, video streaming is insignificant whereas, in Germany, it constituted 7.75 % of Internet traffic. Lastly, HTTP traffic made up 10.05 % and 26.05 % of Internet traffic in Germany and the Middle East, respectively. Traffic types not surveyed are either not relevant to this study or make an insignificant contribution.

Relevant to this study, P2P, VoIP, streaming video and HTTP traffic are considered in turn, focussing on both content data (at the packet-level) as well as connection-level models. Each of these Internet services subscribe to a specific IEEE 802.16 TC and together, constitute almost all Internet traffic: VoIP and real-time video streaming services subscribe to UGS and rtPS TCs, respectively. However, both P2P and web browsing traffic is considered to subscribe to the BE TC.

Peer-to-peer

In a peer-to-peer (P2P) network, content files (including real time data, such as telephony traffic) are shared between machines, called *peers*. Contrary to the *client-server* architecture, in pure P2P each and every peer operates as both client and server machines and may initiate a communication session. Examples of P2P applications are Napster [46], Kazaa [45], Gnutella [44] and BitTorrent [43], where BitTorrent traffic was most prevalent in the Internet in 2007 for all countries surveyed [56].

Accurate characterisation and analysis of P2P traffic have become more challenging with the development of modern P2P applications. Early P2P systems used TCP and fixed ports to communicate, making traditional port-based traffic monitoring and classification an effective and robust approach. However, contrary to these first generation P2P applications, present-day P2P systems disguise their traffic by continually changing between TCP and UDP connections and jumping ports. Traffic identification is thus a non-trivial task, not to mention the accuracy

attainable.

Perényi *et al.* [50] propose a heuristic P2P traffic identification method and characterise modern P2P systems using their method to obtain performance data. Robust P2P traffic properties were used to associate packets with the various P2P applications supported. They used their "trustiness" measure, describing the ratio of packets of a certain size belonging to a specific application, to conclude that the typical packet size for BitTorrent traffic is 128 bytes. Their results show that P2P packet sizes are not fixed however, no accurate attempt can be made since a large portion of traffic is unknown.

Distribution models for web and P2P connection-level behaviour were determined from a large edge network by Basher *et al.* [7]. They found that P2P IAT is heavy-tailed and, specifically, BitTorrent session duration is long-tailed whereas Gnutella session duration is heavy-tailed. Each characteristic was found to be best-represented by a piecewise continuous function, similarly to Klemm *et al.* [2], whereas Erman *et al.* [19] found that session IAT and duration were better modelled using a two-stage hyperexponential- and lognormal distribution, respectively. The distribution models determined by Erman for BitTorrent traffic are shown in Table 2.6, together with the associated parameter values, where the two-stage hyperexponential distribution PDF is given by Equation 2.4.

$$H_2(x) = p\lambda_1 e^{-\lambda_1 x} + (1-p)\lambda_2 e^{-\lambda_2 x} \tag{2.4}$$

Model	**Function/Distribution**	**Parameters**
IAT	Two-stage hyperexponential	$p = 0.6575$,$\lambda_1 = 0.0566$, $\lambda_2 = 0.3653$
Duration	Lognormal	$\mu = 8.16$, $\sigma = 1.33$

Table 2.6: P2P connection-level distribution models and associated parameter values, taken from Erman *et al.* [19]

Also considering the differences between web and P2P traffic, particularly Kazaa, Gummadi *et al.* [22] attribute the differences to Kazaa objects being immutable; web objects change more frequently than P2P objects and need to be 'fetched' repeatedly.

Voice over IP

A variety of encoding/decoding algorithms (codecs) exist for speech encoding, many of which are standardised by the *Telecommunication Standardization Sector* of the *International Telecommunications Union* (ITU-T)[4]. Of these standards, G.711 [29] is a pulse code modulated (PCM) high bit-rate (64 Kbps) ITU-T standard, used in telephone networks and supported by most VoIP providers, that results in high quality voice communication. To ensure delay remains as low as possible, no compression is performed by G.711. Compression processing can contribute significantly to delay, especially when 8000 byte samples are taken per second, which is the case for G.711.

When modelling voice traffic, the speech process is considered. For the unidirectional flow of data, two relevant events (or states) are identified and need to be modelled, namely the *talk-spurt* and *silence*. As detailed by Adas [1], this ON–OFF activity process is modelled by a two-state Markov model, where the ON state represents the talk-spurt and the OFF state represents the silence period: In the ON state, the speaker is speaking and voice data is generated. In the OFF state, no data is generated since the speaker is silent. As recommended by the ITU-T artificial conversational speech standard (P.59) [30], the sojourn time in the ON and OFF states are exponentially

[4]Available on-line: http://www.itu.int/ITU-T/, 4 May 2009.

distributed with mean values of of $\lambda_{ON} = 1.004$ s and $\lambda_{OFF} = 1.587$ s, respectively, with respective average talk-spurt and silence periods of 38.53 % and 61.47 %. They determined this information by averaging results reported by various relevant studies.

While in the ON state, packet-level data are generated. Voice generates constant bit rate (CBR) traffic and thus packet IAT and size are deterministic. The values for packet IAT and size depend on the codec used. As Seger [57] explains, for the G.711 codec, the $64Kbps$ stream comprises packets containing at least one 8 bit voice sample (payload). In addition to the payload, network overhead per packet requires a real-time protocol (RTP) header of 12 bytes, a user datagram protocol (UDP) header of 8 bytes and an Internet protocol (IP) header of 20 bytes, i.e. total header size of $12 + 8 + 20 = 40$ bytes. Since 8000 samples must be transmitted per second and considering the ratio of the payload versus the total header size, it is sensible then that multiple samples constitute a single payload, where the number of samples per payload varies, depending on the codec and parameters used. Seger and Chuah [13] mention payloads of 80 and 160 bytes for G.711 voice packets, respectively, and a packet IAT of 20 ms.

He [23] models the connection-level VoIP characteristics by fitting distributions to call IAT and duration data using a trace of a network serving both VoIP and transmission control protocol (TCP) traffic. Call IAT was found to be exponentially distributed while call duration followed the Pareto distribution. Furthermore, since different workload intensities were noted during different periods of the day, three sets of parameters were determined for selected times of the day. These parameters are shown in Table 2.7.

Time of day	**Call IAT parameters**	**Call duration parameters**
09:00 – 10:00	$\lambda = 6$ minutes	$\alpha = 1$, $\kappa = 0.547$
16:00 – 17:00	$\lambda = 5$ minutes	$\alpha = 0.88$, $\kappa = 0.6$
17:00 – 18:00	$\lambda = 10$ minutes	$\alpha = 1.21$, $\kappa = 0.472$

Table 2.7: VoIP connection-level parameters drawn from a VoIP trace, taken from He [23]

Real-time Video Streaming

Video streaming is a real-time application that includes applications for e-learning, video conferencing and Video on Demand (VoD).

Ideally, data should arrive and play out continuously without noticeable interruption. However, fluctuations in network conditions constrain this service's ability to perform. In an attempt to minimise the unwanted fluctuations, an adaptive streaming server may be commissioned to monitor the network conditions and adapt the quality of the stream to minimise interruptions, as described by Cranley and Davis [16].

Like for VoIP, different codecs are used to deliver streaming video over the Internet. Two techniques, namely CBR and variable bit rate (VBR), may be employed. CBR is used by applications requiring fixed data rates continuously available during a connection's lifetime and a relatively tight upper bound on transfer delay. Uncompressed audio and video typically employ the CBR technique. VBR encoding is designed to work optimally in high bandwidth scenarios and is especially suited for encoding content that is a mixture of simple and complex data. Fewer bits are allocated to the simple parts of the content while enough bits remain to produce good quality for the more complex portions.

Recent studies by Shin and Ryu [58] and Xu [68] model VBR real-time video service with the Pareto distribution. Shin and Ryu suggest that the time between frames, each frame made up of 8 video packets, is 100 ms. Video packet IATs follow the truncated Pareto distribution with a mean value of 6 ms, a maximum value of 12.5 ms and distribution parameters $k = 2.5$ and $\alpha = 1.2$. Packet sizes are also truncated Pareto distributed with a maximum value of 125 bytes and distribution parameters $k = 20$ and $\alpha = 1.1$.

Web Browsing

Studies, such as those by Walters [64] and Choi [12], identify similar distributions, for web browsing traffic, to each other. Walters, following Staehle *et al.* [17], developed an analytic workload model for individuals browsing the web from packet traces of web traffic over a fixed-line network. The model comprises the distribution functions of specified mathematical families and their derived parameters for workload information parameters, such as web client IAT.

Walters [64] determined that the Weibull and lognormal distributions best fit the trace for IAT and packet lengths, respectively, with the relevant packet-level parameters given in Table 2.8. As shown in the table, IAT parameters are the same for both UL and DL traffic however, are significantly different for packet sizes. DL packet sizes are significantly larger than UL packet sizes, as expected. IAT was measured at millisecond resolution and the packet size at byte resolution.

Component	**Distribution**	**Parameters**
Web client request IAT (UL & DL)	Weibull	$\gamma = 0.371$, $\alpha = 315778.506$
Non-cached web client response size (DL)	Lognormal	$\varsigma = 7.401$, $\sigma = 1.405$
Web client request size (UL)	Lognormal	$\varsigma = 5.883$, $\sigma = 0.331$

Table 2.8: UL and DL HTTP packet-level workload parameters

Furthermore, the connection-level parameters are given in Table 2.9. These parameters include the connection request IAT and connection holding time. The *browsing inter-session time* models the time between the end of a browsing session and the beginning of the next and was measured at minute resolution since the second resolution was found too small, with parameter values of larger than 15 minutes and a maximum of 480 minutes. Holding time was modelled by *the number of web user requests per browsing session* and *the number of web client request per user request*.

Component	**Distribution**	**Parameters**
Browsing inter-session time	Gamma	$\gamma = 0.645$, $\alpha = 102.342$
Number of web user requests per browsing session	Pareto	$\alpha = 1$, $\beta = 0.558$
Number of web client request per user request	Lognormal	$\varsigma = 2.155$, $\sigma = 1.377$

Table 2.9: HTTP connection-level workload parameters

CHAPTER 3

ANALYTICAL FRAMEWORK

Chapter 2 describes the IEEE 802.16 protocol layers and surveys the state of the art admission controllers and schedulers for such networks. In this chapter, an analytical framework is introduced for developing IEEE 802.16 performance models. It accounts for admission control and scheduling components but queue management and routing are not considered. It highlights the respective responsibilities and relationship between the IEEE 802.16 Connection Admission Controller (CAC) and scheduler, different from the work done by Cicconetti *et al.* [14]. Their location in the protocol stack and the data flow through these layers, through the CAC and scheduler, is described. High-level BS and SS model architectures are developed and it is explained how these architectures are suited to model all four operation modes described in Chapter 2. These model architectures are combined and extended in Chapter 4, reporting on the performance model developed.

3.1 CONNECTION ADMISSION CONTROLLER

The CAC is responsible for connection-level management of the network, either admitting or denying bandwidth (BW) requests (BWR). A BWR is either a Dynamic Service Addition (DSA) request, (a connection requesting admission to the network) or a Dynamic Service Change/Deletion (DSC/DSD) request. On arrival of a BWR, the CAC obtains information from the scheduler regarding the status of the medium, as shown in Figure 3.1. Based on these two inputs, the CAC evaluates the BWR and informs the scheduler whether the connection is admitted.

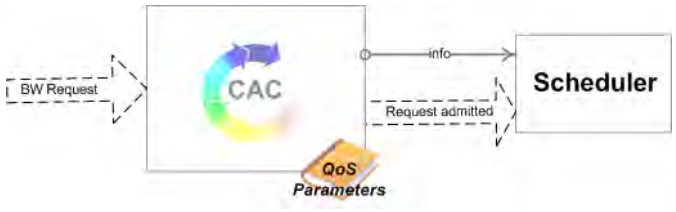

Figure 3.1: Roles of, and relationship between, the CAC and scheduler from the CAC's perspective

3.2 SCHEDULER

The scheduler manages up-link (UL) and down-link (DL) transmission (by the radio) by obtaining information from the scheduler's virtual queues [14] and MAC memory buffers, respectively. It then executes the scheduling

algorithms, for both UL and DL, shown in Figure 3.2. The virtual queues are memory buffers updated by the CAC for the scheduler to know amongst which connections to share the medium. The MAC memory buffers hold the actual data routed through the BS. ULMAP and DLMAP information is decided for the succeeding MAC frame MAP and is communicated as MAP information with the SSs.

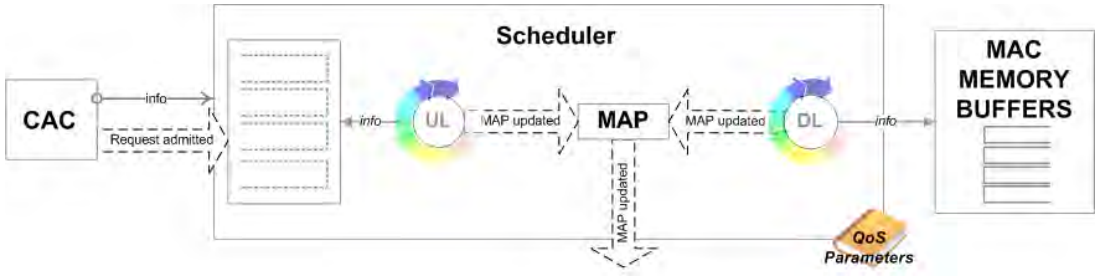

Figure 3.2: Roles of, and relationship between, the scheduler and CAC from the scheduler's perspective

3.3 Framework Outline

In the left-hand part of Figure 3.3, the protocol stack shows the data flow for UL and DL traffic arriving at and departing, respectively, from the BS. On the right-hand side, the relationship between the scheduler and CAC are shown as discussed above.

On the UL, information from the wireless medium enters the IEEE 802.16 Physical Layer (PHY), and the MAP maps incoming data or BWRs with a particular connection as scheduled in the previous DLMAP. This information enters the MAC layer, either being buffered there or not, depending on the implementation, and then passes through the Security Sublayer, MAC Common Part Sublayer (MAC CPS) and the Service-Specific Convergence Sublayer (SS CS), in that order. The data or BWR then enter the Network layer where the CAC would consider BWRs, as explained. Data, not destined for the BS, will be routed appropriately.

Information that does not have as destination the BS, i.e. information that must be routed to another station, together with information generated by admitted connections of the BS itself, passes down into the MAC layer, through the CS, CPS and Security Sublayer. Information entering the MAC CPS is queued in the MAC memory buffers. Information about these queues is, as mentioned before, used by the scheduler to allocate or re-allocate resources. Information enters the PHY from the MAC memory buffers according to the DLMAP and is transmitted over the wireless medium.

3.4 BS and SS Architectures

Figure 3.4 shows the BS model architecture of the framework. Frames arriving at the BS were previously scheduled on the UL as either BWR or data units to be forwarded on the DL or, less frequently, data destined for the BS itself (not indicated in the figure). A BWR is taken up by the CAC that decides whether to admit the it and, if so, will pass this information to the scheduler.

Information about these newly admitted BWRs as well as buffer lengths, traffic types, and so on, of current connections are used by some scheduling algorithm to update the ULMAP and DLMAP for the next transmission. The ULMAP essentially informs the schedulers at the SSs in the network of their turn to transmit on the UL, assuming TDD, while the DLMAP informs the SSs which data in the frame is destined for them.

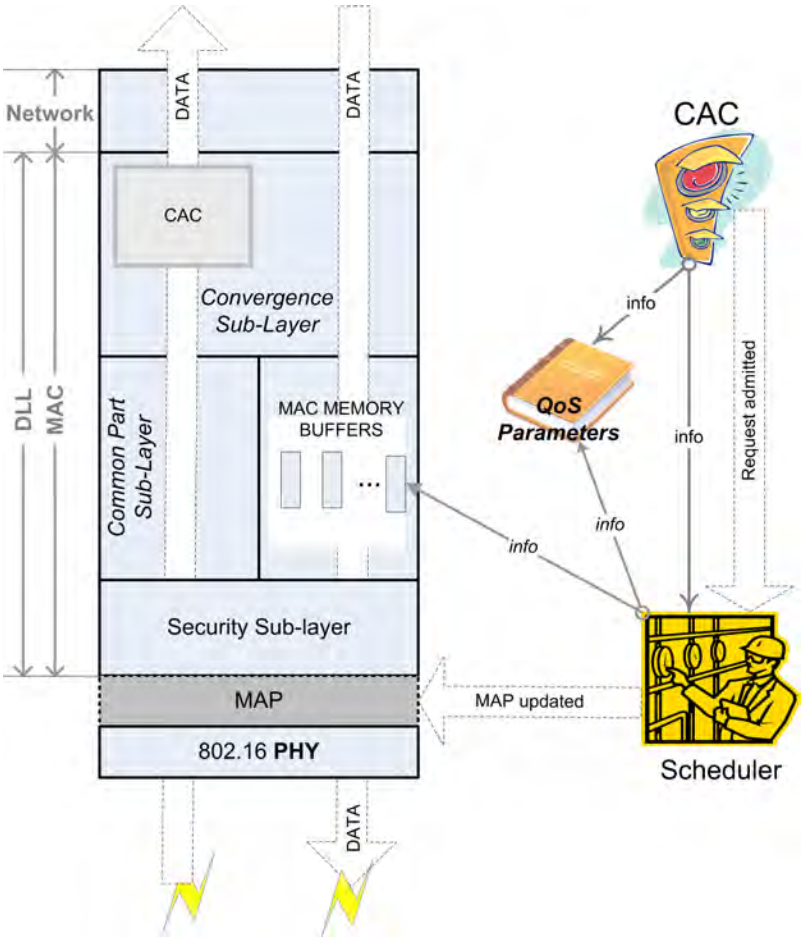

Figure 3.3: Information flow between the scheduler and CAC and data flow

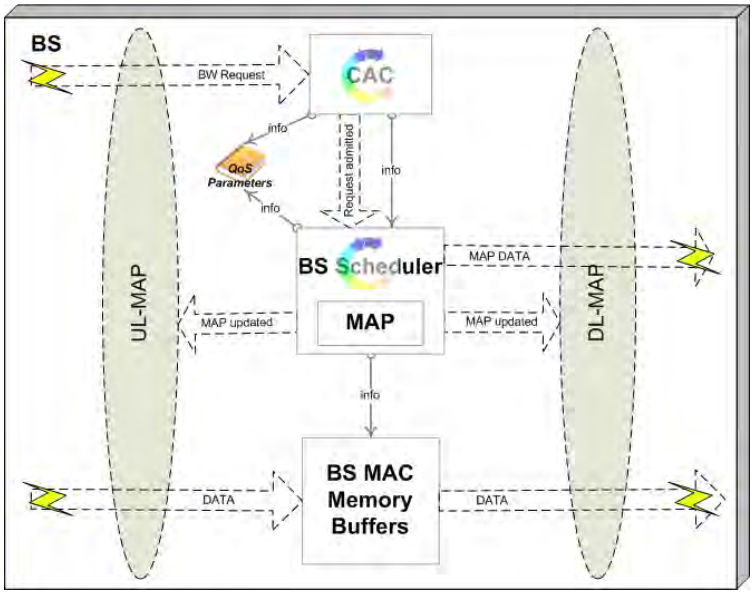

Figure 3.4: BS model architecture

The SS model architecture is illustrated in Figure 3.5. Information arriving at the SS, as transmitted by the BS on the DL according to the BS DLMAP, are either updates for the SS ULMAP, data to be routed through the SS, or data destined for the SS (not indicated in the figure).

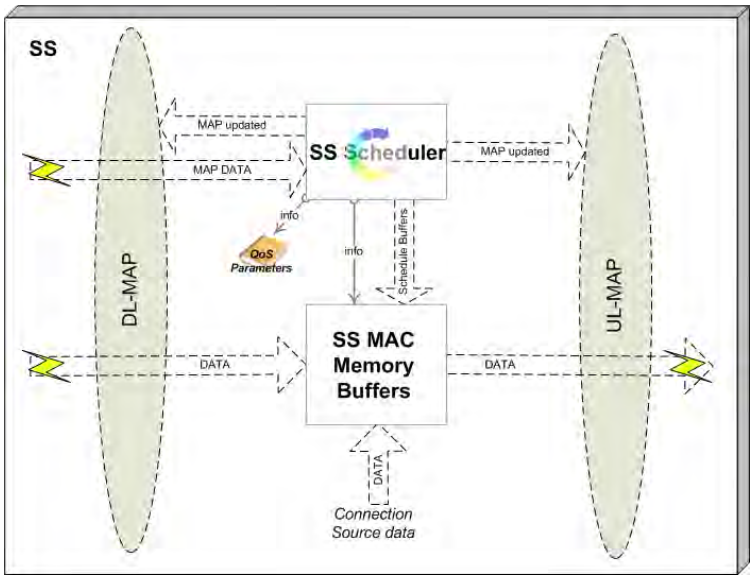

Figure 3.5: SS model architecture

The SS scheduler has not been described in detail because it adheres to the MAP specified by the BS. However, in operation modes 2 and 4 (described in Chapter 2.1.1), the SS still adheres to the MAP schedule as dictated by the BS, but must also manage connection transmission schedules internally since, in those modes, the SS has been allocated resource for a group of connections. The SS therefore has an *SS UL Schedule* to manage these connections – including the data to be routed. The SS MAC memory buffers are managed according to the ULMAP and additionally by the SS scheduler in operation modes 2 and 4.

3.5 MULTI-MODAL PROPERTY

The framework described here is for the BS operating in a PMP topology with GPC admission control (operation mode 1). The framework however, and the derived model architectures, are suited to developing models of networks operating each of the four operation modes: The framework can be extended trivially from GPC to GPSS admission control paradigm (i.e. operation mode 2), meaning that the SS scheduler algorithm will be more complex but the SS model would not change. The BS model would remain the same and the GPSS call admission may imply fewer BWRs. Extending from operation mode 1 to 3, i.e. from PMP to MSH, means that the scheduler and CAC at the BS will be far more complex and lead to worse performance at the BS. However, no structural changes need to be made to the BS model. Extending from operation mode 2 to 4 can be done in the same way as was done for extending from operation mode 1 to 3. Therefore the framework is suitable for developing performance models of IEEE 802.16 networks operating in any one of the four operation modes. The performance model that Chapter 4 describes, models PMP topology with GPSS admission control, i.e. operation mode 2.

CHAPTER 4

PERFORMANCE MODEL

This chapter reports the development of the system performance model (SPM) and its associated network of queues (NoQ) models to be implemented as a discrete-event simulation model. Much care has been taken in producing the various software engineering artefacts to ensure a valid, correct model. As the model is refined, the level of abstraction should become clear, as motivate for the various assumptions, not already motivated in the Chapter 2, are provided.

In this chapter, the Chapter 3 analytical framework is first extended to include a fixed-line Internet interface at the BS; in Section 4.1, the BS model architecture, proposed in Chapter 3, is modified and described. The SPM is then presented in Section 4.2. Also in this section, the abstracted TDD frame is described, detailing the various abstractions made. Subsequently, in Section 4.3, separate NoQ models are derived for various parts of the SPM and ultimately, are combined to form a single NoQ model to represent the entire system. Lastly, in Sectionřefsec:performancemetrics, before reporting the development of the discrete-event simulator, the performance metrics are identified and discussed with reference to the system model.

4.1 BS MODEL ARCHITECTURE FIXED-LINE INTERFACE EXTENSIONS

Since only the BS has fixed-line internet connectivity in the network, to extend the analytical framework to suite the application, i.e. to include a fixed-line interface, only the BS model architecture, shown in Figure 3.4, must be modified. Figure 4.1 shows the modified model architecture. Data arriving at the BS are treated exactly as before in the case of wireless arrivals. These wireless arrivals are placed in the MAC memory buffers to be forwarded to its destination, being either another SS or the fixed-line connection internet node. Data from a wireless source is represented by a broken arrow and, as shown in the figure, wireless arrival leave the memory buffers either over either one of the wireless or fixed-line DL interfaces.

Fixed-line arrivals, represented by a solid arrow in the figure, are also placed in the memory buffers but, differently from the wireless arrivals, fixed-line arrivals will always leave the memory buffers over the wireless interface. It does not make sense for fixed-line data to enter the backhaul network if its destination is not in one of the SS subnetworks.

As shown in the figure, BWRs are treated the same for both fixed-line and wireless requests when referring to

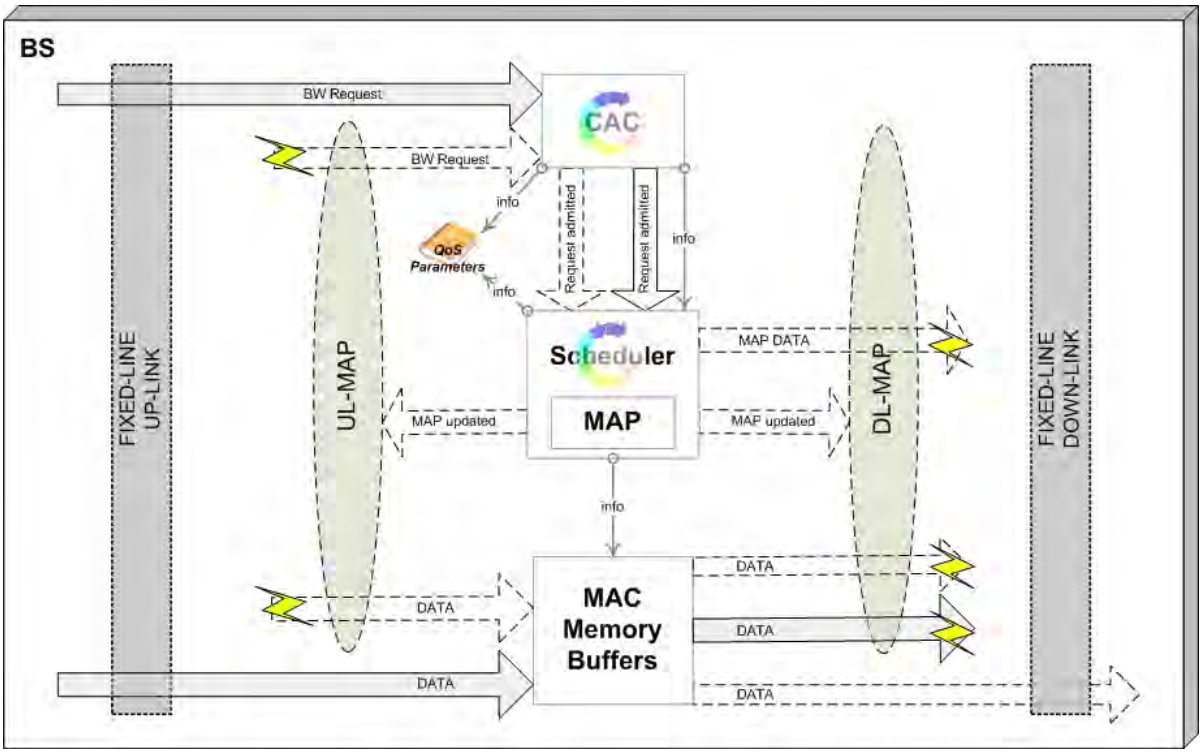

Figure 4.1: BS model architecture extended to include the fixed-line interface

data flow. The details of the CAC and schedulers are not represented by the model architecture.

4.2 SYSTEM PERFORMANCE MODEL

The system performance model, shown in Figure 4.2, presents a high-level abstraction of the networking application described in Chapter 1. The aim of the SPM is to identify and describe the network structure and management components chosen to be model. The abstracted functional frame shown in Figure 4.4 describes the frame abstraction, i.e. DLMAP, ULMAP, DL TDM, etc.

As mentioned before, the network carries BWRs and data per connection. Data arrive at each individual SS, where they are queued at the SS connection queues. An SS transmits data from these queues during its scheduled period of the UL subframe to the BS. The BS, in turn, receives data according to the ULMAP it communicated at the beginning of the current frame. Each SS employs its scheduling algorithm since the network is operating GPSS. The BS is therefore only responsible for scheduling each SS UL time and each SS must schedule its own per connection data transmissions. Data are transmitted from an SS to the BS over the wireless UL-server (W_{UL}) and from the BS to an SS over the wireless DL-server (W_{DL}).

The fixed-line input/output physical links (F_{in}/F_{out}) at the BS concurrently accept data arrivals from, and transmits data to, the internet (INET), respectively. Data is transmitted over servers F_{in} and F_{out} between the INET node and the BS at a very high data rate such that the time taken for data to be transmitted across F_{in} and F_{out} is close enough to zero that one may safely assume instantaneous arrival of data over each link.

In the system, BWRs arrive at an SS and are buffered there until the connection contention period, specified

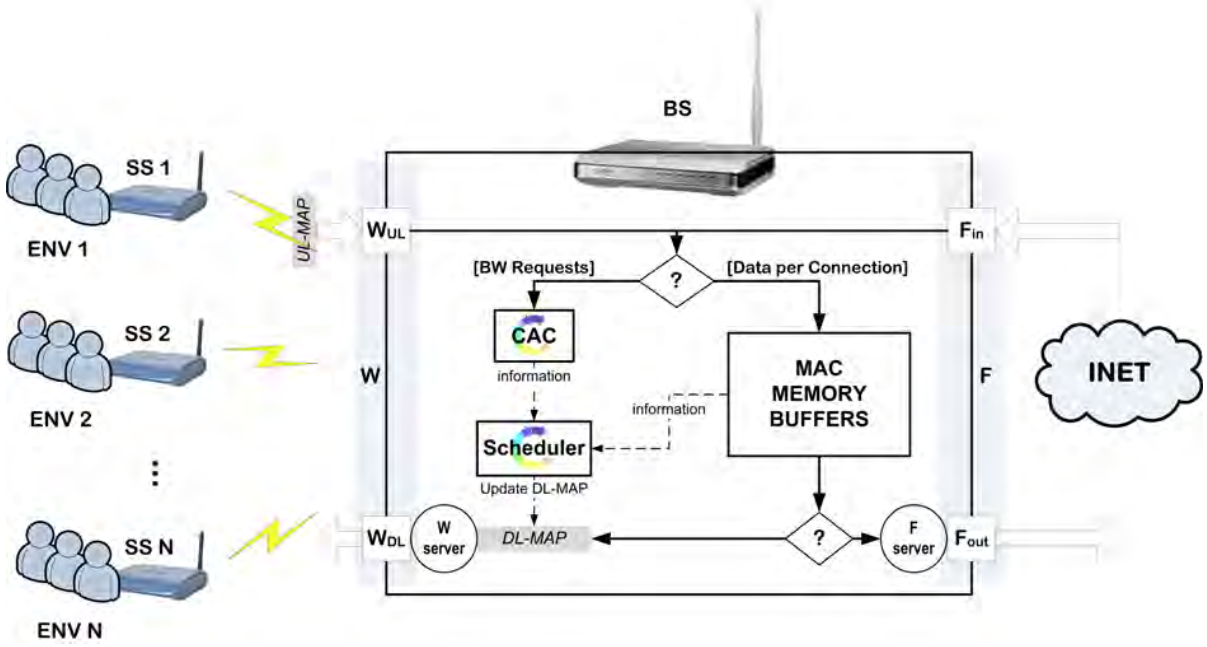

Figure 4.2: System performance model

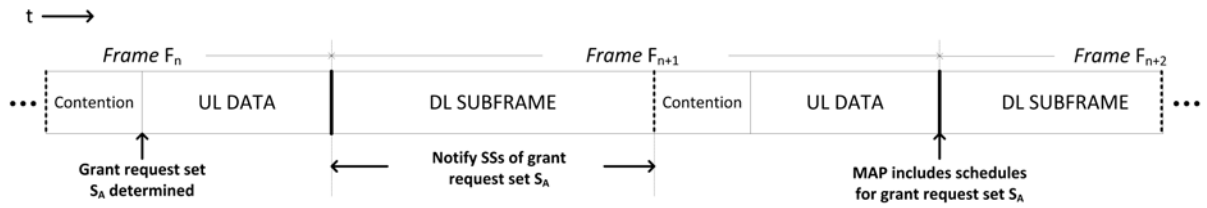

Figure 4.3: Request admission-notification process

in the ULMAP, starts. During this period, BWRs are transmitted to the BS according to a truncated exponential backoff process to help with collision avoidance and are queued at the BS CAC. In the model, BWR arrivals at the SS are forwarded directly to the BS CAC waiting line and no collisions of BWRs are assumed. Also, all BWRs in the CAC waiting line are evaluated at the end of the connection contention period. These assumptions are made since the actual process is too detailed to model in addition to everything else.

As shown in Figure 4.3, at the end of the connection contention period of the current frame F_n, the CAC processes BWRs and keeps the granted requests as connection set S_A. The CAC then informs the BS scheduler of its admission decisions during the current frame F_n. The BS scheduler does not use this information, i.e. set S_A, to calculate the new MAPs for the next frame F_{n+1}. Rather, the BS first informs the SSs during the frame F_{n+1} of the admitted connections in S_A and only uses grant request set S_A information to calculate the MAP for frame F_{n+2}.

The BS scheduler uses virtual queue information, together with the scheduling algorithm and QoS parameters, to determine the new ULMAP and DLMAP. These MAPs are transmitted from the W_{DL} to the SSs at the beginning of the next TDD frame.

Data per connection arriving at the BS are queued at the BS MAC memory buffers. The fixed-line output and

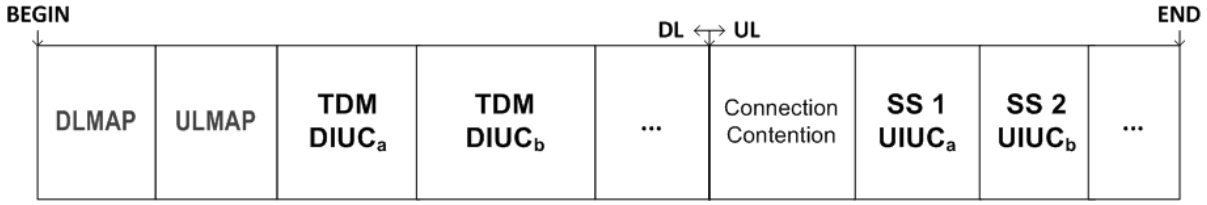

Figure 4.4: Abstracted functional frame

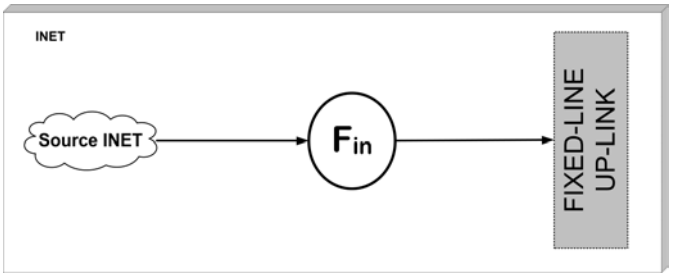

Figure 4.5: Network of queues model abstracting the INET fixed-line source and F_{in} server

wireless UL server (F_{out} and W_{UL}, respectively) each have their own associated logical waiting-line structures. However, in our model we assume that F_{out} is a high data rate link such that its service time is zero, resulting in no waiting-line at this server.

Taking into consideration our abstracted SPM, we abstract the functional UL and DL frames shown in Figures 2.11 and 2.12 of Chapter 2 and provide, in Figure 4.4, the abstracted functional frame. The functional periods shown in this frame (which we intend to model) are the MAPs, DL profiles, UL connection contention period and the individual SS UL profiles. This frame is stripped of all preambles and the maintenance opportunities period.

4.3 NETWORK OF QUEUES MODEL

In this section we develop four network of queues (NoQ) models that are combined to form the NoQ model our simulator is based on. All NoQ models show only one traffic category (TC) for a clearer explanation. Figures 4.5 and 4.6 show the NoQ models for the data sources, abstracting the internet UL and the UL resource sharing between the N SSs, respectively. As shown in Figure 4.5, the INET source generates data, i.e. BWRs and data per connection (data), served by the fixed-link UL server F_{in} and, since the fixed-link UL is high-speed, F_{in} is assumed to enforce no delay on data. The data arrive at the BS along the fixed-link UL interface. Figure 4.6 shows how SS sources generate data that are queued at the respective SS connection data queues. The wireless UL W_{UL} server serves the SS connection queues according to the ULMAP specified by the BS, where each SS executes the SS scheduler and serves its internally managed connection queues during the UL period allocated to it. Each SS transmits data along the wireless UL at a certain rate (specified in the IE of the ULMAP) and data arrive at the BS along the wireless UL interface.

Data per connection arrives at the BS wireless receiver from the SS NoQ model, shown in Figure 4.6. Wireless UL data has as destination either the fixed-line internet or another wireless SS. Data is therefore either queued at the MAC memory buffers for wireless SS-designated data or transmitted to the internet along the F_{out} fixed-line link. Data arriving from the internet at the BS along the F_{in} fixed-line interface, shown in Figure 4.5, has as destination some wireless SS and is thus queued at the MAC memory buffers.

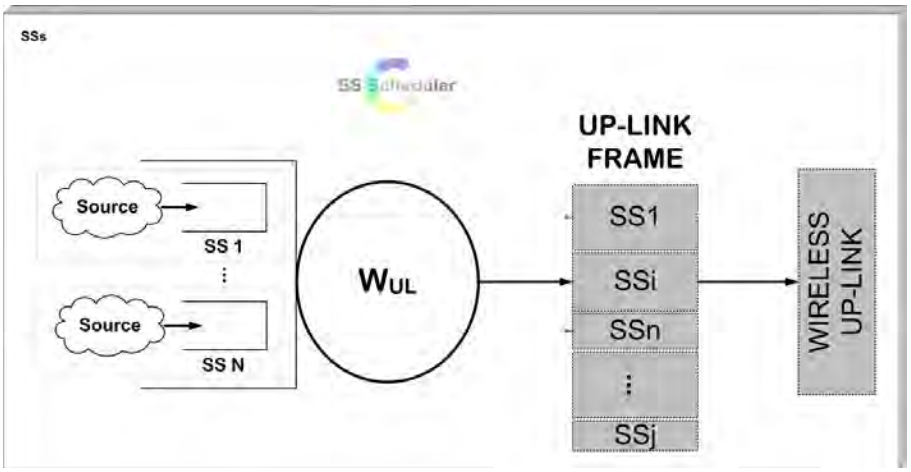

Figure 4.6: Network of queues model model abstracting wireless UL data at the N SSs per TC

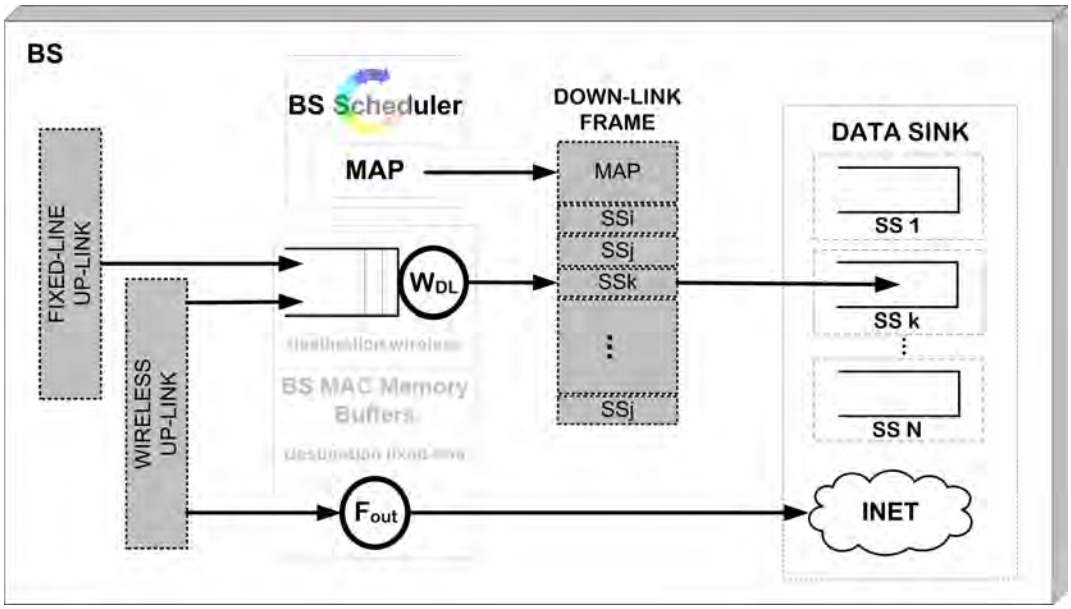

Figure 4.7: Network of queues model model abstracting both wireless and fixed-line data at the BS per TC

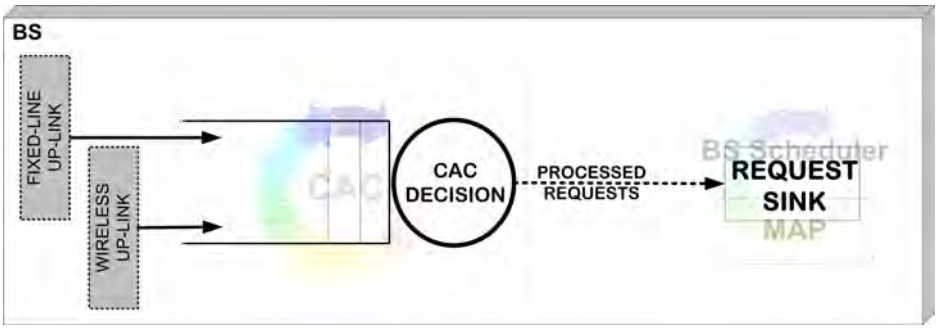

Figure 4.8: Network of queues model model abstracting both wireless and fixed-line BWRs at the BS per TC

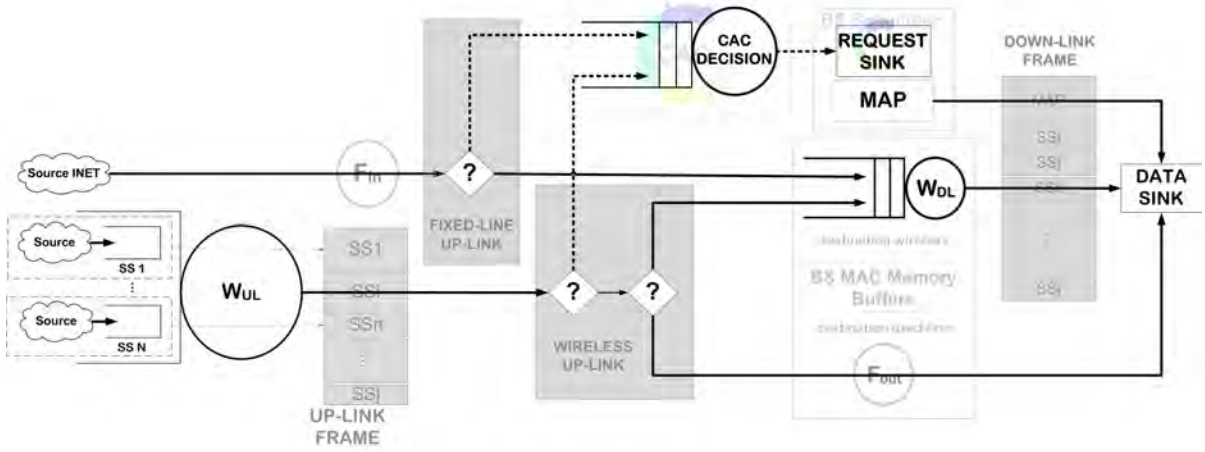

Figure 4.9: Network of queues model model abstracting both wireless and fixed-line data and BWRs at the BS per TC

During the DL, the BS transmits data from the MAC memory buffers along the DL, as shown in Figure 4.7. The BS scheduler is, as shown in the figure, responsible for generating the MAP information and transmitting it to the SSs.

BWRs are generated at both the SSs and internet sources. As described in Section 4.2 and illustrated with Figure 4.3, connections admitted and rejected during the current frame are first notified of admission or rejection during the DL of the next frame. Only after this period will the scheduler include these admitted connections in the MAP assembly decisions. The assumptions regarding BWR arrivals and evaluation have been described in Section 4.2 and hence we derive the NoQ model abstracting both wireless and fixed-line BWRs at the BS per TC, as shown in Figure 4.8. BWRs originating at the fixed-line and wireless sources are queued at the CAC. The processed requests are then forwarded to the BS scheduler.

By combining the NoQ models shown by Figures 4.5, 4.6, 4.7 and 4.8, the NoQ shown in Figure 4.9 is realised. This model represents the entire system to model as one NoQ model based on the framework model architectures, TDD frame structures and SPM described in Chapters 3, 2 and 4.2.

4.4 DEEP SIMULATION

There are many simulation-specific programming languages and environments that support the effective development of simulation, such as *inter alia* the OPNET Modeler [28],ns2 [41], QualNet [62] or OMNET++ [42, 63]. Alternatively, one can develop one's own simulator by implementing the simulation engine and components in a language such as Java or C++, i.e. develop a *deep* simulator.

It is generally accepted that simulation tools make simulation model development easier, faster and more effective and may include features such as experimental design, advanced reporting and exploitation of computer-architecture to improve execution speed. Users of these tools may encounter some obstacles while learning to use them at first. Thereafter, simple simulation models may be developed fast and efficiently. However, developing simulation models of a more complex nature is a much more challenging task; it requires users to have an expert understanding not only of the system but of the tool used as well.

By the mere fact that such tools need to be general, they present users with certain limitations, primarily due to the structure of the simulator in terms of the simulation paradigm followed and the language(s) used to realise the model. For simple models these tools are appropriate but for more complex ones, they are not. Users may want to (re-)configure the system to a larger extent and/or at a lower level, such as making changes to certain protocols fixed in the assumptions of the tool. As one example, OMNeT++ is a modular general-purpose simulation environment with graphical and statistical components and offers a wide range of software suites, such as its mobility framework. However, it is seldom clear which details of the network stack are being modelled and where the associated parameters may be found.

For proprietary reasons, commercial products may not always make clear the particular techniques used for obtaining statistically significant [34] performance estimates. Furthermore, should the modeller require non-conventional data analysis, as was the case for this study, or simply inspect the performance data during data analysis, proprietary tools may not necessarily provide such functionality.

Most importantly, the accuracy of a simulation model developed, using a simulation tool, is also questionable. A study by Cavin *et al.* [9] shows how different implementations of the same simulation model, each developed using a different simulation tool, may result in a significant divergence between the results obtained from these simulators. This is partly due to the different levels of detail provided for implementation and configuration. Simulation tools may prove very accurate if the abstraction-level and simulation paradigm supported suit the system being modelled. Otherwise developing a deep simulator may prove not only more accurate but also more useful by allowing the developer a greater degree of flexibility of manipulating the system model and allow the developer greater insight into the true nature of the system.

IEEE 802.16 technology and the vertically integrated wireless Internet scenario, described in Chapters 1 and 2, are particularly complex. Additionally, the study required specific abstractions to be made. We therefore developed a deep simulator for this study. The design, implementation and testing of our simulator are reported in Appendix A.

4.5 PERFORMANCE METRICS

The standard addresses the following QoS parameters[1]:

[1] page 702 of the IEEE 802.16-2004 standard

1. *Tolerated jitter* is defined as the maximum delay variation (jitter) of a connection.

2. *Maximum latency* defined the maximum latency between the reception of a packet by the BS or the SS on its network interface and the forwarding of the packet on its wireless(RF) interface.

3. *Maximum sustainable traffic rate* defines the peak information rate of the service. The rate is expressed in bits per second and pertains to the SDUs at the input to the Convergence Sublayer. If this parameter is omitted or set to zero, then there is no explicitly mandated maximum rate. This field specifies only a bound, not a guarantee that the rate is available. For the WirelessMAN-SC^{TM}interface the maximum value of this parameter is 80 Mbps.

4. *Minimum reserved traffic rate* specifies the minimum rate reserved for this service flow. The rate is expressed in bits per second and specifies the minimum amount of data to be transported on behalf of the service flow when averaged over time. The specified rate shall only be honoured when sufficient data is available for scheduling.

   The BS and SS shall be able to transport traffic and satisfy BWRs *for a service flow* up to its Minimum Reserved Traffic Rate. If less bandwidth than the its Minimum Reserved Traffic Rate is available requested for a service flow, the BS and SS may reallocate the excess reserved bandwidth for other purposes. The data for this parameter is measured at the input of the Convergence Sublayer. The aggregate Minimum Reserved Traffic Rate of all service flows may exceed the amount of available bandwidth. The value of this parameter is calculated from the byte following the MAC header HCS to the end of the MAC PDU payload. If this parameter is omitted, then it defaults to a value of 0 bits per second (i.e., no bandwidth is reserved for the flow).

   Neither IEEE 802.16-2004 nor IEEE 802.16-2005 specifies a minimum rate for a service flow.

From these parameters, delay and jitter are identified as performance metrics. However, to study the utilisation of the network operating a specific CAC and scheduler configuration, throughput should also be observed. Three performance metric are finally chosen to describe the packet-level performance of our system, namely

- throughput,
- response time, average end-to-end delay over the backhaul network, and
- jitter, variation in delay.

The simulation of the abstracted system, described in Sections 4.2 and 4.3, must produce our performance data. In our analysis, we focus on end-to-end quality of service across the backhaul network. The performance of the network is observed from the point at which data enters (i.e. as data arrives at the SS connection queues, or data arrives from the INET at the BS) until it exits the backhaul. The following is a step-by-step example detailing the various delays experienced by traffic being served by the network and is shown in the accompanying Figure 4.10.

- At SS_r, $r = 1, \ldots, N$, on connection C_{rj}, $j = 1, \ldots$, MAC PDU M_{rj} is queued at time t_i.
- At time t_{i+1}, after M_{rj} experienced a queuing delay σ^x_{rj}, $x \in \{UGS, rtPS, nrtPS, BE\}$, SS_r starts transmitting M_{rj}, i.e. $t_{i+1} = t_i + \sigma^x_{rj}$.

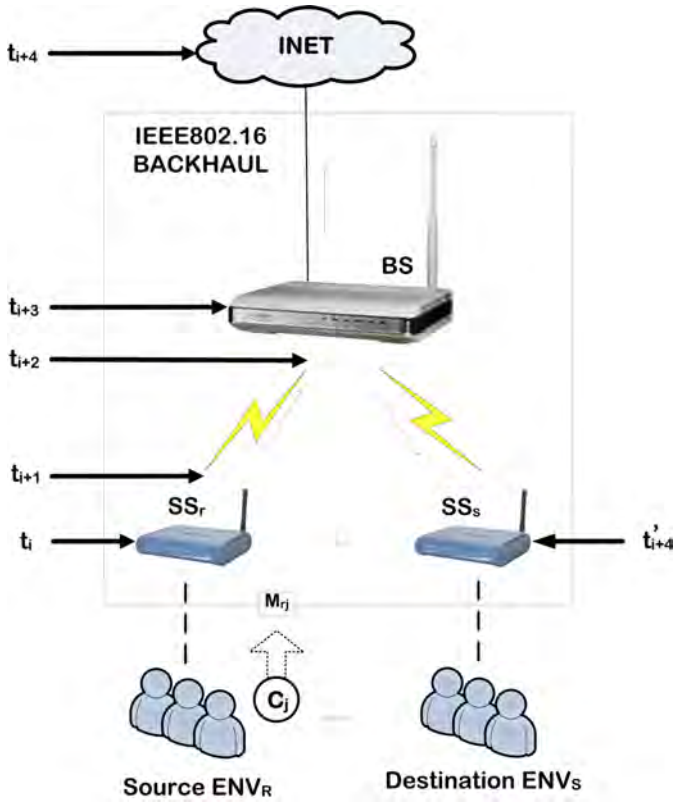

Figure 4.10: Example detailing the delays experienced by network traffic

- M_{rj} starts arriving at the BS at time t_{i+2} with a service time delay of τ_{rj}^{UL}, which depends on the length of M_{rj} and the UL service rate profile for SS_r (i.e. mean link capacity). Therefore $t_{i+2} = t_{i+1} + \tau_{rj}^{UL}$.
- Let δ be the propagation delay. At time $t_{i+3} = t_{i+2} + \delta_i$, M_{rj} is in the BS memory buffers.
- Two cases arise:
  - If M_{rj} is destined for the INET, it is assumed to experience no delay other than a mean delay Δ in the internet. M_{rj} therefore reaches its destination at $t_{i+4} = t_i + \sigma_{rj}^x + \tau_{rj}^{UL} + \delta + \Delta$
  - If M_{rj} is destined for another SS, say SS_s, $s = 1, \ldots, r-1, r+1, \ldots, N$, In this case M_{rj} experiences
    - mean BS queueing delay ψ_{rj}^x,
    - mean DL transmission delay τ_{rj}^{DL} , and
    - the same fixed propagation delay δ

    so that in this case, M_{rj} reaches its destination at time $t_{i+4}^{'} = t_i + \sigma_{rj}^x + \tau_{rj}^{UL} + 2\delta + \psi_{rj}^x + \tau_{rj}^{DL}$

The performance metrics identified and discussed so far all describe packet-level performance. In addition to this, since we are also studying connection admission control, we are concerned with blocking probability, a connection-level performance descriptor. This is the probability that a connection request is rejected, i.e. denied admission on request.

CHAPTER 5

SYSTEM COMPONENTS

The development of the RaCM and workload components are reported in this chapter; each are described in terms of design[1], implementation and testing. Since our simulator, whose development is thoroughly reported in Appendix A, was designed from the start to be modular, these components could be developed in isolation from each other. However, certain design considerations were made when selecting RaCM component features:

Our hypothesis is that there is a synergistic relationship between the system scheduler and CAC components. On proving our hypothesis, the premise that one should utilise the synergistic relationship between the scheduler and CAC design when designing the RaCM, naturally follows. Therefore, in our experiments we were not concerned with determining the optimum - or optimal - RaCM design.

However, we gave much thought to RaCM design: For the scheduler and CAC we used a state of the art RaCM design. Moreover, in order to combine various borrowed features, innovation was necessary. Lastly, out of necessity it was ensured that the scheduler and CAC components are compatible to interact; the necessary traffic categories are supported, the appropriate QoS metrics are considered by both components and both CAC and scheduler must understand which real-time performance estimations, as detailed in Section 5.1.3, are made.

5.1 DESIGN

5.1.1 SS UL and BS DL Schedulers

The same scheduling process is executed by the BS DL and SS UL schedulers[2]. Since this study is not primarily concerned with partnering scheduling processes to optimise some aspect of system performance, it is necessary to ensure that the processes selected are at least compatible with one another. By using the same scheduling process for both the BS DL and SS UL schedulers, we ensure this compatibility.

MAC memory buffers are logically organised into four TC waiting lines: UGS, rtPS, nrtPS and BE. The QoS waiting lines, i.e. comprising of the first three TCs, are served in turn by a work-conserving WRR scheduler.

Service quanta are assigned to each QoS waiting line. The quantum, called a *data quantum* in what follows,

[1] Design features selected for these components were taken from the state of the art design, presented in Chapter 2.
[2] It is not required that these processes are the same but, for the purpose of this study, this is the case.

represents the amount of data to serve from the associated QoS waiting line while it has backlogged data. Since PDUs vary in length across and within TCs, an approximation to the WRR algorithm is implemented namely, each quantum represents the amount of data that must *at least* be served before the next waiting line may start receiving service.

The BE waiting line is only served while all QoS waiting lines are empty since BE traffic makes no demand on QoS. Wireless link utilisation is therefore increased when the QoS waiting lines are not backlogged. If a QoS PDU arrives at its waiting line and a BE PDU is already in service, the BE PDU will first be served completely before the QoS PDU will be put in service.

Each of the QoS and BE waiting lines are served in a FCFS fashion, with no distinction made between the connections subscribed to the same service.

Furthermore, to control the jitter to a greater extent, the scheduler visits QoS waiting lines for a certain number of rounds, i.e. the QoS waiting line visit frequency per frame. Weights are chosen to share the BW of each frame amongst the TCs and are used to determine the service quanta. By increasing or decreasing the *round* parameter, i.e. the number of rounds, the quanta are reduced or increased in size, respectively. In this way, jitter control is attempted.

The configuration of the data quanta and round parameter values is fixed and specified by the service provider at network start up. However, it is possible to enhance the homogeneous scheduler to be more opportunistic by implementing a cost function that considers the present network conditions and updates these parameters. Our implementation does however not consider opportunistic extensions. The BS DL and SS UL schedulers may have different parameters configured. In fact, each SS may have its own parameter configuration.

The process flows of the scheduler are illustrated using two process flow charts (PFCs), as shown in Figures 5.1 and 5.2. Figure 5.1 shows the arrival process in isolation, which may be executing in parallel with data being served from the waiting lines, shown in isolation in Figure 5.2. The variables used in these PFCs are defined in Table 5.1. In Figure 5.2, there is a time delay, Δt, between putting a PDU in service and taking it out of service. Also, when $isqp$ equals -1, it implies that the QoS queues are empty.

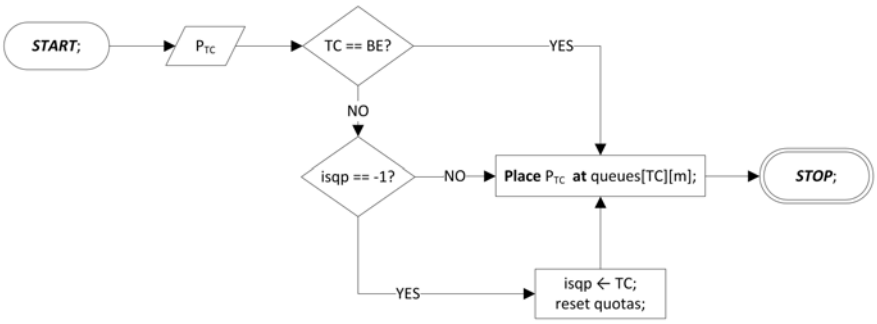

Figure 5.1: BS DL and SS UL PFC for arrivals at the buffers

In the case of the arrival process, shown in Figure 5.1, PDU P_{TC} arrives and is evaluated in terms of its TC. If it is not a BE PDU and if the QoS queues are empty, the queue pointer must point to the TC of P_{TC}, the quotas

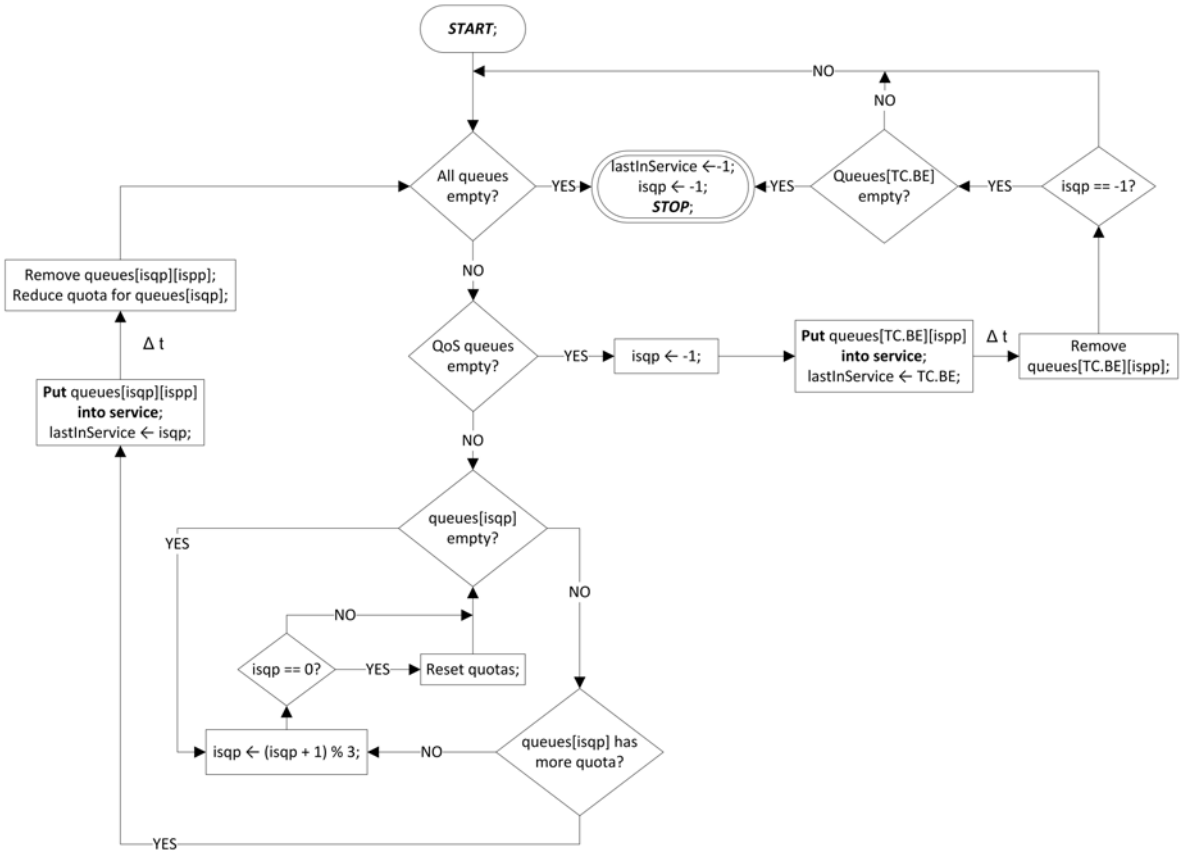

Figure 5.2: BS DL and SS UL PFC for serving PDUs from the buffers

Variable	Definition
$lastInService$	Indicates the waiting line with the current packet in service.
$isqp$	In-service queue pointer, pointing to the waiting line that must be served next.
$ispp$	In-service packet pointer, pointing to the PDU in waiting line isqp to be served next.
$queues$	TC waiting line array.

Table 5.1: BS DL and SS UL scheduler variable definitions

must be reset and the PDU must be placed at the end of the corresponding TC queue. Otherwise, and also in the case that TC is BE, the PDU must be placed at the end of the corresponding TC queue.

Serving the PDUs from the buffers include starting the service of a PDU and thereafter taking it out of service, as shown in Figure 5.2. To put the next PDU into service, it is first determined if all buffers are empty. If so, the service variables are reset and the process is stopped. If the buffers are not empty, it is determined if the QoS queues are empty:

If empty, $isqp$ is set to -1 to indicate that the QoS queues are empty. The PDU at the front of the BE queue is put into service and the $lastInService$ pointer is set to point to the BE queue. After some service time Δt, the PDU in service is taken out of service. If $isqp$ still equals -1, the process either stops – in the case that the buffers are empty – or restarts.

If the QoS queues are not empty, it is determined if the particular QoS queue (pointed to by $isqp$) is empty. If it is empty, $isqp$ visits the next QoS queue, resetting the quotas each time $isqp$ points to 0. Otherwise, if the current QoS TC has no more quota, $isqp$ visits the next QoS queue, again resetting the quotas each time $isqp$ points to 0. If the current QoS TC has enough quota to transmit the PDU, it is put into service and $lastInService$ is set to point to the current QoS queue. After some time, Δt, the PDU in service is removed from the buffers, the quota is reduced for that particular QoS TC and the process is restarted.

5.1.2 BS UL Scheduler

The BS UL scheduler must allocate portions of the UL subframe to the SSs wanting to transmit data to the BS. The BS keeps record of connections in service and which SSs are associated with these connections. The BS therefore knows which SSs may have data to transmit and must decide which of these SSs may transmit in the next frame. However, to attempt fairness and to prevent starvation of BE traffic at an SS with no QoS connections established, all SSs are to be scheduled along the UL in each frame. As already explained, scheduling too many SSs along the UL in a single frame may results in a significant amount of wasted BW due to the preamble necessary for synchronisation at the start of each SS BW grant. This algorithm may pose a problem for the scalability of the network.

Also, it is possible to schedule any SS more than once in a frame. This too may lead to the wasting of BW due to the SS preambles. The BS UL scheduler does not attempt to control jitter since this would require that each SSs be scheduled more than one UL BW grant during each frame. Note that our assumption is that the INET source has a dedicated fixed line UL connection to the BS and therefore it is not considered by this scheduler.

Since the system schedules connections per SS (i.e., GPSS), the BS UL scheduler generates UL BW grants for each SS to share amongst its connections subscribed to any of the TCs: An SS with 1 UGS connection and 4 rtPS connections admitted at present receives a single BW grant for all five connection. Additionally, each SS is assumed to have BE traffic. This is accounted for in the BW grant. It is the responsibility of the SS to schedule transmission for these connections' PDUs during the BW period granted. Scheduling connections of various TCs in one BW grant period requires that the TCs must be weighted according to the service they require – specified by the various QoS parameters. We assume all connections subscribed to a TC has the same QoS requirements as the other connections subscribed to that TC.

The UL transmission subframe is shared between SSs by determining a weight, called credit, per SS that defines the portion of the UL data subframe that should be allocated to the SS: An SS is awarded a basic credit to account for fairness and BE traffic and a certain number of credit per connection presently in-service, where each TC has a fixed number of credits associated with it for each connection subscribed. For example, if an SS, SS_i, has 2 UGS and 3 nrtPS connections admitted at present and UGS and nrtPS connections are weighted with 3 and 5 credits per connection, respectively, the total credits for SS_i are $(2 \times 3) + (3 \times 5) + 4 = 25$ credits, where $basic\ credit = 4$ credits. Furthermore, if there are only 2 SSs to be scheduled and the second SS has a total of 5 credits, then the weights for each of the SSs, SS_1 and SS_2, are $\frac{25}{(25+5)} = \frac{5}{6}$ and $\frac{5}{(25+5)} = \frac{1}{6}$. The UL subframe is then divided into BW grants, allocating the fraction of the subframe to the associated SS. This algorithm is a WRR approach to UL frame scheduling.

Since it is assumed that all SSs use the same robustness profile, i.e. transmit using the same modulation coding scheme (MCS, specifically QPSK), there is no need to factor the MCS into the credit calculation.

5.1.3 BS Real-time performance estimation

As explained in Chapter 3, to evaluate a DSA request, the CAC requests real-time performance information from the scheduler. In our system, the BS must estimate real-time network performance in terms of current mean delay and current mean jitter for the relevant TCs. These values are an estimation of network performance for a time-frame that represents the *current* network status.

In order to avoid a point of dead-lock where after no more connection requests can be admitted, we must make sure that the estimation reflects the most recent status of network performance. To achieve this sense of recency, we must choose an appropriate observation period and call it the *observation window* (T_{window} s). In other words, should the CAC request the system status from the BS at time t_i, it returns the mean delay and mean jitter values for the interval $[t_i - T_{window}, t_i]$.

The mean delay is calculated by summing up the local BS delay of successfully transmitted packets from the BS to its destination per TC and dividing by the number of connections that had packets transmitted. The mean jitter is calculated by doing the same for the jitter values per connection.

Deciding on the appropriate value for T_{window} is an experiment on its own. If T_{window} is chosen too small, the network performance will appear to change too dynamically from the CAC's point-of-view since there would not be enough performance samples to estimate the mean values from. On the other hand, if T_{window} is chosen too large, the system could run into periods of QoS connection starvation: Assume a QoS threshold is reached (say UGS delay). Furthermore, assume all the UGS connections end and the remaining UGS PDUs fail to decrease the mean delay estimation below the threshold value. As the observation window passes, fewer UGS delay samples will be included in the mean US delay estimation until the estimate falls below the threshold. In the worst case, where $T_{window} = \infty$, if this scenario is realised, no more QoS connections will be admitted for the CAC algorithm detailed in Section 5.1.4, i.e. QoS connection starvation.

5.1.4 Connection Admission Controller

The CAC manages BWRs transmitted from an SS to the BS. Each BWR is associated with an SS and connection IDs (SSID, CID) and is either a request for establishment of a new connection (dynamic service addition, DSA) or a request for connection change of a presently in-service connection (dynamic service change, DSC, or dynamic service deletion, DSD). Since it is assumed that every connection, subscribed to a TC, demands the same QoS from the network as all other connections subscribed to the same TC, DSC requests are not considered here. DSD requests are necessary for closing connections.

BWRs arrive directly at the BS CAC queue and are completely evaluated by the end of the connection contention period since we do not explicitly model the this contention process, as stated before. Even though the system operates in GPSS mode, the system does not issue requests for aggregated connections: Our interpretation of GPSS is to allow and evaluate individual requests per SS at the CAC while, still true to GPSS mode, single resource grants are made per SS by the BS UL scheduler. The SS is therefore still responsible for sharing its aggregated resource grant amongst its connections. Our rational for sending the BWRs per connection (without aggregating these), is that the system under study is not envisioned to have thousands (or even hundreds) of SSs. Evaluating non-aggregated connection requests is likely to result in higher network utilisation in terms of the number of concurrent connections at present – which translates to higher BW utilisation and arguably higher revenue for the network provider.

Figure 5.3 shows the PFC for the CAC. On evaluating a request, the CAC first determines whether it is a DSD or DSA. DSD requests are always admitted since this releases resource for future connections. DSA requests for the BE TC are also always admitted since these make no QoS demand. QoS DSA requests, i.e. associated with UGS, rtPS and nrtPS TCs, are evaluated as follows.

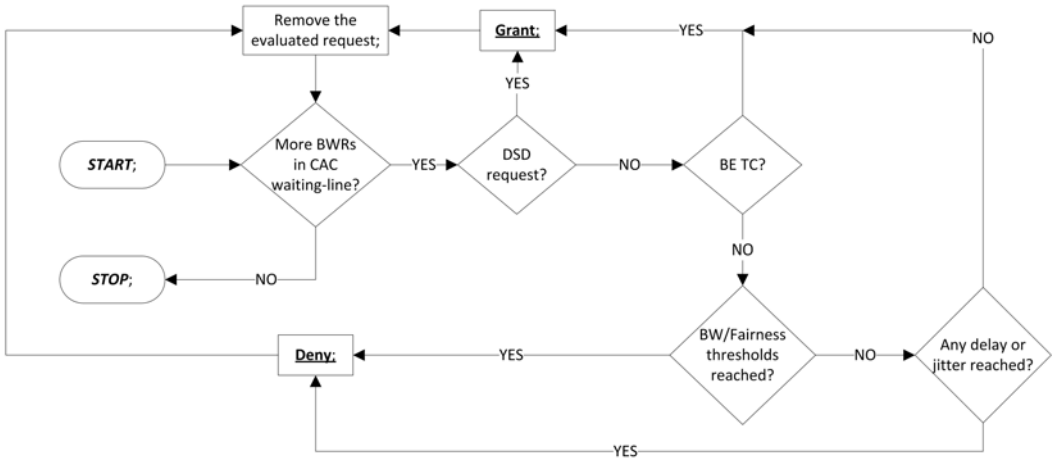

Figure 5.3: Process Flow Chart for the threshold-based QoS-aware CAC

There are two aspects to consider when evaluating a DSA request for a non-BE TC: fairness and QoS. These factors overlap; fairness considers the way in which BW is allocatable to SSs. QoS status considers the minimum mean BW necessary for a connection to operate properly. The notion of fairness may be understood in the strict sense of the word as being fair across all TCs and to every SS. Equally, this makes it possible to discriminate amongst TCs and/or SSs in that the BW sharing mechanism may be used by the network provider to try and solve some other optimisation problem such as revenue maximisation. To ensure fairness across all QoS TCs, thresholds are put in place at the SSs and BS to limit the maximum number of connections that may concurrently be in service per SS and BS. Additionally, there are thresholds for the total number of concurrent connections in service for the SSs and BS regardless of the TCs. To ensure that sufficient BW is available to concurrent connections, the fairness thresholds should consider the minimum mean BW requirements across all QoS TCs. If these parameters are not selected correctly, QoS (in terms of BW) may degrade. In summary, the following threshold parameters must be specified: The maximum number of concurrent connections

1. per SS per QoS TC.
2. per SS, for all QoS TCs combined.
3. at the BS, i.e. over the entire network, per QoS TC.
4. at the BS, i.e. over the entire network, for all QoS TCs combined.

The effect of admission of some TC connection on the QoS performance of other TCs is unknown and therefore all thresholds for delay and jitter must be considered – no matter which QoS TC the DSA request is for. If any of these thresholds have been reached, the request is denied. The BS scheduler is responsible for estimating the

real-time network performance in terms of delay and jitter. A choice of the threshold values involves an estimation of the effect a QoS connection admission may have on delay and jitter, not only on the connections own TC, but also on the QoS on the other TCs.

5.1.5 Workload Generator

As explained in Section 2.3.2, we consider the system workload to constitute of VoIP, video streaming, P2P and HTTP traffic. For all these traffic classes one needs to consider both the generation of connections and the different characteristics of the data passing over each connection. At the packet level we us a Markov Modulated Arrival Process (MMAP) to model traffic generation.

The workload generator generates both connection-level and packet-level traffic: For packet-level traffic generation an MMAP is implemented, where each state of the MMAP is represented by an individual traffic model (TM), one TM per Internet application. Each of the MMAP states, i.e. each individual TM, generates connection-level requests in parallel. To promote generality of the workload model (WLM), it was designed to be

- modular, accommodating easy and effective TM integration, and
- extensible, allowing the integration of multiple traffic generators (TGs).

MMAP Arrival Process

The WLM is an MMAP where each state represents a different TM. Figure 5.4 shows the general scenario of the system actor (simulating machine) requesting workload information from the WLM. One of the traffic generators returns this information, which can either be

- connection or packet IAT,
- connection holding time,
- packet size, or
- connection or packet traffic category information.

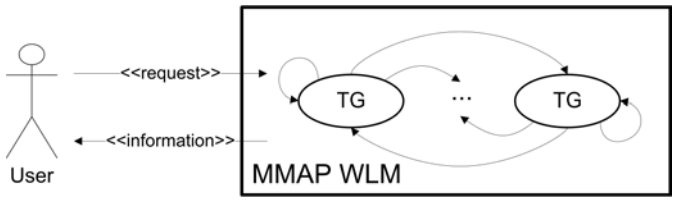

Figure 5.4: Scenario of a system actor using the MMAP WLM

There are well-known techniques, as described by Klemm *et al.* [32] in the MMBP (Bernoulli) case, to determine state vector π and transition probability matrix P from measured traffic data.

Traces of Internet traffic seldom differentiate between traffic types: The application which generated an IP packet is identified by the port number in the packet. In the case of P2P different port numbers are used and

one can therefore not easily distinguish the P2P traffic. There are techniques, such as clustering discussed by Symington [60], for differentiating between traffic profiles, but this is not reliably accurate to distinguish between QoS classes.

For reasons already provided, traffic models (TMs) are implemented per Internet application. Each TM subscribes to a TC in order to associate traffic generated with that TC. However, the WLM is designed such that, if suitable traces were to become available, these can be used in the WLM. State vector π is used, where the value of each state $\pi_i,\ i = 1, 2, 3, 4$, representing web browsing, VoIP, P2P and interactive video streaming, respectively, were derived from Internet traffic measurements. However, P was derived knowing that P is a stochastic matrix and that π is a probability vector. P is derived from solving $\pi P = \pi$.

Connection-level Arrival Process

Each TG generates connection-level requests in parallel. However, an admission (DSA) and a deletion (DSD) request-pair must be generated for each connection. On generation of a DSA, the partnering DSD must be generated. The DSD is scheduled to be sent after a randomly generated duration time.

Each of TG must maintain a *present time* variable and a list of the requests already generated, i.e. requests that are scheduled to be sent at a later time in the simulation. The WLG makes sure that these parallel traffic streams are synchronised by determining the earliest next request to be sent and advancing the *present time* values for all generators.

P2P Traffic Model

The traffic model we used in the P2P case is based on the BitTorrent application.
Distribution models for connection inter-arrival times and durations were identified in Chapter 2.3.2. However, the parameters for these models suggest that sessions typically last for a few hours (in fact). Since the simulation-time resolution is measured in milliseconds, the connection-level characteristics are not modelled. P2P is managed as a BE service in the simulation with no QoS requirements. Hence, a single P2P connection is set up at the start of the simulation and lasts for the entire simulation.

BitTorrent packets are of fixed length and have an IAT equal to the value of propagation delay[3].

VoIP Traffic Model

Taken from He [23], call IAT is exponentially distributed and call duration follows the Pareto distribution.
An ON–OFF model is implemented with exponentially distributed period durations [1, 30]. During the ON period, packet size and IAT are deterministic.

Video Streaming Traffic Model

Real-time video traffic is represented by a video conferencing application model [4].No connection-level model was found for this application. Since it is assumed that user-level video conferencing may behave similarly to real-time voice (VoIP), video conferencing connection IAT is exponentially distributed with Pareto distributed duration.

[3]The propagation delay referred to here is that from the external environment traffic source to the SS.
[4]Other Internet applications that fall into this category are Video on Demand (VoD) or IPTV applications. However, in our study we only consider video conferencing traffic.

Video conferencing is assumed to have a relatively larger mean connection IAT since it is assumed that users use the VoIP application more frequently, while conference sessions are assumed to last longer, on the average, than VoIP calls. Traffic is generated in frame periods, each being a fixed number of packets in length. Between frames, there is an inactivity period of fixed time. During frame activity, truncated Pareto distribution models are used for both packet IAT and size generation.

HTTP Traffic Model

Distribution models for connection inter-arrivals and durations were identified in Chapter 2.3.2. For reasons similar to those given for P2P connection-level workload modelling, a single HTTP connection is set up at the start and lasts for the entire duration of the simulation. Packet-level distribution models are taken from Walters [64]. IAT is Weibull distributed and packet size is lognormally distributed. Since we assume that for every UL HTTP request there is a single DL HTTP object, we account for the significant difference between UL and DL packet sizes by sampling from two lognormal distribution models (each parametrised differently) and sample from each with equal probability.

5.2 IMPLEMENTATION

5.2.1 Schedulers

The SS and BS schedulers were implemented according to the interface specifications provided in Appendices A.2.3 and A.2.3, respectively.

5.2.2 Connection Admission Controller

The CAC was implemented according to the interface specification provided in Appendix A.2.3.

5.2.3 Workload Generator

The main entry point to the workload generator is the *WLG* class. The *WLG* class was implemented according to the workload interface specification provided in Appendix A.2.3. It implements the MMAP structure, described in Section 5.1.5. A generic *TM* class is created which is extended to implement the various TC traffic generators. 4 applications traffic generation processes are modelled:

- BitTorrent P2P traffic: implemented in the *BitTorrent* class
- Web browsing traffic: implemented in the *HTML* class
- VoIP traffic: implemented in the *VoIP* class
- Real-time video streaming traffic: implemented in the *VideoStreaming* class

Each of these classes inherit and redefine the methods of the *TM* class, as shown in Figure 5.5.

5.3 TESTING

As shown in Appendix A.3, the simulator was validated through the derivation of the various abstract models, carefully developed for the system under study. Our testing verifies that the simulator was built correctly,

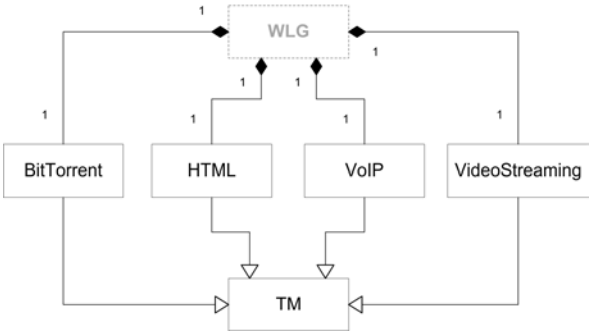

Figure 5.5: UML class diagram of the workload generator-related classes

acknowledging that the simulator state space is simply too large to exhaustively test every function.

Due to the component nature of the simulator, a modular testing approach was taken: CAC, schedulers and workload generator components were tested first, reported in this section. Thereafter, the simulation engine was tested, as reported in Appendix A.3.1. Lastly, in Appendix A.3.2, the integration of these components was tested. Positive testing was done to ensure the simulator components affect the system state variables in the way they should be affected. Negative tests, whereby it is tested whether the simulator "...does not do what it is not supposed to do" [66], were not conducted.

Structural *white-box* tests were conducted to ensure that the simulator event process flows were correctly translated from PFCs into code: Integration testing, reported in Appendix A.3.2, was done by inspecting the program source, i.e. tracing the method-chains invoked to the point where the required system state variables are affected.

Behavioural *black-box* tests were also done to ensure various methods operate correctly: The workload generator was tested in terms of the workload units (PDUs or BWRs) produced on invocation of the defined methods. The SSJ library [35] was used for random number generation and is a well-known and tested statistical library. Therefore, the random number generation source was not tested.

Where possible, test case artefacts were reported as evidence of testing. Test cases are either simple or compound: the simple test case tests for a single feature or function, whereas the compound test case test for various features in one test case scenario. Each test case consists of five fields, as shown in the Table 5.2. In Appendix B, examples of test cases are provided. The other test cases, not provided in this appendix, are available on request. This section only describes program source tests conducted.

Field	**Content**
ID	*unique test case identifier*
Description	*description of the test*
Preconditions	*system state before the test*
Expected results	*expected system state after the test*
Outcome	*PASS/FAIL status of the test*

Table 5.2: Test case template

5.3.1 SS UL and BS DL Schedulers

Since these two schedulers implement the same algorithm, test cases needed to be constructed for only one. The SS UL scheduler was used for testing. 15 test cases were contructed to test the various branches of the SS UL scheduler PFC, illustrated in Section 5.1.1:

Test case S1.1: A BE PDU arrival.

Test case S1.2: Arrival of a QoS PDU where at least one QoS queue is backlogged.

Test case S1.3: Arrival of a QoS PDU where all QoS queues are empty.

Test case S1.4: Next PDU is put into service where all queues are empty.

Test case S1.5: Next PDU is put into service where only the BE queue is backlogged.

Test case S1.6: Take the currently in-service BE PDU out of service. All queues are empty afterwards.

Test case S1.7: Take the currently in-service BE PDU out of service. All QoS queues are empty but the BE queue is still backlogged afterwards.

Test case S1.8: Take the currently in-service BE PDU out of service. QoS queues are not all empty afterwards.

Test case S1.9: Serve the next QoS PDU while all QoS queues are backlogged, except for the in-service queue that is empty. isqp has not reached the initial queue and thus the next queue status must be checked.

Test case S1.10: There is at least one QoS queue backlogged. isqp has reached the initial queue and thus the quotas must be reset before the next queue status must be checked.

Test case S1.11: The in-service QoS queue is backlogged and there is at least another QoS queue backlogged. However, the in-service queue has exceeded its quota. The next PDU is put into service.

Test case S1.12: The next QoS PDU at queue, pointed to by the isqp, is put into service.

Test case S1.13: Take the in-service QoS PDU out of service. Quota must be reduced for the associated TC.

Test case S1.14: Re-transmission of a PDU already in service.

Test case S1.15: Attempt to take a PDU out of service when the buffers are empty and NO PDU is in service.

5.3.2 BS UL Scheduler

6 test cases were drawn up for the BS UL scheduler component:

Test case S2.1: Calculate the size of the MAP in bits.

Test case S2.2: Calculate and generate the DLMAP and thereafter the ULMAP.

Test case S2.3: Update the delayed pool with an empty BWR vector. To illustrate the update, the initial delayed pool should not be empty.

Test case S2.4: Update the active pool with a non-empty delayed pool only containing DSA requests. Ensure that all SSs and the INET each have at least 1 connection admitted.

Test case S2.5: Calculate the new MAPs after test case S2.4 has been done.

Test case S2.6: Test deletion of connections, i.e. DSD request. Delete a request at an SS and recalculate MAPs.

5.3.3 Connection Admission Controller

4 test cases were drawn up for testing the CAC component:

Test case C1: Tests the CAC at the time of evaluation when it has no BWRs waiting to be evaluated.

Test case C2: Tests whether the CAC can evaluate multiple different requests. Furthermore, it tests the correct evaluation of 1 BE DSA and 2 QoS DSA. The first DSA comes from the INET source and the second comes from an SS source and both must be admitted.

Test case C3: Tests whether the CAC can evaluate a DSD request correctly.

Test case C4: Tests 3 DSAs separately. It shows how the CAC will deny a BWR if it violates any of the QoS threshold values of fairness, delay or jitter.

5.3.4 Workload Generator

The workload generator was tested to make sure that both packet-level and connection-level data are generated correctly. The underlying TMs for HTML, BitTorrent, VoIP and Video Streaming traffic were tested individually for PDU generation. The MMAP workload generator was tested to ensure that it generates traffic (1) in the correct volume and (2) for all TCs catered for.

We tested that the HTML and BitTorrent TMs generate only one DSA request each at the beginning of connection request and then disable the possibility of generating any further requests. Lastly, it was tested that the VoIP and Video Streaming TMs generated BWRs for connections with different CIDs and that each CID was uniquely associated with one DSA and one DSD request.

Test case W1: All TMs generate PDUs correctly.

Test case W2: MMAP WLG generates PDUs in the correct volume.

Test case W3: MMAP WLG generates PDUs for all TCs. However, if test case W2 passes and all TCs present in the system were tested for, then this test case passes too.

Test case W4: HTML and BitTorrent generates only one DSA request each at the beginning of request generation and disable the possibility of generating any further requests.

Test case W5: VoIP and Video Streaming TMs generate BWRs for connection with unique CIDs, one DSA and DSD per CID.

Test case W6: Testing INETWLG traffic volume generation for varying γ and S per TC.

CHAPTER 6

STEADY-STATE ANALYSIS

Before executing any simulation, one needs to decide the values of the many parameters in the simulation. Some of these are given in the standard, such as frame duration, MAP size and so on. Others are a system choice, such as the number of SSs to simulate. Values for certain free parameters, such as the mean arrival rate of the various TCs cannot be chosen, arbitrarily. The system has to be in steady state and utilisation of the radio link cannot exceed unity. In this Chapter, we set out to find a relationship between certain parameters, and choosing values for them while ensuring the system will remain in steady state.

Figure 6.1, derived from Figure 4.9, shows annotations appropriate for the steady-state analysis.

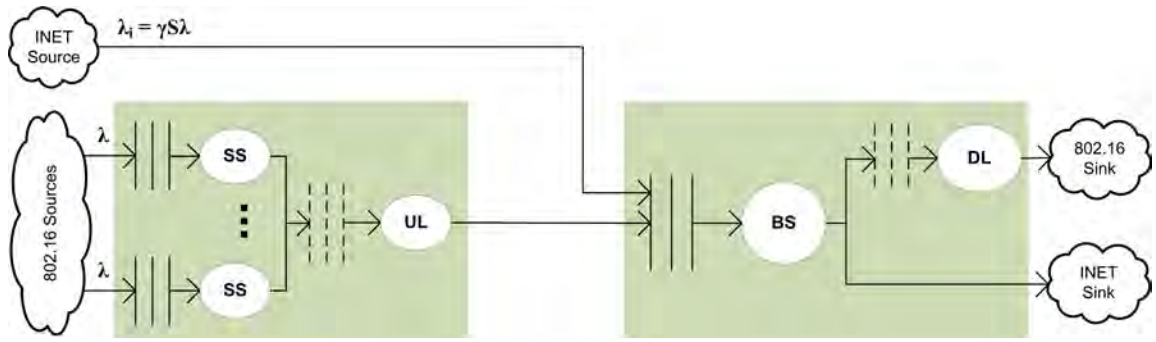

Figure 6.1: Network of queues annotated for stability analysis

Packets arrive at each SS queue at a mean arrival rate λ. The SS server (SS in the figure) performs certain unspecified functions, including scheduling packets for transmission on the UL. Scheduled data are queued at the virtual UL queue and reside there until they are to be transmitted on the UL according to the ULMAP.

On arrival at the BS along the UL and the fixed-line INET interface, packets are queued in the BS buffers. The packets destined for the Internet are scheduled for transmission along a high-speed fixed line modem which we assume takes negligible time. Otherwise, the BS server (BS in the figure) performs certain unspecified functions, including scheduling packets for transmission on the DL. Scheduled data are queued at the virtual DL queue and reside therein until they are to be transmitted on the DL according to the DLMAP.

In both the SS and BS, the SS and BS servers are ignored in the analysis.

Next, we define the following parameters:

1. λ: mean packet-level arrival rate at an SS in bits/s
2. λ_i: mean packet-level arrival rate at the BS from the Internet in bits/s
3. MCS_{UL}: fixed data modulation rate along the UL in bits/s
4. MCS_{DL}: fixed data modulation rate along the DL in bits/s
5. S: number of SSs in the network
6. γ: mean fraction of SS traffic arrivals destined for the Internet
7. ω_{UL}: fraction of the frame dedicated to UL data transmission
8. ω_{DL}: fraction of the frame dedicated to DL data transmission
9. ρ_{UL}: UL subframe link utilisation
10. ρ_{DL}: DL subframe link utilisation

From Figure 6.1 we can derive the formulae for the UL and DL utilisations, ρ_{UL} and ρ_{DL}, respectively:

In the case of the UL, the arrival rate at the UL channel server is the sum of the arrival rates over all the SSs, i.e. $S\lambda$. The UL channel service rate is derived as $\mu_{UL} = \omega_{UL} MCS_{UL}$ since data is modulated and transmitted along the UL at a rate of MCS_{UL} for ω_{UL}. ρ_{UL} is given by Equation 6.1.

$$\rho_{UL} = \frac{S\lambda}{\omega_{UL} MCS_{UL}} \tag{6.1}$$

ρ_{DL} is given by Equation 6.2: The arrival rate at the DL channel server is the arrival rate at the BS buffers of data destined for the wireless domain. All packets arriving from the Internet are destined for the wireless domain, at a mean arrival rate λ_i. On average, $(1-\gamma)$ of the traffic arriving at the BS along the UL is destined for the wireless domain, at a mean arrival rate $(1-\gamma)\mu_{UL}$, where the arrival rate at the BS along the UL is the service rate of the UL channel. As for μ_{UL}, $\mu_{DL} = \omega_{DL} MCS_{DL}$.

$$\rho_{DL} = \frac{\lambda_i + (1-\gamma)\omega_{UL} MCS_{UL}}{\omega_{DL} MCS_{DL}} \tag{6.2}$$

The over-all link utilisation is defined as $\rho_{link} = \rho_{UL} + \rho_{DL}$.

The analysis assists the network operator with parametrising the network such that it is in a steady-state during network operation. The conditions for steady-state are

1. $\omega_{UL} + \omega_{DL} = 1$
2. $\rho_{UL} \leq 1$

3. $\rho_{DL} \leq 1$

As a consequence of conditions 2 and 3, $\rho_{link} \leq 2$.

One needs to, or should choose parameter values which ensure maximum ratio link utilisation. Ideally, we want to maximise both UL and DL utilisations, while maximising the utilisation of the link ρ_{link} as well.

We select system parameters that result in the maximisation of both UL and DL utilisation: Firstly, the number of SSs and the PHY layer implemented must be selected by the network operator, which depends on physical device, deployment and operating costs. These parameter values are regarded as being given.

The remaining parameter values to choose are therefore ω_{UL}, λ and γ, noting that λ_i can be expressed in terms of S, λ and γ. Selecting a value for the mean arrival rate λ may at first seem to contradict the notion that the workload is external and not determined by the network. However, the concept of QoS means that the network operator indeed needs to exercise connection admission control to maintain adequate service levels; the network operator limits network traffic by implementing RaCM protocols.

From condition 1, we can substitute ω_{DL} by $1 - \omega_{UL}$, in Equation 6.2. Since we wish to optimise both UL and DL utilisations, we set $\rho_{UL} = \rho_{DL}$. Hence, we obtain the second order polynomial given by Equation 6.3 with coefficients a, b and c given by Equations 6.4 through 6.6, respectively.

$$a\omega_{UL}^2 + b\omega_{UL} + c = 0 \tag{6.3}$$

$$a = (1-\gamma)MCS_{UL}^2 \tag{6.4}$$

$$b = S\lambda(\gamma MCS_{UL} + MCS_{DL}) \tag{6.5}$$

$$c = -S\lambda MCS_{DL} \tag{6.6}$$

For valid parameter values,

- $a \geq 0$,
- $b \geq 0$ and
- $c \leq 0$.

Solving for ω_{UL} is trivial. The real root for ω_{UL} is given in Equation 6.7, where $0 \geq \gamma < 1$.

$$\omega_{UL} = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \tag{6.7}$$

Finally, we want to determine λ and γ for $\rho_{UL} = \rho_{DL} = 1$, which is the maximisation of both UL and DL utilisations for the ω_{UL} given in Equation 6.7.

For a chosen γ, by substituting Equation 6.7 into Equation 6.1, we derive the second order polynomial given by

Equation 6.8, with coefficients d and e given by Equations 6.9 and 6.10, respectively.

$$d\lambda^2 + e\lambda = 0 \tag{6.8}$$

$$d = S((1-\gamma)MCS_{UL} + \rho_{UL}(\gamma MCS_{UL} + MCS_{DL})) \tag{6.9}$$

$$e = -\rho_{UL}^2 MCS_{DL} MCS_{UL} \tag{6.10}$$

Since, for valid parameter values,

- $d \geq 0$, and
- $e \leq 0$.

we obtain the positive real non-zero root of λ, given by Equation 6.11, by solving Equation 6.8, where $d \neq 0$.

$$\lambda = -\frac{e}{d} \tag{6.11}$$

As an example, consider a network of 6 nodes, $MCS_{DL} = 40$ Mbps and $MCS_{UL} = 80$ Mbps. Furthermore, assume that 50% of the traffic generated by the 802.16 sources is destined for the Internet, i.e. $\gamma = 0.5$. By choosing $\rho_{UL} = 1$, Equation 6.11 gives $\lambda = 4.\dot{4}$ Mbps. As shown in Figure 6.2, $\rho_{UL} = \rho_{DL} = 1$ for $\lambda = 4.\dot{4}$ Mbps. With $\lambda = 4.\dot{4}$, Equation 6.7 gives $\omega_{UL} \approx 0.332$, verified by the figure as well.

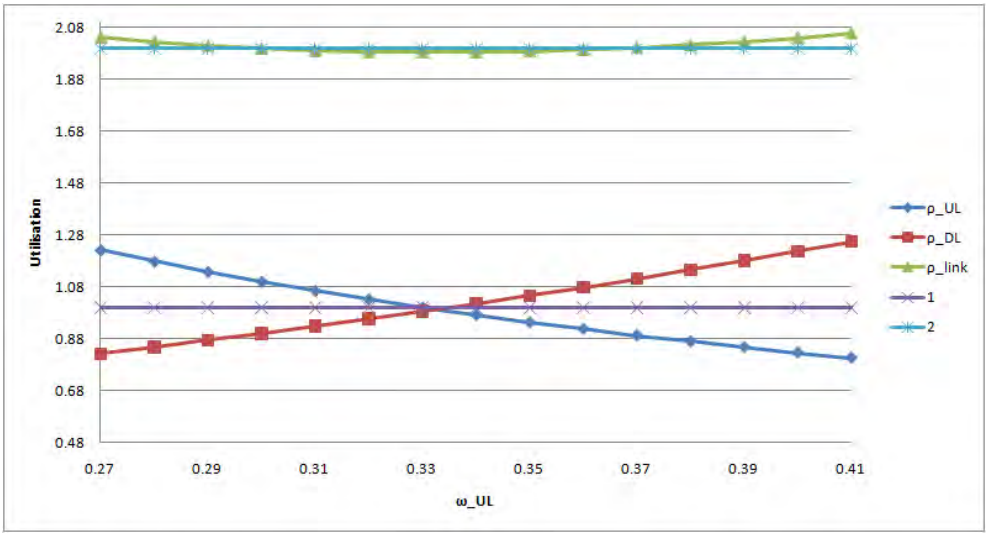

Figure 6.2: Link utilisations for a changing UL data frame ratio, with $\lambda = 4.4$ Mbps and $S = 6$

Keeping in mind that the analysis neither includes the MAP transmission time nor the UL connection contention period, the utilisation predictions are over-estimates of the true link utilisations. Moreover, the analysis assumes that the entire UL and DL sub-frames are dedicated to data transmission, i.e. all scheduling over-head such as synchronisation preambles and idle periods, *inter alia*, are not considered. In itself, the latter assumption also implies over-estimated link utilisation predictions. Therefore, since the conditions for steady-state operation are upheld, the parameter relationships given in their respective equations maintain steady-state operation of the network.

In conclusion, it should be the objective of the RaCM to maintain a traffic level of λ over the network. Ideally, this traffic should mainly be made up of QoS traffic that results in maximisation of revenue according to the specific revenue model the operator employs.

CHAPTER 7

EXPERIMENTATION

7.1 HYPOTHESIS

We distinguish between two levels of system activity, namely at the connection-level and packet-level. The connection-level includes the management of connection-related data units, while the packet-level includes the management of PDU packets within the network.

As shown in Figure 7.1, the CAC operates at the connection-level of system activity while the scheduler operates at the packet-level. This means that the connection-level performance, represented by A, is managed by the CAC, while the packet-level performance, represented by B, is managed by the scheduler.

We hypothesise that there is synergism between these components: In other words, we expect that performance measured at the connection-level is not only directly controlled by the CAC but also indirectly by the scheduler. Moreover, performance measured at the packet-level is expected not only to be directly controlled by the scheduler but also indirectly by the CAC. The impact that these components have on each other's performance-levels is the synergistic relationship between these two components within the RaCM.

In our study, we nominated connection blocking probability as our connection-level performance measure, i.e. metric A, since it is the designated responsibility of the CAC solely to grant or deny connection requests. Furthermore, as packet-level performance indices, i.e. at B, we nominated the throughput, delay and jitter performance metrics, as experienced by the user per connection per traffic category, since the scheduler is solely responsible for managing PDU transmission.

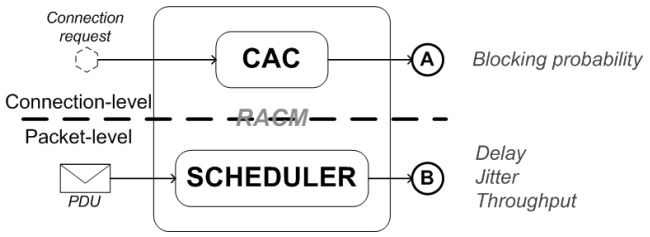

Figure 7.1: Conceptual system activity levels and components involved

To prove our hypothesis, we experimented with different CAC and scheduler configurations: Assume that we have two different CACs, CAC1 and CAC2, and two different schedulers, SCHED1 and SCHED2. All other system parameters remained the same throughout all experiments. We must therefore show that, by experimenting with

- CAC1-SCHED1 and CAC1-SCHED2 RaCM configurations, i.e. fixing the CAC, we *expect* to observe a significant difference, not only in scheduler performance metrics but more specifically, in the connection blocking probability for either one or more of the traffic categories.
- CAC1-SCHED1 and CAC2-SCHED1 RaCM configurations , i.e. fixing the scheduler, we *expect* to observe a significant difference, not only in connection-level performance metrics but more specifically, in the throughput, delay or jitter performance for either one or more of the traffic categories.

7.2 EXPERIMENTAL DESIGN

7.2.1 Independent Variables

The wireless Internet scenario, introduced in Chapter 1, is a fixed WMAN. This means that SSs are added to or removed from the network on occasion, but rarely. Therefore the number of SSs in the network was fixed in all of the experiments.

As explained in Section 7.1, we observed the system performance for different RaCM configurations. For each experiment the arrival rate λ at each SS (in Mbps) was the independent variable.

7.2.2 Methodology

Altogether, 4 experiments were conducted. For each experiment, we changed the RaCM configuration. The 4 experiments were

1. CAC1-SCHED1,
2. CAC1-SCHED2,
3. CAC2-SCHED1, and
4. CAC2-SCHED2.

Each experiment reported both connection– and packet-level QoS performance for the system operating under different workload intensities (as explained in Section 7.2.1). In particular, as a connection-level performance indicator, we considered blocking probability for UGS and rtPS separately. As packet-level indicators, we considered delay, i.e. response time, and jitter experienced over the WMAN network, also for UGS and rtPS separately. Specifically, we considered the delay and jitter per connection per TC, having defined jitter as the absolute difference in the delay of two consecutive PDUs associated with a particular connection. Finally, throughput was considered for UGS, rtPS and BE separately, as well as for the combined (overall) case, i.e. the cumulative throughput across all TCs; throughput was computed per TC only.

There are several advanced sampling techniques, such as the *repeated runs* and *batch means* methods [34], whereby point-estimate predictions are possible using only a few (about 30 [33, 34]) independent and identically distributed (IID) samples. By using sufficient IID samples, one can statistically predict the accuracy of the point estimate under consideration with a certain level of confidence [6, 33, 34].

However, we had to take into account the practicalities of the system: Due to the system's complexity, it takes a long time (both simulated and real time) for an accurate QoS estimation. Therefore one must consider the *time-relevance* of the performance estimator: When compared to the average connection holding time, it does not make sense to report a mean value for a network operation period many times longer. Also, when the variance in the system performance is high over such a practically long time and varies drastically over different time intervals, as we show in Section 7.3, mean values over the entire duration of the network life-time as a performance indicator does not make much sense. A running mean (or moving average) is a better estimator of QoS since it relates more accurately to the quality of experience (QoE).

Furthermore, it is not the objective of this dissertation to prove any one configuration superior. We are only concerned with obtaining *comparable* data to show that inter-RaCM synergy exists. At the start of a simulation run, the simulator does not immediately exhibit typical behaviour of the system, primarily due to starting conditions [6, 33, 34]. We inspected our performance results for the simulation runs and identified an initial transient time after which we could start considering data samples to be representative. The initial samples, assumed to be biased, were not considered to compute performance statistic [6]. By plotting the mean number of connections active in the system over time, the initial transient was identified, with an example shown in Appendix C.1.

Additionally, the system connection behaviour was investigated to identify activity-regions which were likely to be a result of some rare events. As the example in Appendix C.2 shows, the mean number of connections in the system were inspected over time to identify regions that may be rare event occurrences. By ignoring rare event data, we can compare performance statistics without having to consider the influence of the rare event occurrences.

We identified a section of time after the initial transient that does not contain any of these rare event occurrences, making sure that this section was the same for each of the different experiments. We consider the data contained in the section as *consequential* data, i.e. they result from the system operating a particular RaCM configuration: All experimental variables of the baseline model, described in Section 7.2.3, were initialised to the same state at the beginning of each experimental run for each experiment. Furthermore, the experiments generated a vast amount of performance data. Even though this means that the samples are correlated for each individual run, the particular correlation effects are consequential for the samples of each experiment.

Since we have consequential data, it was not necessary to employ advanced statistical techniques, such as calculating confidence intervals. Due to the correlation between the consequential data samples, confidence intervals would firstly be optimistic. Furthermore, given the particularly large number of samples obtained, the confidence intervals would tend to zero.

The mean value of each performance estimate suffices to contrast the experiments' results. In addition to the mean value, the standard deviation may also be used, as shown briefly in Section 7.3.

Finally, since SSs may go into a state of deadlock due to the BS UL scheduling process, as is explained in Section 7.2.3, and buffer management are out of the scope of this study, some PDUs are in the system for an abnormally long time. Practically, if a connection is established but the user does not receive service for his data,

the user would typically terminate the connection after some time, even though the data are still waiting to be served by the network. These performance data were considered irrelevant since the user did not perceive the performance at the actual time at which these data were transmitted.

7.2.3 Model Parameterisation

Baseline Model

The baseline model is the system model definition excluding the RaCM components and simulation execution parameters. It includes the PHY layer, workload model, frame aspect ratio and fragmentation parameters. The baseline model was devised in order to ensure that the experimental results are that of a typical or at least plausible system scenario.

The PHY layer parameter values are listed in Table 7.1. These values are selected from the standard specifications of the system PHY layer, as outlined in Section 2.1.3. To select UL and DL data frame ratios (ω_{UL} and ω_{DL}, respectively), a steady-state analysis of the system was described in Chapter 6. The frame aspect ratio and fragmentation parameter values are also listed in Table 7.1.

Parameter	**Value**
Frame duration (T_{frame})	$1ms$
UL wireless modulation rate (MCS_{UL})	80 Mbps
DL wireless modulation rate (MCS_{DL})	40 Mbps
Connection contention duration (T_{conn})	$50\mu s$
DL:UL frame ratio	668:332
Maximum PDU fragment size (C_{frag})	1280 bits

Table 7.1: Physical Layer parameter values selected for the baseline model

The maximum size of PDU fragments may effect the system performance: Since a PDU (or fragment thereof) can only be sent in its entirety or not at all, if the fragment size is too large, UL BW grants to an SS will be unused. SSs will receive transmission opportunities but may not be able to use them if the grant is too small. However, since this study does not concern itself with this particular problem, we need only to prevent the system from entering a state of deadlock, i.e. permanent congestion: The BS allocates UL transmission time to each SS. However, an SS can only transmit a PDU if there is sufficient time to transmit the entire PDU. If the SS is allocated time less than that of the fragment size it will not be able to transmit that PDU/fragment. If the BS is not able to allocate enough BW during system operation to an SS, that SS becomes deadlocked.

We also know that all SSs are scheduled along the UL during each frame. Furthermore, in our system, all SSs are treated equally in terms of workload and scheduling process parameter values. Therefore we must determine the PDU fragment size C_{frag} (in bits), such that $0 < C_{frag} \leq C_{UL}^{S}$, where C_{UL}^{S} is the mean amount of data that each SS may be able to send per UL frame for some number S of SSs.

Considering the MAP transmission time T_{MAP} and connection-contention period T_{conn}, T_{UL} is given by Equation 7.1. Furthermore, we chose $T_{conn} = 50\mu s$, $\omega_{UL} = 0.332$ and $S = 6$, and T_{MAP} is given by Equation 7.2.

$$T_{UL} = \omega_{UL} \times (T_{frame} - T_{MAP}) - T_{conn} \tag{7.1}$$

$$T_{MAP} = \frac{140 + 32 \times S}{MCS_{DL}} \tag{7.2}$$

With $C_{frag} = C_{UL}^{S}$, where $C_{UL}^{S} = \frac{T_{UL} \times MCS_{UL}}{S}$, we must select $C_{frag} \leq 3723$ bits. As mentioned before, this study is not concerned with the particular effect of PDU fragment size on QoS performance of the system. We selected $C_{frag} = 1280$ bits: Each VoIP PDU is 1280 bits long. To prevent VoIP fragments, we chose 1280 bits as PDU fragment size. Furthermore, since we expect that HTML response objects will be fragmented and know PDUs can not be transmitted in part in our system, a smaller fragmentation size is desired.

With $C_{frag} = 1280$ bits, temporary congestion is still possible. Since the UL scheduler allocates UL transmission periods based on the current connection status of the system, it may be possible that an SS may experience a period during which it can not transmit any data. However, since the connection-level behaviour is dynamic and SSs are considered the same, the SS will recover from this congestion state at some point.

Table 7.2 shows the workload model parameters and their selected parameter values. We relied on existing studies for each of the traffic classes we use in our system simulation, surveyed in Section 2.3.2.

The packet-level parameter values were selected considering the various studies also previously introduced in Section 2.3.2. However, in order to manipulate the average arrival rates (over all TCs), we adjusted the IAT parameter values of each traffic application such that all application generate traffic at a mean rate of 2 Mbps. 2 Mbps was a convenient target rate to manipulate the various traffic models to generate data at and manipulation of this rate is achieved by the command-line workload scaling parameter, described in Appendix A.2.4.

At the connection-level, the parameter values were selected based on the survey presented in Chapter 2.3.2. The IATs were reduced, ensuring that the transitions rates between the different TM applications were maintained. The same was done for the mean connection duration.

The percentage of traffic $\overline{P}$ generated by each application type, was estimated using the Internet traffic study [56] introduced in Section 2.3.2. In order to ensure that the volumes of traffic remain proportional, as given by $\overline{P}$, we need to adjust the transition probabilities in the workload generator with respect to the mean PDU lengths $\bar{l}$.

Let $\overline{L}$ be the mean PDU length generated by the MMAP generator regardless of traffic type and define the transition probability matrix (TPM) as $\overline{V}$. $\overline{L}$ can then be expressed using $\bar{l}$ and $\overline{V}$, given by Equation 7.3.

$$\overline{L} = \sum_{app=1}^{4} (l_{app} \times V_{app}),\ app \in \{1 = VoIP,\ 2 = Video,\ 3 = P2P,\ 4 = HTML\} \tag{7.3}$$

We also have the following equalities:

1. $V_1 + V_2 + V_3 + V_4 = 1$
2. $P_1 + P_2 + P_3 + P_4 = 1$

By multiplying equality 2 by $\overline{L}$ and comparing the resulting equation to Equation 7.3, it follows that $V_{app} \times l_{app} = P_{app} \times \overline{L}$, where $app \in \{1, 2, 3, 4\}$; rearranging, we obtain Equation 7.4. Substituting Equation 7.4 into

Parameter	Connection-level		Packet-level	
HTML	Connection IAR Connection duration	Not applicable Not applicable	Packet IAT (ms) Packet UL size (byte) Packet DL size (byte)	Weibull: • $\lambda_{scale} = 315778.506$ • $\alpha_{shape} = 0.10494$ • $\delta_{location} = 0$ Lognormal: • $\mu_{scale} = 5.883$ • $\sigma_{shape} = 0.331$ Lognormal: • $\mu_{scale} = 7.401$ • $\sigma_{shape} = 1.405$
BitTorrent	Connection IAR Connection duration	Not applicable Not applicable	Packet IAT (μs) Packet size (byte)	Deterministic: • 512 Deterministic: • 128
VoIP	Connection IAR ($connections/s$) Connection duration ($minutes$)	Exponential: • $\lambda_{scale} = 0.2$ Pareto: • $\alpha_{shape} = 1.21$ • $\beta location = 0.2$	'ON' duration (s) 'OFF' duration (s) Packet IAT (ms) Packet size (bytes)	Exponential: • $\lambda_{scale} = 1.004$ Exponential: • $\lambda scale = 0.587$ Deterministic: • 0.406 Deterministic: • 160
Video Streaming	Connection IAR ($connections/s$) Connection duration ($minutes$)	Exponential: • $\lambda_{scale} = 0.1$ Pareto: • $\alpha_{shape} = 1.21$ • $\beta location = 0.4$	Packets per frame (packets) Inter-packet time in frame (ms) Packet IAT (ms) Packet size (byte)	Deterministic: • 8 Deterministic: • 0.01 Truncated Pareto: • $\alpha_{shape} = 1.2$ • $\beta location = 0.0614$ • $Maximum = 0.0052$ Truncated Pareto: • $\alpha_{shape} = 1.1$ • $\beta location = 20$ • $Maximum = 125$
TPM, V	HTML= 1%, BitTorrent= 71%, Video= 24% and VoIP= 4%			
Application, P	HTML= 20%, BitTorrent= 65%, Video= 10% and VoIP= 5%			

Table 7.2: Workload model parameter values for the system

equality 1, we solve for $\overline{L}$, given by Equation 7.5, since all the necessary variables are known. Finally, we can solve for $\overline{V}$ by substituting Equation 7.5 into Equation 7.4. The resulting TPM perventages, i.e. $\overline{V}$, is listed in Table 7.2.

$$V_{app} = \frac{P_{app} \times \overline{L}}{l_{app}} \tag{7.4}$$

$$\overline{L} = \frac{1}{\sum_{app=1}^{4} \frac{P_{app}}{l_{app}}} \tag{7.5}$$

As for the packet-level arrival rate at the BS from the INET node, the mean arrival rate scaling parameters were changed by a factor of $S \times \gamma$, where S is the number of SSs in the system and γ is the percentage of SS-generated

traffic destined for the Internet. In our system, we choose $\gamma = 0.5$ but clearly this is arbitrary.

Admission Controllers

We experimented with two admission controllers, described in Section 5.1.4, namely CAC1 and CAC2. In our study, we used the same admission controller algorithm for each CAC. However, we differentiated between the CACs by selecting different algorithm parameters. CAC1 and CAC2 parameters are listed in Table 7.3. The number-of-connection thresholds were chosen by inspecting preliminary simulation runs with an admit-all CAC, as to allow more voice than video over the network. The delay and jitter thresholds were selected to be of milli-second resolution. Also, stricter threshold values were set for voice than for video.

Parameter	$CAC1$ **Value**	$CAC2$ **Value**
UGS delay threshold	0.100	0.050
UGS jitter threshold	0.100	0.050
RTPS delay threshold	0.300	0.150
RTPS jitter threshold	0.150	0.075
Maximum UGS connections per SS	15	15
Maximum RTPS connections per SS	10	10
Maximum UGS connections per BS	75	75
Maximum RTPS connections per BS	50	50
Maximum connections per SS	20	20
Maximum connections per BS	100	100

Table 7.3: CAC parameter values for experimentation

Schedulers

As in the case of the admission controllers, we experimented with two system schedulers, described in Sections 5.1.1 and 5.1.2, namely SCHED1 and SCHED2; each consisting of three distinct schedulers, namely the BS UL–, BS DL– and the SS UL schedulers, described in Section 2.2.1, page 14. The same scheduling algorithms were used for each of the system schedulers, differentiating between the two by selecting different algorithm parameters. SCHED1 and SCHED2 parameters are listed in Table 7.4.

Scheduler parameters were selected such that either voice or video traffic would be favoured, or both would be served the same. Therefore, SCHED2 was parametrised by adjusting various service ratios to favour the classes differently. The real-time performance estimate time-window, described in Section 5.1.3, remained the same.

Execution Parameters

The simulation execution parameter values remained the same for each experiment. Table 7.5 shows the parameter values for all the experiments. As shown in Chapter 6, for the particular w_{UL} selected, the system is fully utilised at a rate of 4.4 Mbps. We therefore selected a range of traffic intensities below 4.4 Mbps for our experiments.

From preliminary runs, it was apparent that the initial transient was over at $t = 1000$s. We decided to run the simulations for 6500s because we wanted to obtain as much performance data as possible given the amount of available RAM and ROM memory. This provided us with enough simulation time from which we could identify a period of simulation time (the same for each experiment run) to compare the experiments.

Parameter	$SCHED1$ **Value**	$SCHED2$ **Value**
BS UL scheduler		
UGS credit per connection	4	5
RTPS credit per connection	3	2
Fairness credit per SS	3	2
Real-time performance estimate time-window	$0.5s$	$0.5s$
BS DL scheduler		
UGS weight	0.2	0.8
RTPS weight	0.8	0.2
NRTPS weight	0	0
Rounds	1	2
SS UL scheduler		
UGS weight	0.5	0.3
RTPS weight	0.5	0.7
NRTPS weight	0	0
Rounds	1	1

Table 7.4: System scheduler parameter values for experimentation

The initial seeding value of the random number generators is arbitrarily chosen and provided for repeatability of the experiments.

Parameter	Value
SS arrival rate (minimum)	1.6 Mbps
SS arrival rate (increment)	0.4 Mbps
SS arrival rate (maximum)	3.6 Mbps
Number of SSs	6
Simulation duration	6500s
Tracing starting time	0s
Random number generator seed	11111

Table 7.5: Experiment execution parameter values

Trace file processing parameters

Once performance data were generated by the simulation experiments, we considered the system to be no longer within the initial transient after $t = 1000$s. This was determined by inspecting the number of connections metric over time for each of the simulation runs. Then, on inspection of the number of connections results, we selected the section from simulation time $t = 4000$s to $t = 6000$s for performance estimation and comparison. This section was chosen by visually identifying the initial transient and rare event occurrences. Appendix C shows an example of the identification of initial transient and rare events.

During these 2000s, blocking probability samples were taken every 5 seconds; the mean blocking probability was calculated using these 5s-samples. We chose 5s since we had performance data for a 2000s section of simulation time. This resulted in a sufficient representation of the standard deviation of the performance statistic.

The delay and jitter samples were first aggregated per connection and then over an entire TC. We consider a user's connection as already being closed if its priority traffic has a delay longer than or equal to 30s. UGS and rtPS PDUs with delays or jitters larger than 30s were not considered when calculating delay and jitter statistics.

7.3 RESULTS

We ran simulations for the four CAC/scheduler pairs and for six different arrival rates, i.e. 24 simulation runs, in parallel, four runs at a time. Two 2.13 Intel Core 2 Duo machines with two GB of RAM each were used and generated approximately 200 GB of trace data, taking about two weeks. The trace data were processed, requiring several more hours. The resulting statistics are as follows:

As is found in related literature, such as by Ali *et al.* [3], by changing the scheduler, packet-level performance is effected. This expectation is verified by considering the mean [1] UGS and rtPS delay and jitter as workload intensity varies: Plots of mean UGS and rtPS delay are shown in Figures 7.2 and 7.3, respectively; mean UGS and rtPS jitter are shown in Figures 7.4 and 7.5, respectively.

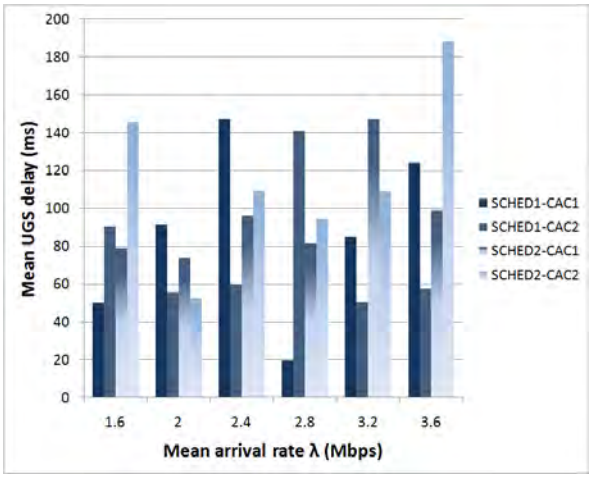

Figure 7.2: Mean UGS delay for different workload intensity settings, showing the different RaCM configurations' results

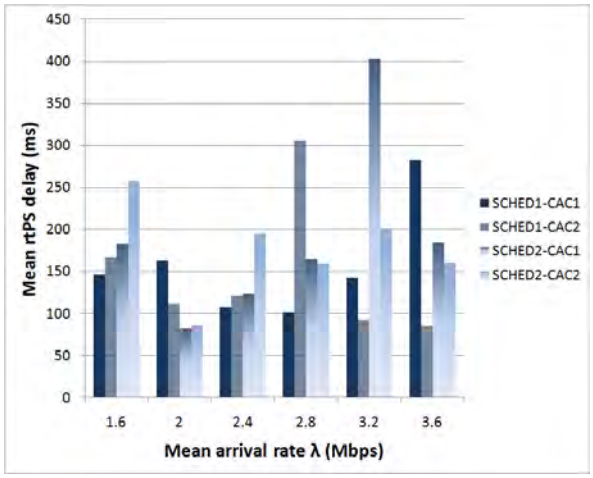

Figure 7.3: Mean rtPS delay for different workload intensity settings, showing the different RaCM configurations' results

Even though Ali *et al.* mention the importance of the CAC, which "...works in tandem with the scheduling algorithm..." [3], they describe the design of the schedulers without investigating the effect of the particular inter-dependent relationships between each of the CAC-scheduler configurations. Inspecting the delay and jitter results, it is apparent that, by operating different CAC configurations with the same system scheduler configuration, the network performance at the packet-level is affected significantly, i.e. up to 600%, as is apparent in Figure 7.2 for SCHED1 experiments and workload intensity 2.8 Mbps. This proves that there exists a dependency relationship: The scheduler-managed performance is directly affected by the particular CAC employed and therefore, the scheduler is dependent on the CAC, although it is left for future work to determine this relationship. It should also be pointed out that the packet-level performance is impacted differently *for each of the different workload intensities*.

The graphs of mean UGS, rtPS and BE throughputs are show in Figures 7.6 through 7.8; Figure 7.9 shows the mean throughput for the combined traffic in all the categories. There does not seem to be a significant difference in the mean throughput metric for the different experiments. This is to be expected since throughput is measured per TC as an aggregation of connection throughputs, different from the delay and jitter metrics.

[1] Note that all mean values are computed using samples taken from the comparable observation period starting at simulation time $t = 4000$s and ending at $t = 6000$s, as mentioned in Section 7.2.3

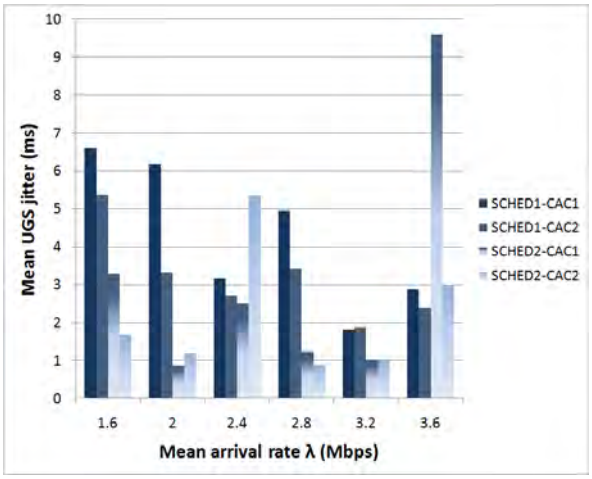

Figure 7.4: Mean UGS jitter for different workload intensity settings, showing the different RaCM configurations' results

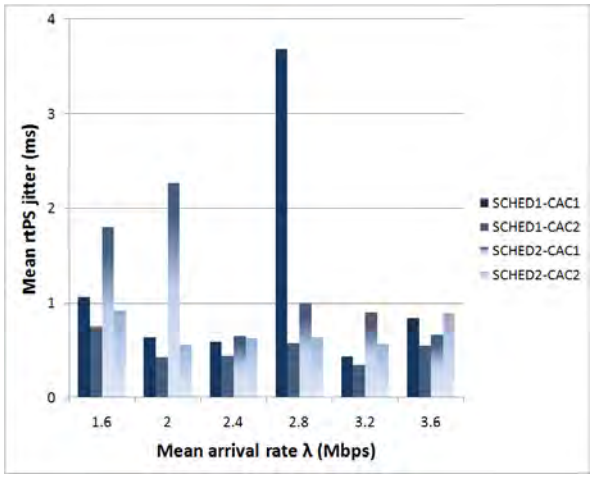

Figure 7.5: Mean rtPS jitter for different workload intensity settings, showing the different RaCM configurations' results

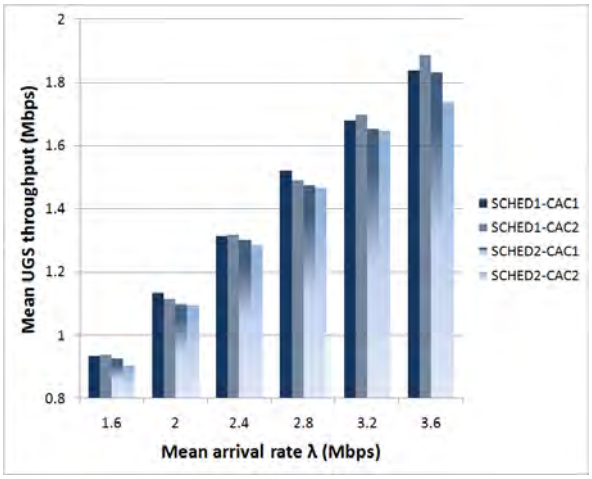

Figure 7.6: Mean UGS throughput for different workload intensity settings, showing the different RaCM configurations' results

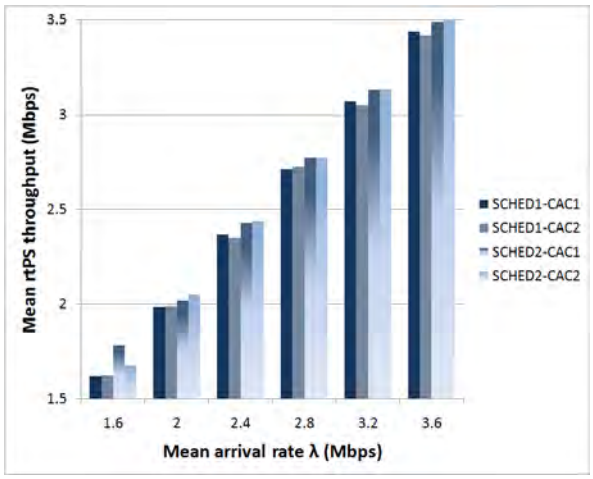

Figure 7.7: Mean rtPS throughput for different workload intensity settings, showing the different RaCM configurations' results

The mean UGS and rtPS throughputs increase monotonically for an increasing mean arrival rate, as well as for the overall TC aggregate – as one would expect. The mean BE throughput initially increases; at some point it starts to decrease as the UGS and rtPS throughputs become sufficiently large. This illustrates how the scheduler uses BE data effectively when there is little or no priority traffic. Additionally, Figure 7.9 shows the overall mean throughput increasing at a decreasing rate as the wireless link becomes fully utilised.

Similarly for the connection-level performance metric, i.e. connection blocking probability, the mean blocking probability is expected to differ when experimenting with different CACs and a fixed system scheduler configuration. These differences are evident, as shown in Figures 7.10 and 7.11 for UGS and rtPS, respectively. More importantly however, by fixing the CAC and experimenting with different scheduler configurations, it is also apparent that the mean blocking probability differs significantly, i.e. up to 35% as is apparent in Figure 7.10 for the CAC1 experiments and workload intensity 3.6 Mbps. This implies that there is a dependency relationship

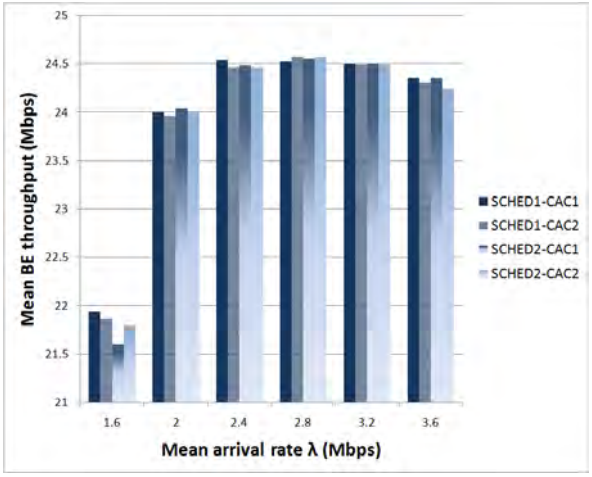

Figure 7.8: Mean BE throughput for different workload intensity settings, showing the different RaCM configurations' results

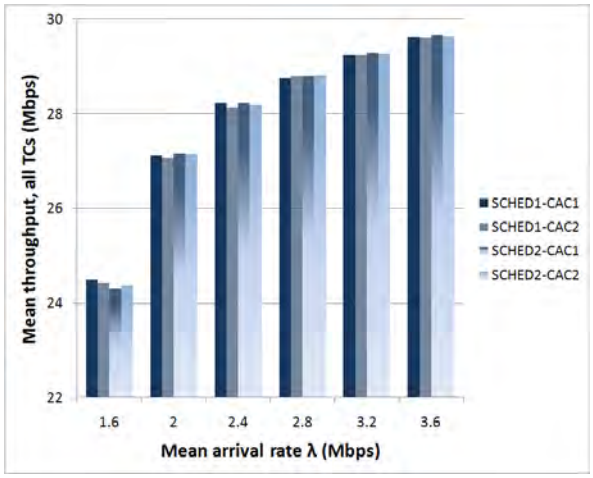

Figure 7.9: Mean overall throughput for different workload intensity settings, showing the different RaCM configurations' results

between the CAC and scheduler components; this is a relationship in which the CAC is dependent on the particular scheduler.

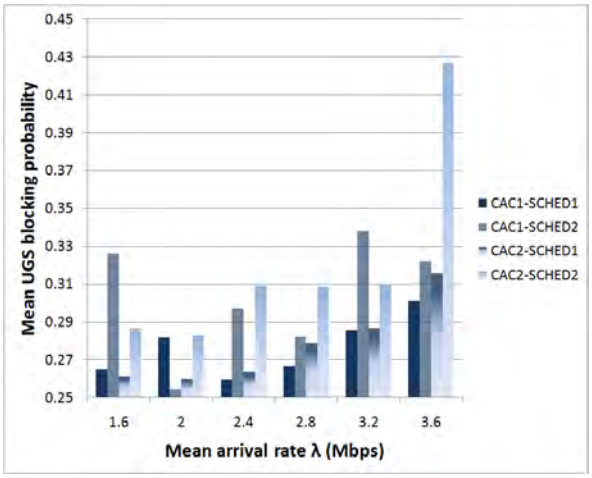

Figure 7.10: Mean UGS blocking probability for different workload intensity settings, showing the different RaCM configurations' results

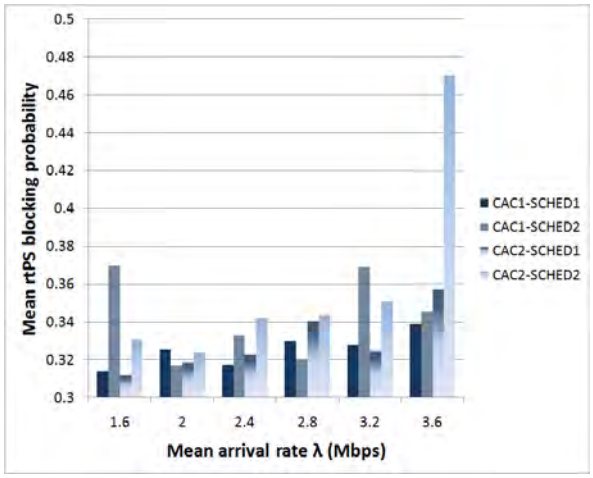

Figure 7.11: Mean rtPS blocking probability for different workload intensity settings, showing the different RaCM configurations' results

The results therefore suggest that there is an interdependent relationship between the CAC and scheduler components, as hypothesised: There is a synergism between CAC and scheduler. As mentioned before, the standard deviation may also be used to illustrate the synergy.

Rather than repeat all the mean value results, consider the standard deviation for UGS and rtPS delay, shown in Figures 7.12 and 7.13, respectively. Performance data where the scheduler remains the same but the CAC is changed, are considered in particular, namely $SCHED1 - CAC1$ and $SCHED1 - CAC2$; each for workload intensities 2.4 and 3.6 Mbps. Finally, as the independent variable, we compute the standard deviation in delay for consecutive 1000s *intervals*, with the first starting at $t = 1000$s and the last starting at $t = 4000$s.

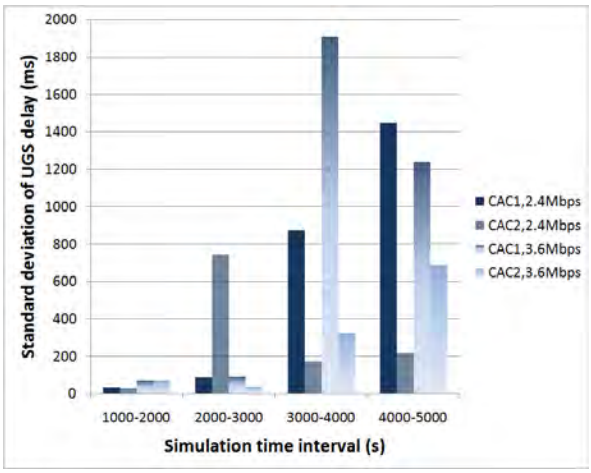

Figure 7.12: Standard deviation of UGS delay for consecutive 1000s intervals, showing the different CAC configurations and different workload intensity settings

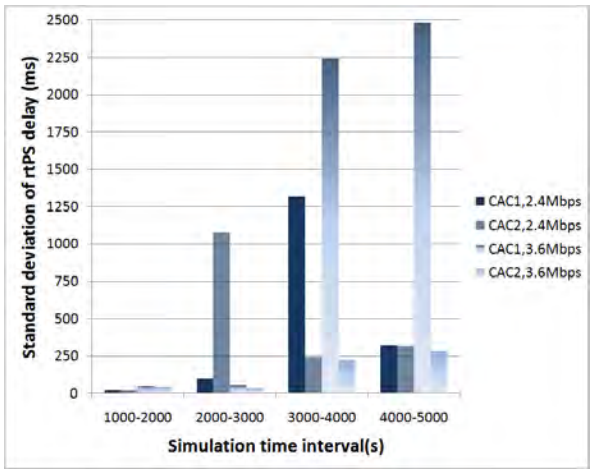

Figure 7.13: Standard deviation of rtPS delay for consecutive 1000s intervals, showing the different CAC configurations and different workload intensity settings

For both workload intensities, it is apparent that a significant difference in the standard deviation of UGS and rtPS delay results for different CAC configurations, confirming once more that the scheduler is dependent on the particular CAC employed.

An important aspect of our experimental methodology is also clearly shown in Figures 7.12 and 7.13: By considering consecutive 1000s intervals for the same workload intensity and RaCM configuration, the standard deviations of UGS and rtPS delay both fluctuate greatly over time. It is therefore not correct to attempt determining the true mean of system performance indices for the system operation time tending to infinity. Rather, as just shown, we considered a mean for a more relevant period of time, specified in Section 7.2.3.

Finally, it is important to note that, in the case of both connection– and packet-level statistics, it appears that the extent of the degree in performance differences are not necessarily constant. This is firstly attributed to the complex nature of the system: Again, as shown in Chapter 6, the selection of an important performance-impacting parameter (i.e. UL/DL frame ratio) is, *inter alia*, a function of the arrival rate. Lastly, one must take into account the observation interval over which performance data was observed, as explained in the previous paragraph.

CHAPTER 8

CONCLUSION AND FUTURE WORK

The objective of this project was to prove the hypothesis that there is a synergy between connection admission control and scheduling in IEEE 802.16. This has an important implication for RaCM design: There must be a shift from designing admission control and scheduling algorithms independently to acknowledging their co-dependence in cross-layer RaCM design.

Typically, designers choose either one or both of these components based on the independent performance evaluations of them. Instead, one should start with basic and simplistic admission control and scheduler skeletons and iteratively adjust each to ultimately arrive at an optimal RaCM design. By following this methodology, one would be designing a performance-aware RaCM rather than an admission controller and scheduler that could possibly have a negative impact on system performance. Describing the inter-RaCM relationship, however, was not attempted, since we only wanted to show initially that there is in fact an inter-RaCM synergy unique to each scheduler and CAC pair.

Another factor that effects the nature of the particular synergism is the workload. The system has two layers of workload behaviour, namely connection and packet behaviour. Since a change in connection behaviour directly changes the impact of the admission controller on connection-level performance, it indirectly affects the scheduler; since a change in packet behaviour directly changes the impact of the scheduler on packet-level performance, it indirectly affects the admission controller.

As is typical with most simulation studies of complex systems, our simulation runs took a relatively long time to complete and generated vast amounts of performance data. We found that simulation runs took a relatively long time to stabilise with respect to the mean connection holding times. Furthermore, the variance was high. This is not unexpected since we had dynamic connection– and packet-level workload behaviour, which meant that the system behaviour is not homogeneous. A comparable cross-section of time, i.e. continuous period/section of time, containing a large amount of consequential data was used to gain insight into relevant system performance.

Finally, considering both the inter-RaCM design and connection– and packet-level dynamics, we are faced with another problem: The non-homogeneous behaviour of the system makes traditional QoS performance metrics unsuitable indicators of the quality of experience (QoE) of the average individual user. As a first step, we would suggest that one better relates QoS to a QoE. Also, given the computational complexity and the physical memory

required for a simulation run, an innovative (possibly hybrid) approach is necessary for developing a more feasible performance model. It is our opinion that, when developing the performance model, it is highly beneficial to keep in mind that, and make allowance for, other performance impacting factors, such as channel quality, and so on. The system is very complex and may therefore be extremely sensitive to such factors. In particular, a challenging open problem is RaCM support of *elastic* media services.

We have identified and described some important aspects of the synergistic relationship between admission control and scheduling components in this project. However, we did not describe particular implicit aspects. We expect that each RaCM will have a unique relationship. Therefore, defining the synergism for particular RaCMs is an interesting and fruitful avenue of future work.

BIBLIOGRAPHY

[1] A. Adas, "Traffic models in broadband networks," *IEEE Communications Magazine*, vol. 35, no. 7, pp. 82–89, July 1997.

[2] Alexander Klemm and Christoph Lindemann and Mary K Vernon and Oliver P Waldhorst, "Characterizing the Query Behavior in Peer-to-Peer File Sharing Systems," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement 2004*, Alfio Lombardo and James F. Kurose, Ed. Taormina, Sicily, Italy: ACM, 2004, pp. 55–67. [Online]. Available: http://doi.acm.org/10.1145/1028788.1028796

[3] N. A. A. Ali, P. Dhrona, and H. S. Hassanein, "A performance study of uplink scheduling algorithms in point-to-multipoint WiMAX networks," *Computer Communications*, vol. 32, no. 3, pp. 511–521, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2008.09.015

[4] F. Amoroso, "On the Convolutional Square Root of a Nyquist Pulse," *Wireless Personal Communications*, vol. 1, pp. 287–290, 1995.

[5] S. Balakrishnan and F. Özgüner, "A Priority-Driven Flow Control Mechanism for Real-Time Traffic in Multiprocessor Networks," *IEEE Transactions on Parallel Distributed Systems*, vol. 9, no. 7, pp. 664–678, 1998.

[6] J. Banks, J. S. Carson, B. L. Nelson, and D. M. Nicol, *Discrete-Event System Simulation*, Third ed. Prentice-Hall, 2000.

[7] N. Basher, A. Mahanti, A. Mahanti, C. Williamson, and M. Arlitt, "A Comparative Analysis of Web and Peer-to-Peer Traffic," in *Proceeding of the Seventeenth International Conference on World Wide Web (WWW '08)*. New York, NY, USA: ACM, 2008, pp. 287–296.

[8] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for Differentiated Services. Internet RFC 2475," 1998.

[9] D. Cavin, Y. Sasson, and A. Schiper, "On the accuracy of MANET simulators," in *POMC '02: Proceedings of the second ACM international workshop on Principles of mobile computing*. New York, NY, USA: ACM, 2002, pp. 38–43.

[10] S. Chandra and A. Sahoo, "An Efficient Call Admission Control for IEEE 802.16 Networks," in *IEEE LANMAN*, June 2007.

[11] J. Chen, W. Jiao, and H. Wang, "A service flow management strategy for IEEE 802.16 broadband wireless access systems in TDD mode," in *IEEE International Conference on Communications*, 2005, pp. 3422–3426.

[12] H.-K. Choi and J. O. Limb, "A Behavioral Model of Web Traffic," in *ICNP '99: Proceedings of the Seventh Annual International Conference on Network Protocols*. Washington, DC, USA: IEEE Computer Society, 1999, p. 327.

[13] C.-N. Chuah, "A Scalable Framework for IP-Network Resource Provisioning Through Aggregation and Hierarchical Control," Ph.D. dissertation, University of California at Berkeley, 2001.

[14] C. Cicconetti, L. Lenzini, E. Mingozzi, and C. Eklund, "Quality of Service Support in IEEE 802.16 Networks," *Network, IEEE*, vol. 20, no. 2, pp. 50–55, March–April 2006.

[15] C. Cicconetti, A. Erta, L. Lenzini, and E. Mingozzi, "Performance Evaluation of the IEEE 802.16 MAC for QoS Support," *IEEE Transactions in Mobile Computing*, vol. 6, no. 1, pp. 26–38, 2007. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TMC.2007.12

[16] N. Cranley and M. Davis, "Performance evaluation of video streaming with background traffic over IEEE 802.11 WLAN networks," in *WMuNeP'05 - Proc. 1st ACM Workshop on Wireless Multimedia Networking and Performance Modeling*, A. A. F. Loureiro and W. Zhuang, Eds. ACM, 2005, pp. 131–139.

[17] K. L. D. Staehle and P. Tran-Gia, "Source Traffic Modeling of Wireless Applications," University of Würzburg, Technical Report TR 261, 1999.

[18] Demers, A, Keshav, S, and Shenker, S, "Analysis and Simulation of a Fair Queueing Algorithm," *Internetworking: Research and Experience*, vol. 1, pp. 3–26, apr 1990.

[19] D. Erman, D. Ilie, and A. Popescu, "BitTorrent Session Characteristics and Models," in *Proceedings of the Third International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HET-NETs'05)*, Ilkley, United Kingdom, 2005.

[20] D. Ferrari and D. C. Verma, "A scheme for real-time channel establishment in wide-area networks," in *IEEE Journal on Selected Areas in Communication*, vol. 8, no. 3, April 1990, pp. 368–379.

[21] V. S. Frost and B. Melamed, "Traffic Modeling for Telecommunications Networks," *IEEE Communications Magazine*, pp. 70–81, March 1994.

[22] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, "Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload," in *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles (SOSP '03)*. New York, NY, USA: ACM, 2003, pp. 314–329.

[23] Q. He, "Analysing The Characteristics of VoIP Traffic," July 2007.

[24] J. A. Hernández, I. W. Phillips, and J. Aracil, "Discrete-time heavy-tailed chains, and their properties in modeling network traffic," *ACM Transactions on Modeling and Computer Simulation*, vol. 17, no. 4, pp. 17:1–17:11, Sep. 2007.

[25] J. Hou, J. Yang, and S. Papavassiliou, "Integration of Pricing with Call Admission Control to Meet QoS Requirements in Cellular Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. PDS-13, no. 9, pp. 898–910, Sep 2002.

[26] G. Iazeolla, P. S. Kritzinger, and P. P. Pileggi, "Modelling Quality of Service in IEEE 802.16 Networks," in *Software, Telecommunications and Computer Networks, 2008. SoftCOM 2008. 16th International Conference on*, 2008, pp. 130–134.

[27] IEEE, *IEEE Standard for Local and Metropolitan Area Networks*, IEEE 802.16 Standard, 2004.

[28] O. T. Incorporated, "OPNET Modeler," http://www.opnet.com.

[29] ITU-T, "Recommendation G.711 - Pulse code modulation (PCM) of voice frequencies," *Geneva, Switzerland*, 1988.

[30] ——, "Recommendation P.59 – Artificial conversational speech," *Geneva, Switzerland*, 1994.

[31] M. Kargahi and A. Movaghar, "A method for performance analysis of earliest-deadline-first scheduling policy," *J. Supercomput.*, vol. 37, no. 2, pp. 197–222, 2006.

[32] A. Klemm, C. Lindemann, and M. Lohmann, "Modeling IP traffic using the batch Markovian arrival process," *Performance Evaluation*, pp. 149–173, 2003.

[33] H. Kobayashi and B. L. Mark, *System Modeling and Analysis*. New Jersey, 07458: Prentice Hall, 2009.

[34] A. M. Law and D. W. Kelton, *Simulation Modeling and Analysis*. McGraw-Hill, 2000.

[35] P. L'Ecuyer and E. Buist, "Simulation in Java with SSJ," in *Simulation Conference. 2005 Proceedings of the Winter,*, 2005.

[36] J. P. Lehoczky, "Using real-time queueing theory to control lateness in real-time systems," in *SIGMETRICS '97: Proceedings of the 1997 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM, 1997, pp. 158–168.

[37] Q. Liu, X. Wang, and G. B. Giannakis, "Cross-Layer Scheduler Design with QoS Support for Wireless Access Networks," in *QSHINE*. IEEE Computer Society, 2005, p. 21. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/QSHINE.2005.16

[38] I. C. Msadaa, F. Kamoun, and F. Filali, "An Adaptive QoS Architecture for IEEE 802.16 Broadband Wireless Networks," in *Mobile Adhoc and Sensor Systems (MASS 2007). IEEE Internatonal Conference on*, 2007, pp. 1–3.

[39] D. Niyato and E. Hossain, "Analysis of Fair Scheduling and Connection Admission Control in Differentiated Services Wireless Networks," in *IEEE International Conference on Communications*, 2005, pp. 3137–3141.

[40] ——, "Call admission control for QoS provisioning in 4G wireless networks: issues and approaches," *IEEE Networks*, vol. 19, no. 5, pp. 5–11, September 2005.

[41] NS-2, "The Network Simulator - NS-2," http://www.isi.edu/nsnam/ns.

[42] OMNET++, "A discrete event simulation system," http://www.omnetpp.org.

[43] On-line, "BitTorrent™," http://www.bittorrent.com/, 2009.

[44] ——, "Gnutella – A protocol for revolution," http://rfc-gnutella.sourceforge.net/, 2009.

[45] ——, "Kazaa.com," http://www.kazaa.com/, 2009.

[46] ——, "Napster Free – Listen to free streaming music online," http://free.napster.com/, 2009.

[47] M. Papadopouli, H. Shen, and M. Spanakis, "Characterizing the mobility and association patterns of wireless users in a campus," Department of Computer Science, University of North Carolina - Chapel Hill, Tech. Rep. TR04-019, Jul 2004. [Online]. Available: ftp://ftp.cs.unc.edu/pub/publications/techreports/04-019.pdf

[48] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks - The Single Node Case," in *INFOCOM*, 1992, pp. 915–924.

[49] V. Paxson and S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, June 1995.

[50] M. Perényi, T. D. Dang, A. Gefferth, and S. Molnár, "Identification and Analysis of Peer-to-Peer Traffic," *JCM*, vol. 1, no. 7, pp. 36–46, 2006.

[51] H. G. Perros, *An Introduction to ATM Networks*. New York, NY, USA: John Wiley & Sons, Inc., 2002.

[52] H. K. Rath, A. Bhorkar, and V. Sharma, "An Opportunistic Uplink Scheduling Scheme to Achieve Bandwidth Fairness and Delay for Multiclass Traffic in Wi-Max (IEEE 802.16) Broadband Wireless Networks," in *IEEE Global Telecommunications Conference (GLOBECOM '06)*, 2006, pp. 1–5.

[53] N. Ruangchaijatupon, L. Wang, and Y. Ji, "A Study on the Performance of Scheduling Schemes for Broadband Wireless Access Networks," in *Communications and Information Technologies. ISCIT '06. International Symposium on*, 2006, pp. 1009–1012.

[54] P. Salvador, A. Pacheco, and R. Valadas, "Modeling IP traffic: joint characterization of packet arrivals and packet sizes using BMAPs," *Computer Networks*, pp. 335–352, October 2003.

[55] A. Sayenko, O. Alanen, and T. Hämäläinen, "Scheduling solution for the IEEE 802.16 base station," *Computer Networks*, vol. 52, no. 1, pp. 96–115, 2008. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2007.09.021

[56] H. Schulze and K. Mochalski, "Internet Study 2007," http://www.ipoque.com/resources/internet-studies/internet-study-2007, 2007.

[57] J. Seger, "Modelling Approach for VoIP Traffic Aggregations for Transferring Tele-traffic Trunks in a QoS enabled IP-Backbone Environment," in *International Workshop on Inter-domain Performance and Simulation*, 2003, Faculty for Electrical Engineering and Information Technology, Department of Electronic Systems and Switching, University of Dortmund.

[58] S. Shin and B.-H. Ryu, "Packet Loss Fair Scheduling Scheme for Real-Time Traffic in OFDMA Systems," pp. 391–396, Oct 2004.

[59] D. Stiliadis and A. Varma, "Latency-Rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms," in *INFOCOM*, 1996, pp. 111–119.

[60] A. Symington, "A Hardware Testbed for Measuring IEEE 802.11g DCF Performance," Master's thesis, Department of Computer Science, University of Cape Town, December 2009.

[61] T. W. Tang, D. Green, M. Rumsewicz, and N. Bean, "An Architecture for IEEE 802.16 MAC Scheduler Design," in *15th IEEE International Conference on Networks (ICON 2007)*, 2007, pp. 89–94.

[62] S. N. Technologies, "Introducing the VisNet Network Planning Software," http://www.scalable-networks.com.

[63] A. Varga, "The OMNET++ discrete event simulation system," in *Proceedings of the European Simulation Multiconference*. Prague, Czech Republic: SCS - European Publishing House, 2001, pp. 319–324.

[64] L. O. Walters, "A Web Browsing Workload Model For Simulation," Master's thesis, University of Cape Town, May 2004.

[65] H. Wang, B. He, and D. P. Agrawal, "Above packet level admission control and bandwidth allocation for IEEE 802.16 wireless MAN," *Simulation Modelling Practice and Theory*, vol. 15, no. 4, pp. 366–382, April 2007.

[66] J. Watkins, *Testing IT - An Off-the-Shelf Software Testing Process*. Cambridge University Press, 2001.

[67] K. Wongthavarawat and A. Ganz, "Packet scheduling for QoS support in IEEE 802.16," *International Journal of Communication Systems*, vol. 16, pp. 81–96, 2003.

[68] H. Xu, "Video Streaming Traffic Model for 802.16m Evaluation Methodology Document," November 2007, IEEE 802.16 Broadband Wireless Access Working Group.

[69] J. Yeo, D. Kotz, and T. Henderson, "CRAWDAD: a community resource for archiving wireless data at Dartmouth," *SIGCOMM Computer Commuication Review*, vol. 36, no. 2, pp. 21–22, 2006.

APPENDIX A

SIMULATOR DEVELOPMENT

Our simulator was developed according to the process shown in Figure A.1. The simulation engine, RaCM and workload components are first designed individually. These are then implemented and tested in turn. Once all the components have tested correctly, they are integrated. After the integrated implementation has been tested, component interfaces are specified. On completion of the interface implementation, version 1.0 of the simulator is complete.

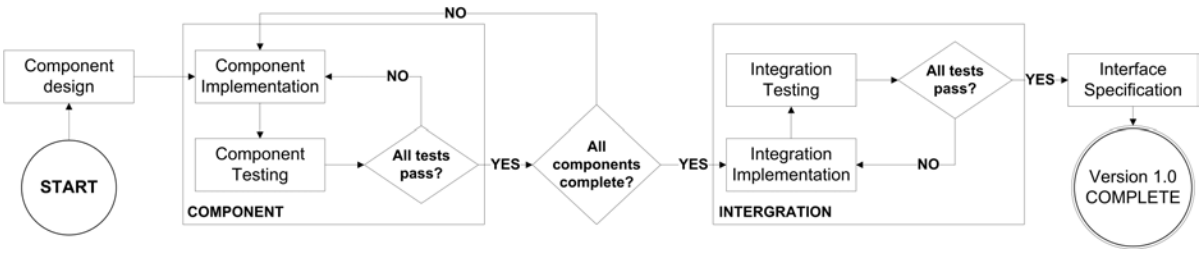

Figure A.1: Overview of the simulator development process

A.1 DESIGN

This section describes the design choices made for the simulation engine, RaCM and workload components, where the latter two components were described in Chapter 2.

A.1.1 Simulation Engine Design

The simulation engine is event-driven, where an event represents a discrete change to one or more state variables at a particular moment in time[1]. The main components of the basic simulation engine and their respective functions are as follows.

Clock – The clock keeps track of simulation time.

[1]To avoid misinterpretation of the term *simulation time*, we define *simulation time* as the virtual time of the system being simulated, whereas *wall-time* is defined as the real time taken for the simulation run to complete.

Calendar – The calendar, also referred to as the event list ("future events" list to some), keeps track of the the next occurrence of each type of event that may occur.

Scheduler – Also referred to as the sequencer, the scheduler[2] determines the next event to occur, advances the clock to the time at which this event must occur and executes the event routine corresponding to this event.

Event routines – These routines are the processes that must execute to change the state variables affected by the occurrence of a specific event.

Figure A.2 illustrates an abstract process flow chart (PFC) for the simulator and also shows the above-mentioned components and their interactions. When the simulator is started, the simulation model is initialised using the parameters provided to the program. Once initialised, control is passed to the scheduler. As shown in the figure, the scheduler interacts with the calendar and the next event to execute is identified. Thereafter, the scheduler advances the clock to the time of the next event identified. Lastly, the scheduler passes control the the event routine process block that must execute next. On completion of the event process flow execution, simulator control is passed back to the scheduler and this cycle continues until an end-of-simulation condition is reached. In our case, the end-of-simulation condition is the simulation run duration and a separate event is created for terminating the program. When control is passed to the end-of-simulation event, the simulation run is over and general house-keeping routines are executed.

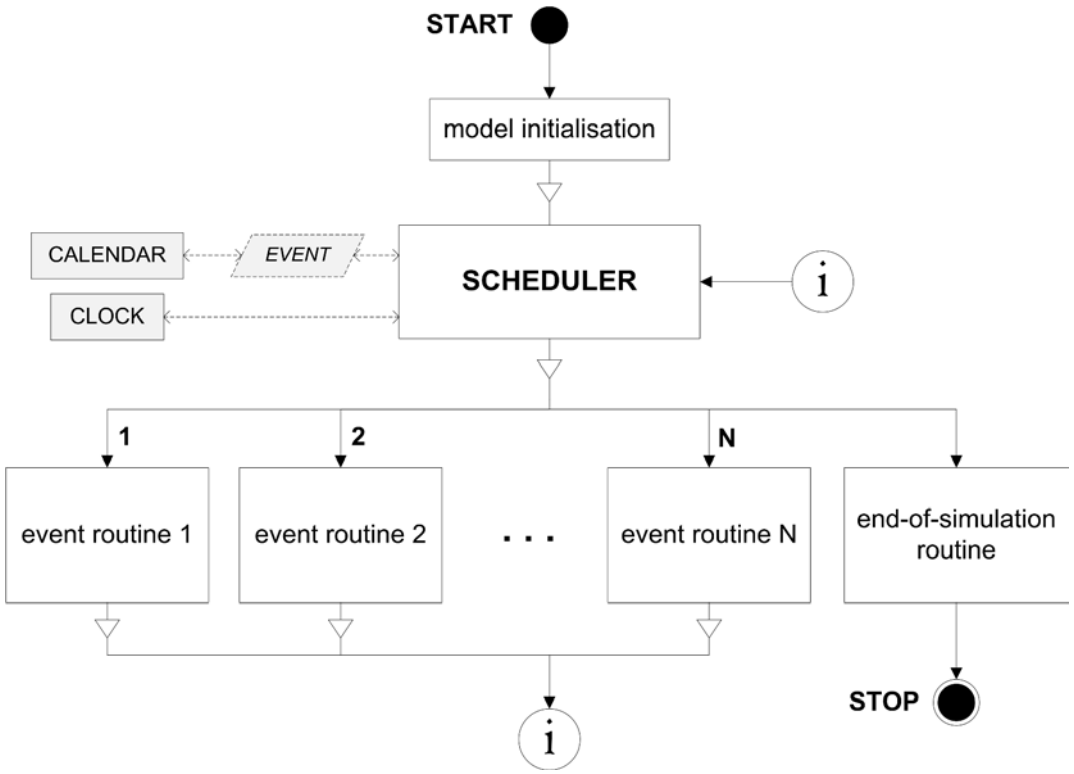

Figure A.2: Basic simulation engine PFC, showing the main components and their interactions

[2]The simulation engine scheduler is not to be confused with the BS or SS scheduler components.

A.1.2 Simulator Event Design

Event Identification

Altogether 11 events were identified: 4 events were identified from the abstracted frame, as shown in Figure A.3, namely

- **EOULSF**: End of UL subframe.
- **NEXTDIUC**: Next DL interval usage code, representing the start of the next DL profile.
- **EODLSF**: End of DL subframe.
- **NEXTUIUC**: Next UL interval usage code, representing the start of the next UL profile, including the UL connection contention period.

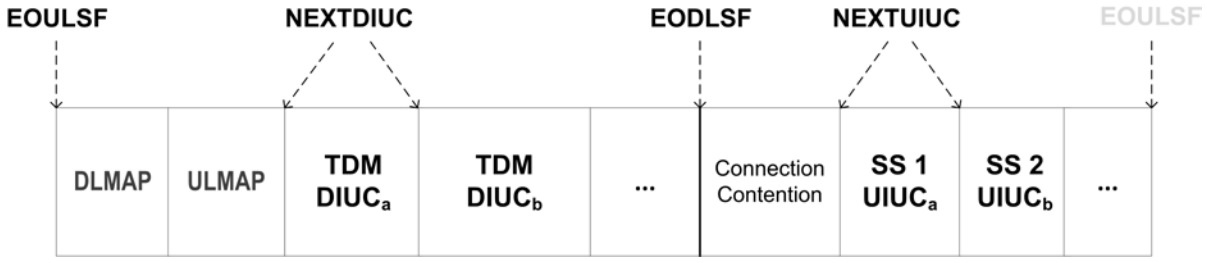

Figure A.3: Abstracted functional frame showing the four relevant events identified

The scheduler implements both an end-of-UL event and an end-of-DL subframe event even though it is possible to reduce these events to a single event. Even though this may significantly impact the wall-time performance of the simulator, it results in a more comprehensible model; reducing the number of events to such an extreme would result in a more complex model that is harder to validate, ultimately discounting its credibility.

An additional 5 events were identified from the NoQ model, as shown in Figure A.3, namely

- **SSARR**: SS arrival, representing a PDU arrival at one of the SSs in the list of SSs.
- **FARR**: Fixed-line arrival, representing the arrival of a PDU from the internet at the BS wireless DL waiting-line.
- **WARR**: Arrival at the wireless interface of the BS, representing the end of service of a PDU in the UL direction from an SS and the arrival of the PDU at the BS wireless DL waiting-line or internet node.
- **EOSWDL**: End of service of the wireless DL server, representing just that.
- **BWRARR**: BWR arrival, representing the arrival of a SS or internet BWR at the BS CAC waiting-line.

Finally, 2 events were identified from the general simulator capabilities needed, namely the sampling (**SAMPLE**) and end-of-simulation (**ENDSIM**) events. The former periodically samples dependent variables (such as queue lengths) while the latter, as mentioned before, performs house-keeping functions at the end of the run.

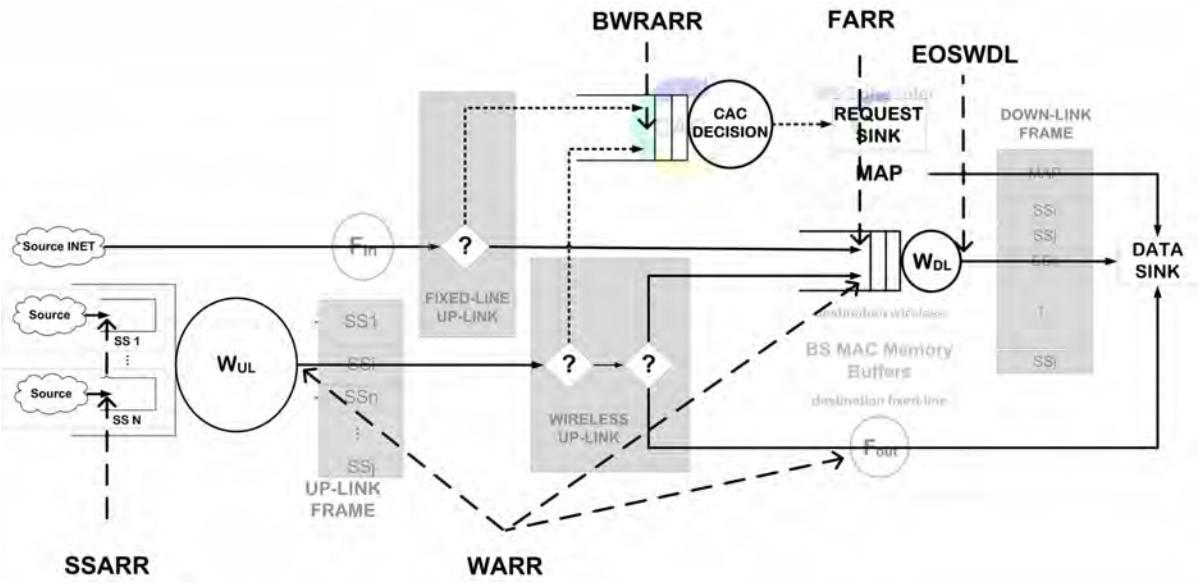

Figure A.4: Network of queues model, abstracting both wireless and fixed-line data and BW requests at the BS per TC, showing the 5 identified events

Event Process Flows

In this section, the 11 events are described in terms of the various process flows which may execute on occurrence of each. The process flows per event are presented as PFCs and necessary chart variables are defined in Table A.1.

Variable	Definition
t_{SIM}	Present simulation time.
t_S	Service time of a PDU.
t_{MAPS}	Time required to transmit the MAP information.
t_{DLSF}	Duration of the DL subframe.
t_{ULSF}	Duration of the UL subframe.
t_{IE}	Duration of an IE of either DLMAP or ULMAP.
t_{SSAT}	Arrival time of the next PDU at some SS.
t_{FAT}	Arrival time of the next PDU at the INET.
t_{BWRAT}	Arrival time of the next BWR from either the INET or an SS.
$index$	SSID of the SS that may transmit on the UL to the BS during the current UL profile period. -1 implies that no SS has been selected and -2 implies the connection contention period during which all SSs may transmit BWRs.
$BSTransmit$	DL transmission status of the BS. $true$ implies the BS may transmit and $false$ implies it may not.

Table A.1: PFC variable definitions

The event PFCs that follow descibe (in detail) the event routine process flow blocks shown in Figure A.5.

EOULSF

EOULSF indicates both the beginning of the DL subframe and the end of the UL subframe and its PFC is shown in Figure A.6. As shown that the figure, the simulator must first determine whether the last UL profile period was

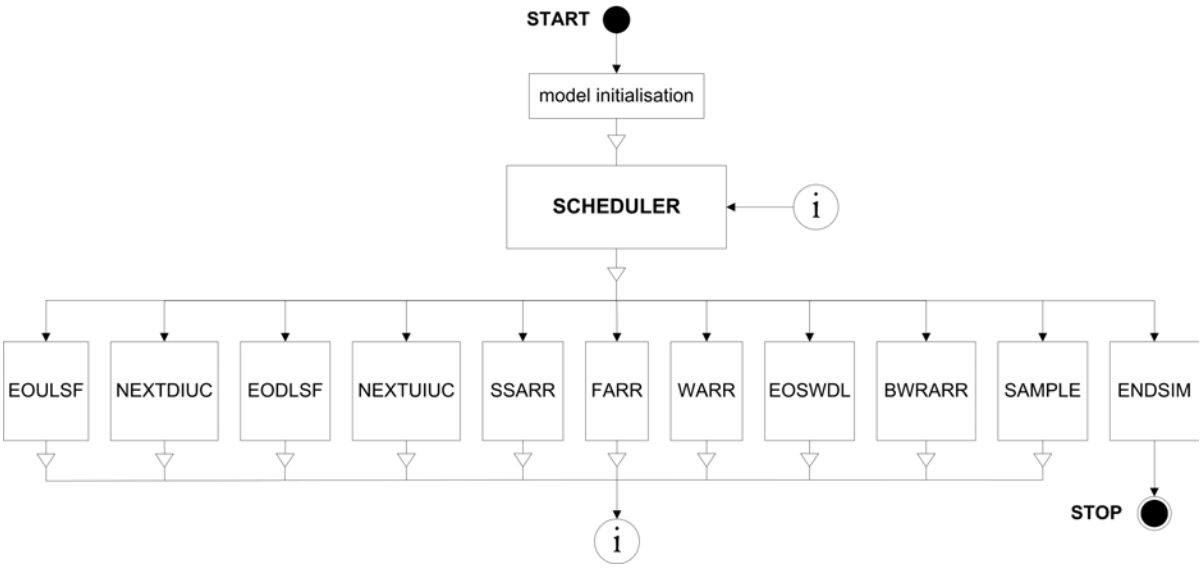

Figure A.5: Basic simulation engine PFC with the process flow blocks of the event routines

a connection contention period. If $index == -2$, then it was, and the current admitted connection list used by the BS scheduler to generate the MAPs must be updated with the delayed connection admission list; a process described in Chapter 4.2. All BWRs in the CAC waiting line are then served. $index$ is set to -1 to indicate that no SS has access to the UL medium. Next, the antenna state is changed from UL to DL. This means that the BS's antenna will be in transmit mode and that the SSs' antennae will be in receive mode. Since TDD single-carrier PHY is modelled, the antenna state is either UL or DL. The current DLMAP and ULMAP are then replaced with the new DLMAP and ULMAP, respectively, as specified by the MAP generator of the BS scheduler. If this DLMAP is empty, i.e. contains no IEs, NEXTDIUC is disabled[3] since there may not be any DL transmission. If the DLMAP is not empty, NEXTDIUC must be scheduled[4] at time $t_{SIM} + t_{MAPS}$. Similarly then, if the ULMAP is empty, NEXTUIUC is disabled, otherwise it is scheduled at time $t_{SIM} + t_{DLSF}$. Finally, EODLSF is scheduled at $t_{SIM} + t_{DLSF}$ and EOULSF and WARR are disabled.

NEXTDIUC

NEXTDIUC changes the system by updating the DL transmission profile. Its PFC is shown in Figure A.7. The IE at the front of the DLMAP is used to update the modulation technique to be used during the this specific DL transmission period, effectively changing the rate of the DL transmission, and is then removed from the DLMAP. After this, the $BSTransmit$ variable is set to $true$ to indicate that the BS may now transmit along the DL. If the wireless DL waiting lines (at the BS) have PDUs waiting to be served, the next PDU (in these waiting lines) is put into service and EOSWDL is scheduled at $t_{SIM} + t_S$. Otherwise, if these waiting lines are empty, EOSWDL is not scheduled. Lastly, if the DLMAP is empty, NEXTDIUC is disabled, otherwise, if the DLMAP is not empty, NEXTDIUC is scheduled at time $t_{SIM} + t_{IE}$, where t_{IE} is the duration of the IE that has most recently been removed off the DLMAP.

EODLSF

EODLSF indicates both the beginning of the UL subframe and the end of the DL subframe and its PFC is shown

[3]Disabling an event involves ensuring that the event will not occur again unless explicitly scheduled to occur by some other event. The way in which this is done in the simulator is to schedule the event to occur at $time = \infty$, which is the maximum value of the data primitive used to store simulated time.

[4]Scheduling an event means to set the next occurrence time of that event.

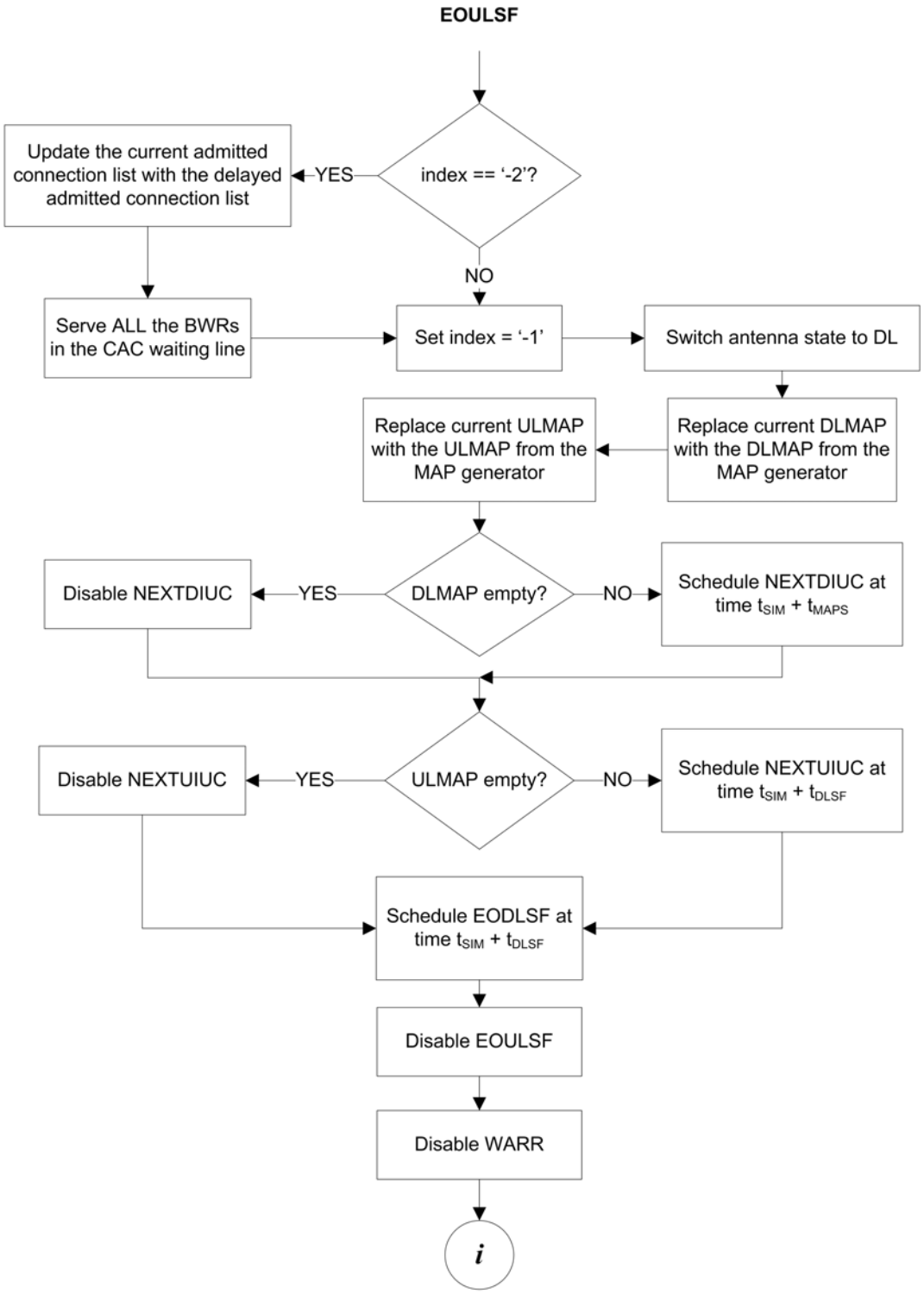

Figure A.6: EOULSF PFC

in Figure A.6. The $BSTransmit$ variable is first set to $false$ since the BS may not transmit more data. Next, the antenna state is changed from DL to UL and then EOULSF is scheduled at time $t_{SIM} + t_{ULSF}$. Note that,

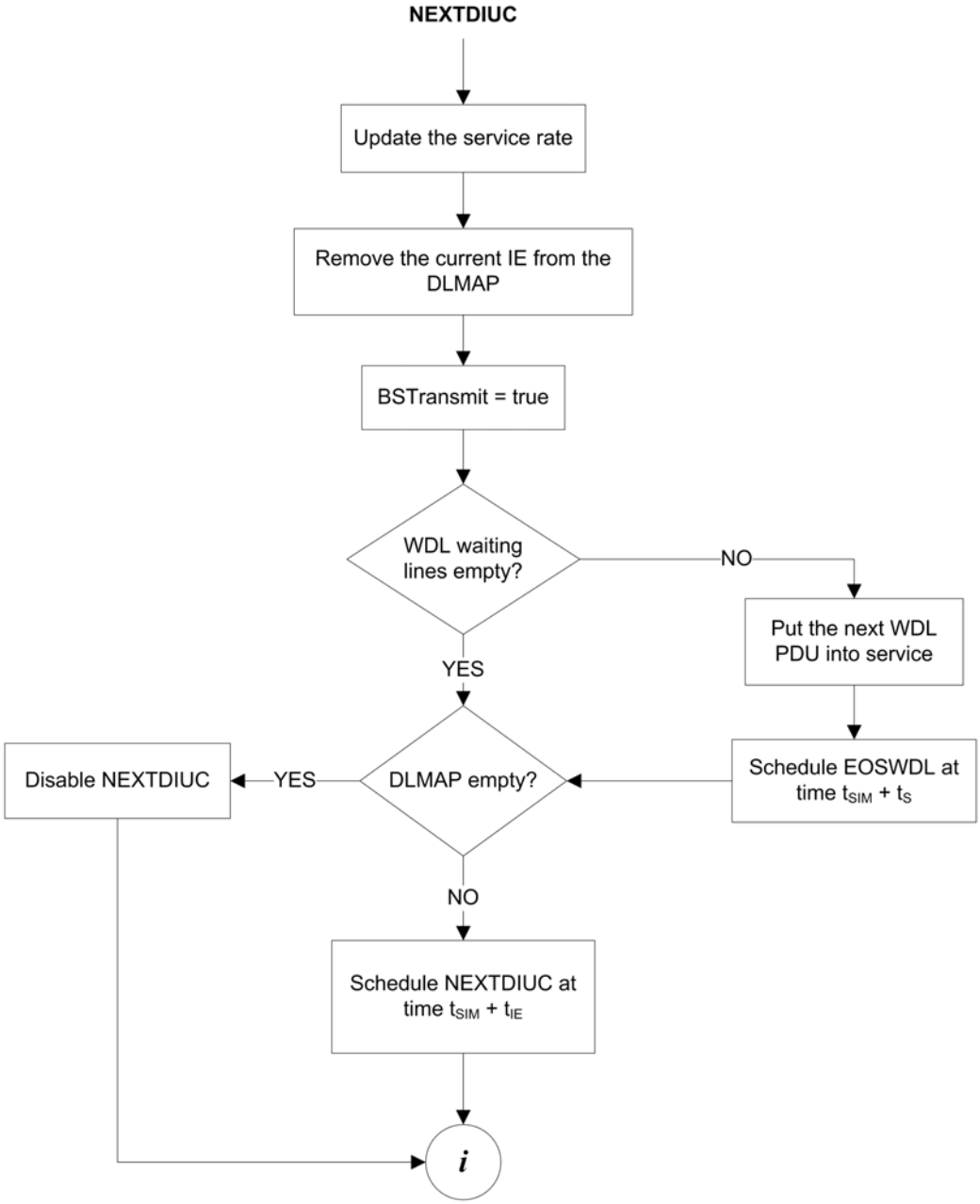

Figure A.7: NEXTDIUC PFC

as described in the PFC of EOULSF shown in Figure A.6, EODLSF and NEXTUIUC (in the case of the ULMAP containing one or more IEs) are scheduled to occur simultaneously. Since the system must first be in the UL antenna state before it may start transmitting along the UL, the EODLSF event routine must always execute first. Handling of the execution order of simultaneous events is discussed in further detail in Section A.1.2. EODLSF and NEXTDIUC are then disabled. The latter is disabled in the case that the MAP generator generated an erroneous DLMAP that spans more of the frame than is allowed by the system specification. Lastly, EOSWDL is disabled since the antenna changed status from DL to UL.

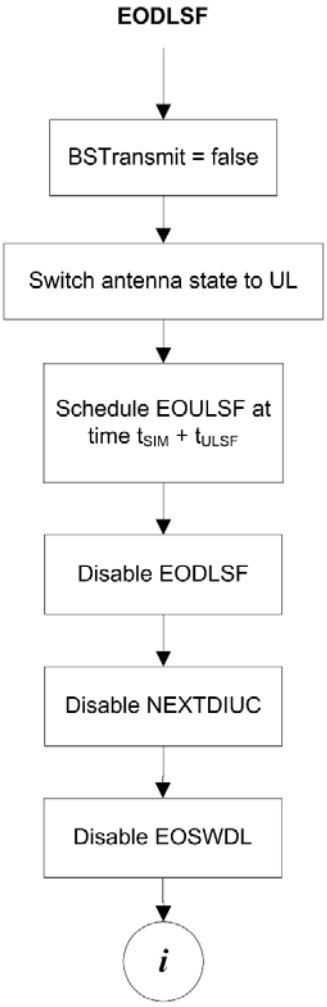

Figure A.8: EODLSF PFC

NEXTUIUC

NEXTUIUC updates the UL transmission profile by effecting the rate and specifying which SS may be transmitting along the UL during the period specified by the next IE in the ULMAP. Figure A.9 shows the PCF for NEXTUIUC. As for EOULSF, the simulator must firstly determine whether the last UL profile period was a connection contention period. If $index == -2$, then it was and the current admitted connection list, used by the BS scheduler to generate the MAPs, must be updated with the delayed connection admission list; a process described in Chapter 4.2. The modulation technique for UL transmission is then set, effectively updating the UL transmission rate for the SS, according to the next ULMAP IE and $index$ is set to the SSID of the SS scheduled for UL transmission. The IE is then removed from the ULMAP.

If the updated $index == -2$, the UL IE specifies a connection contention period. WARR is disabled and, if the ULMAP is empty, NEXTUIUC is disabled. If it is not empty, NEXTUIUC is scheduled at time $t_{SIM} + t_{IE}$.

If the UL IE does not specify a connection contention period, the SS_{index}'s waiting lines are inspected. If these are empty, WARR is disabled. If the waiting lines are not empty, the next PDU is put into service and WARR is scheduled at time $t_{SIM} + t_S$. Lastly, if the ULMAP is empty, NEXTUIUC is disabled, otherwise NEXTUIUC is

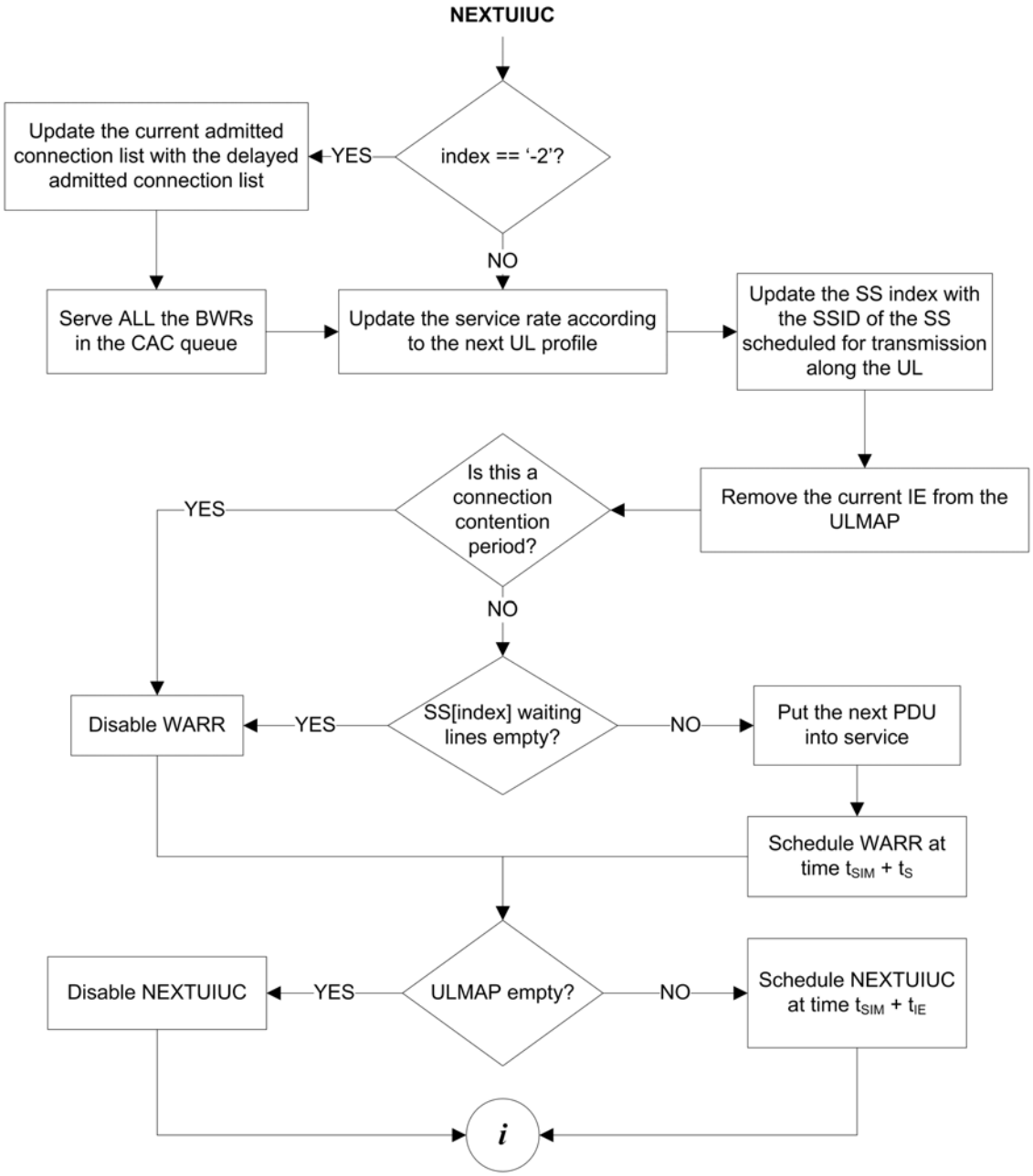

Figure A.9: NEXTUIUC PFC

scheduled at time $t_{SIM} + t_{IE}$.

SSARR

SSARR represents an arrival at an SS in the list of SSs and its PFC is shown in Figure A.10. The SS list of next arrival times is searched and the SS at which the next PDU must arrive is identified (called the *arrival SS*). The PDU is then generated for the arrival SS.

If the arrival SS queues are not empty, no PDU must be put into service. The arrival is only put in the arrival SS's waiting line. On the other hand, if the arrival SS's queues are empty, the arrival is first put in the arrival SS's

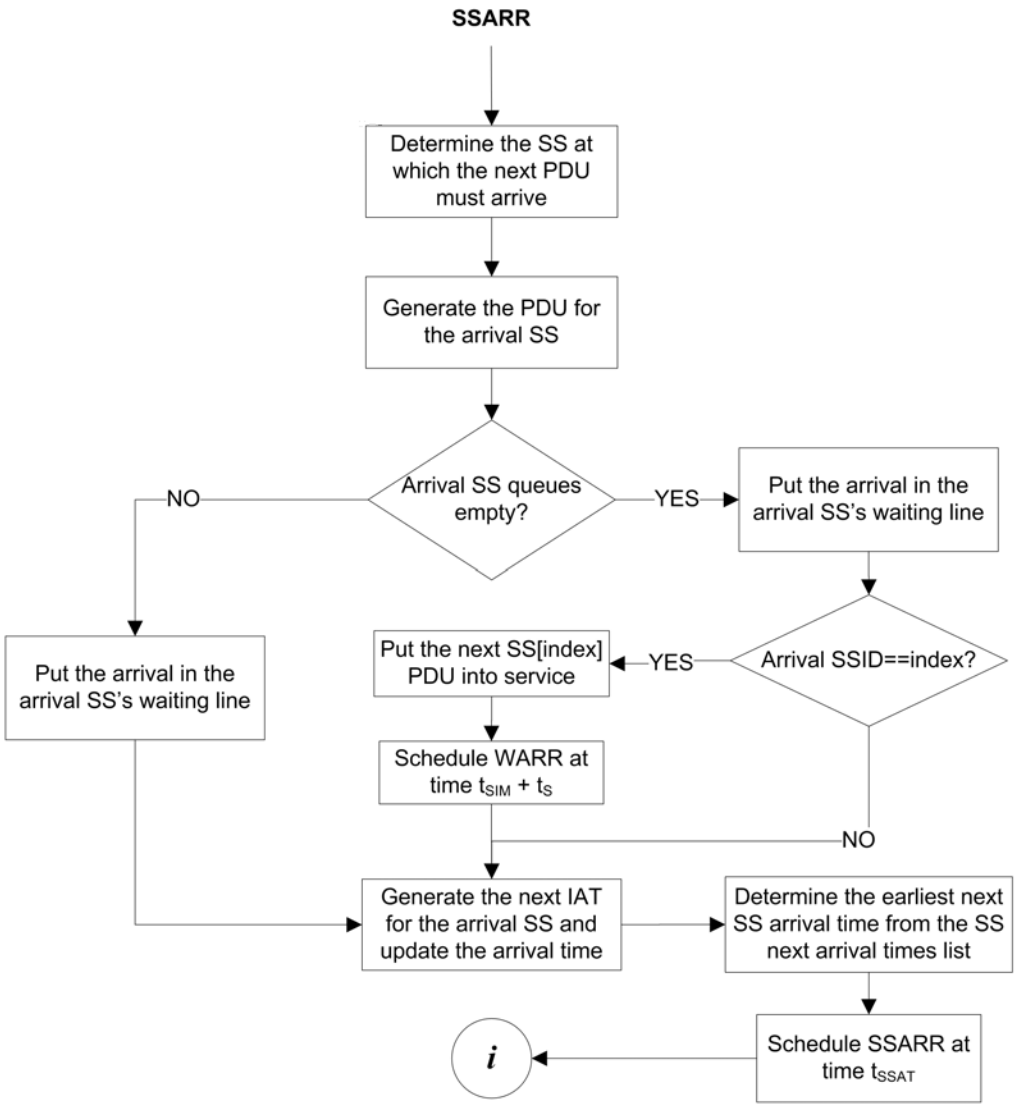

Figure A.10: SSARR PFC

waiting line and then it is determined whether the arrival SS may, at present, transmit along the UL, i.e. if the arrival SS's SSID equals $index$. If the arrival SS may transmit, and since the buffers were empty, the arrival PDU is put into service and WARR is scheduled at time $t_{SIM} + t_S$. Otherwise, if the arrival SS may not transmit on the UL, the PDU is only put in the waiting line.

Lastly, the inter-arrival time is determined for the next arrival at the arrival SS and the SS arrival time is consequently updated. The (earliest) next arrival is then determined from the SS arrival times and SSARR is scheduled at time t_{SSAT} (which already includes t_{SIM}).

FARR

FARR represents a PDU arrival from the internet at the BS wireless DL waiting-line and its PFC is shown in Figure A.11. The INET PDU instance is first generated and, if the wireless DL server queues are not empty,

the arrival PDU is put in the appropriate wireless DL waiting line and the next PDU arrival time at the INET is determined. Otherwise, if the queues are empty, the PDU is put in the appropriate waiting line and it is determined whether the BS may transmit data, i.e. if $BSTrasnmit == true$. If the BS may transmit, the PDU is put into service, EOSWDL is scheduled at time $t_{SIM} + t_S$ and the next PDU arrival time at the INET is determined. Otherwise, if the BS may not transmit, the next PDU arrival time at the INET is determined. Lastly, FARR is scheduled at time t_{FAT}.

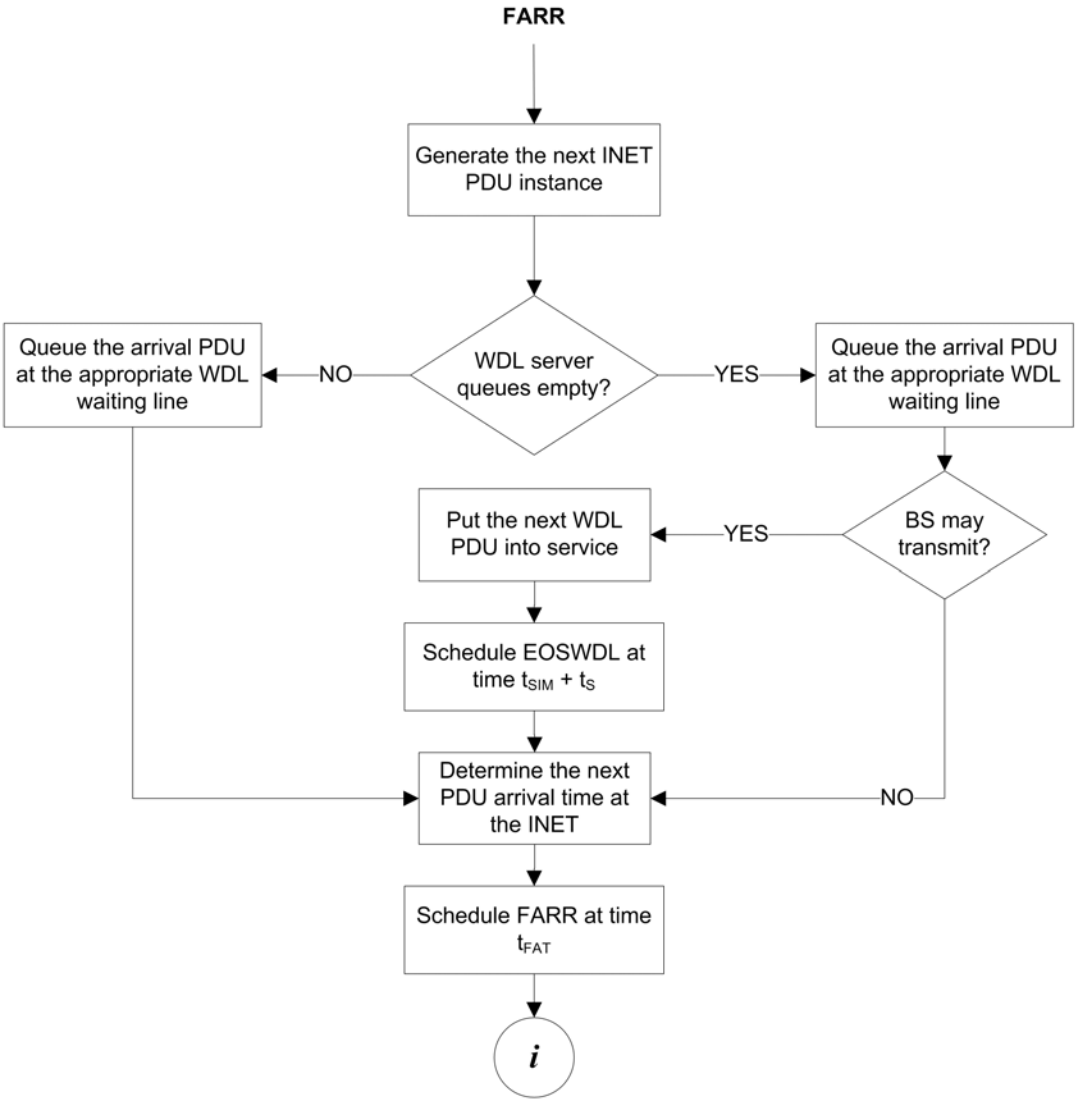

Figure A.11: FARR PFC

WARR

WARR represents the end of service of a currently in-service PDU at the SS that may transmit along the UL and the consequent arrival at either the BS MAC memory buffers or the fixed-line Internet node. The PFC is shown in Figure A.12.

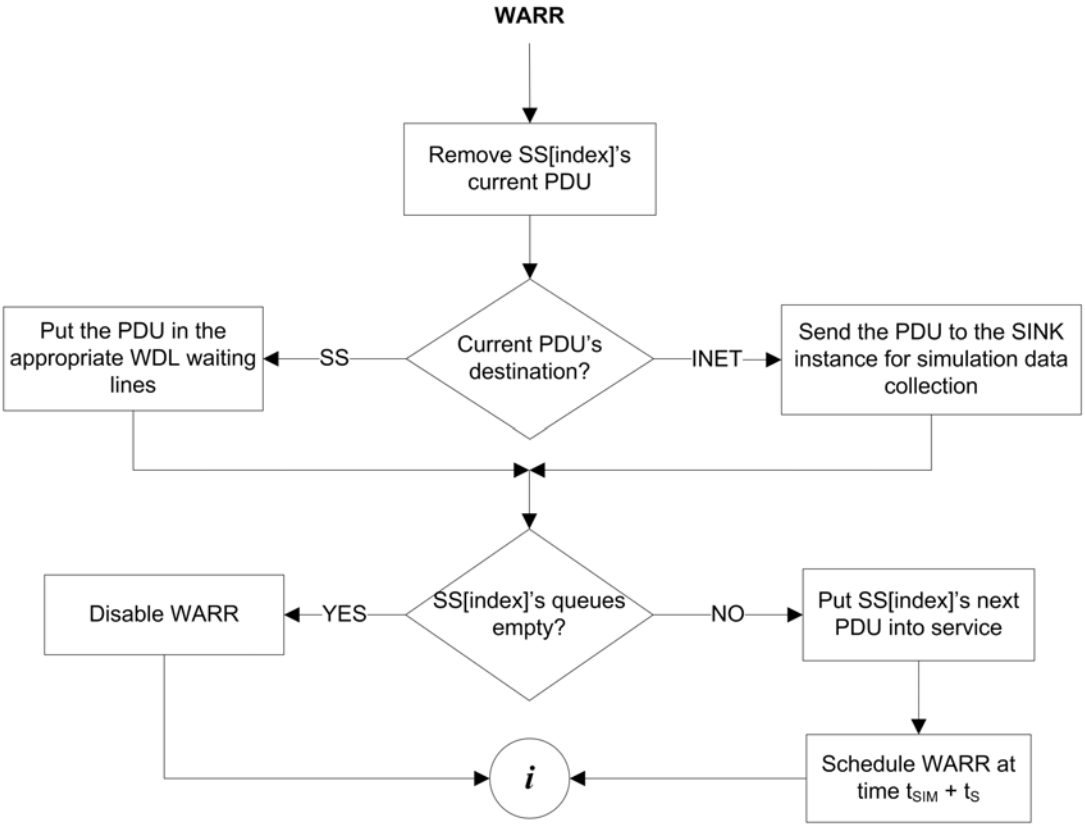

Figure A.12: WARR PFC

The current in-service PDU, at the SS that may transmit, is taken out of service and removed from the SS memory buffers. If the PDU's destination is the internet, it is sent to the data sink for data collection. Otherwise, if it is to be sent to some other SS, the PDU is put in the wireless DL waiting lines.

Next, if the transmitting SS's queues are empty, WARR is disabled. otherwise, the next PDU is put into service at the transmitting SS and WARR is scheduled at time $t_{SIM} + t_S$.

EOSWDL

EOSWDL represents the end of service of a PDU along the wireless DL interface from the BS to an SS and its PFC is shown in Figure A.13. At the end of service, the packet is sent to the data sink which represents successful arrival at the appropriate SS destination and it is removed from the BS memory buffers. If the wireless DL queues are empty, EOSWDL is disabled. Otherwise, the PDU next-in-line in the wireless DL waiting lines is put into service and EOSWDL is scheduled at time $t_{SIM} + t_S$.

BWRARR BWRARR represents the generation of BWR in either the internet or at some SS and the arrivals of these requests at the BS CAC. The PFC is shown in Figure A.14.

First, the next BWR arrival source is determined (being either the INET or some). The appropriate source then generates a BWR which is put into the waiting line at the BS CAC. The service of BWR by the CAC are dealt with by other events already described and the reader is referred to NEXTUIUC and EOULSF. The next BWR inter-

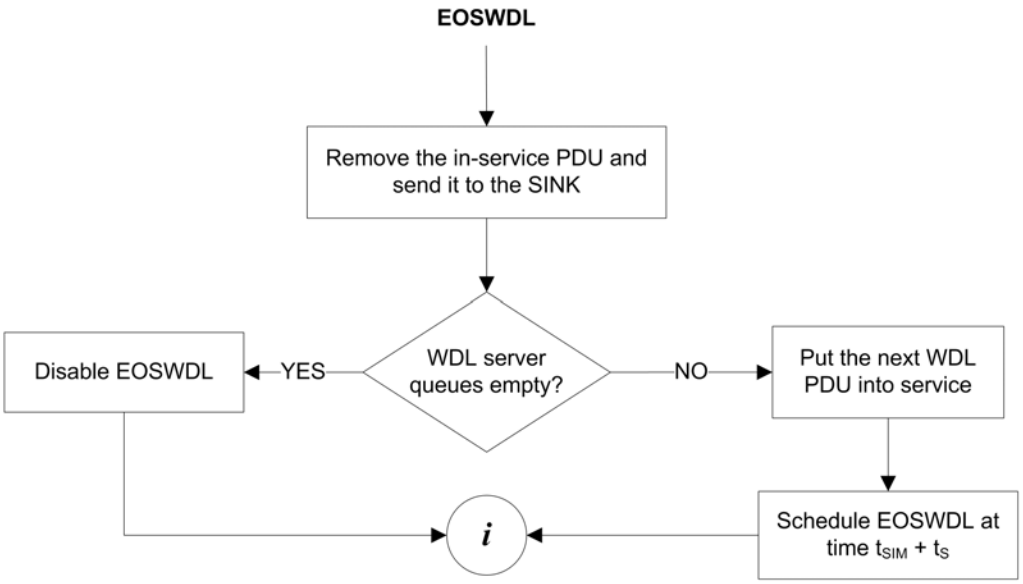

Figure A.13: EOSWDL PFC

arrival time is generated by the corresponding source and the corresponding arrival time is updated accordingly. This list keeps the arrival times of the next BWR at the INET and all SSs. Lastly, BWRARR is scheduled at time t_{BWRAT}, where t_{BWRAT} is the earliest next arrival time in the next BWR arrival times list.

SAMPLE

SAMPLE signals the virtual time at which the simulator must start to record the performance data. A flag is toggled to allow the recording of performance data. Finally, SAMPLE is disabled.

ENDSIM

ENDSIM terminates the simulation run.

Event Order

Simultaneous events that affect each other are managed by considering the structure and search algorithm used to determine the next event to occur in the Calendar. The Calendar is organised as a one-dimensional list, where each event type is allocated a fixed position in this list. The list is searched in a top-down fashion and remembers the event higher up on the list as the next event in the case of simultaneous events. Order matters when the following events occur simultaneously:

- EOULSF and WARR: EOULSF disables WARR and therefore, WARR must be before EOULSF.
- EODLSF and EOSWDL: EODLSF disables EOSWDL and therefore, EOSWDL must be before EODLSF.
- NEXTUIUC and WARR: NEXTUIUC may disable WARR and therefore, WARR must be before NEXTUIUC.

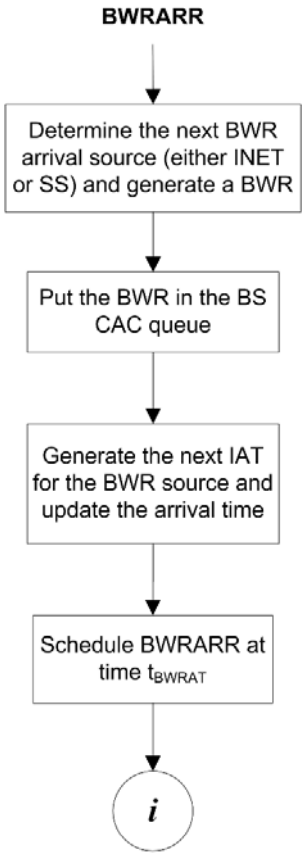

Figure A.14: BWRARR PFC

EOULSF and EODLSF may not occur simultaneously and therefore, their order does not matter. The resulting order of events in the calendar list structure, from top to bottom, is as follows.

1. WARR
2. EOULSF
3. EOSWDL
4. EODLSF
5. NEXTUIUC
6. NEXTDIUC
7. SSARR
8. FARR
9. BWRARR
10. SAMPLE
11. ENDSIM

A.2 IMPLEMENTATION

The design of each system component and the event-driven simulator were translated into Java source code. First the simulation engine was implemented. Thereafter the specific event process flows were implemented. By implementing the process flows and the various system components, the NoQ classes are identified and defined in terms of the necessary attributes (variables) and behaviours (methods). Furthermore, implementing these classes leads to defining the interfaces for the various RaCM and workload model components. Different RaCM and workload components can be implemented using these interfaces specified. UML class diagrams and interface specifications are reported in this section. The Java source code for the various classes are available on request.

For random number generation, the Stochastic Simulation in Java (SSJ) library [35] was used. This is an open-source library that supports random number generation of both uniform and non-uniform random variables.

A.2.1 Simulation Engine Implementation

The main entry point to the simulator is the *BasicSim* class. The additional basic classes created that constitute the simulation engine are the *Scheduler*, *Calendar* and *Clock* classes. The *Event* class is used as a data structure for communicating event information between the *Scheduler* and *Calendar* classes.

The *NoQ* class is the entry point that controls access to the machine and workload components of the system simulated by the simulation scheduler. Figure A.15 shows the simulation engine classes and their relationships. These classes were implemented first, where after the NoQ class was implemented in greater detail, as described in Appendix A.2.2.

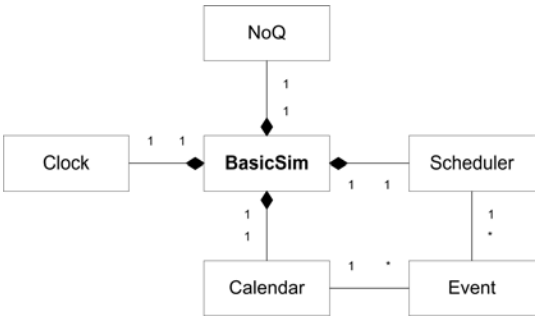

Figure A.15: UML class diagram of the simulation engine-related classes

The *Scheduler* class implements the event routines. These event routines invoke methods that correspond to the PFCs described in Appendix A.1.2.

A.2.2 NoQ Implementation

The NoQ-related classes were created by considering the various NoQ components. The attributes and methods were created as required by methods invoked by the event routines. As shown in Figure A.16, the main components are the Internet, SS, BS and data destination node implemented as the *INET*, *SS*, *BS* and *Sink* classes, respectively.

The *INET* and *SS* classes both inherit from the *Node* class which has a *WLG* class for workload generation. The *SS* class includes the implementation of the SS UL scheduler process. The *WLG* class is detailed later, in

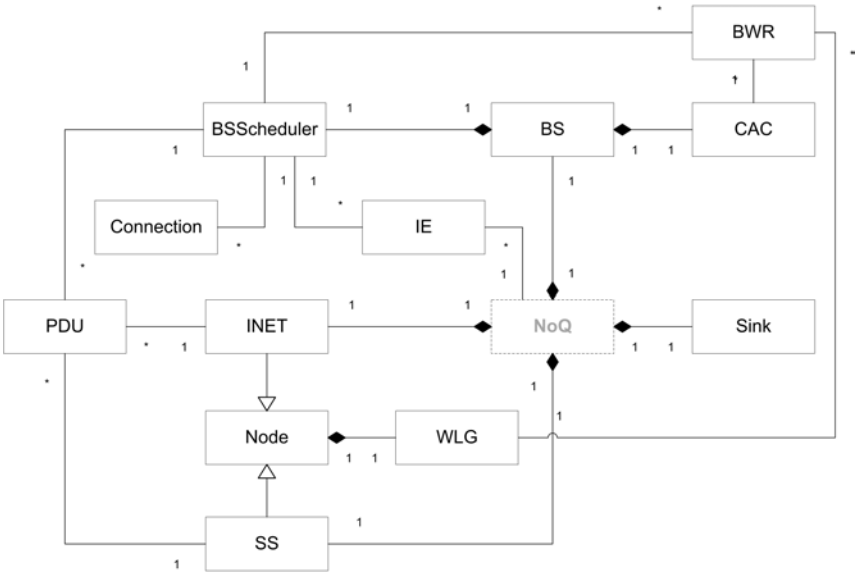

Figure A.16: UML class diagram of the NoQ-related classes

Section 5.2.3.

The *BS* class is composed of the *BSScheduler* and *CAC* classes. The *BSScheduler* class implements the BS UL and DL scheduler processes while the *CAC* class implements the admission control algorithm. Since the BS calculates and generates the MAP information, IEs are communicated between the *BSScheduler* and *NoQ* classes using instances of the *IE* class. Also, the *BSScheduler* class maintains a list of admitted connections, where each connection is represented by the *Connection* class.

The *Sink* class consumes data PDUs that arrive at their destination. These data are written to a PDU trace file which is later used to calculate performance metrics.

BWRs and PDUs are represented by the *BWR* and *PDU* classes, respectively.

A.2.3 Component Interface Specifications

The 4 interfaces specified for the simulator aim to promote modularity in the simulator. Different components can be created by implementing the basic methods specified in these interface classes. The simulator will then invoke these methods in the newly created components. These interfaces are:

Connection Admission Control Interface

The *CAC* class implements the *CACInterface* interface:

public Vector<BWR> serveBWRs(Vector<Object> qosStatus)

Evaluates the BWRs in the request queue and returns a vector containing the successfully admitted BWRs. The QoS status at the BS MAC memory buffers is necessary as input – obtained from the BS scheduler. `qosStatus` is an Object Vector that represents the current network status descriptors.

public void putInWaitingLine(BWR bwr)

Puts the BWR bwr in the CAC waiting ling.

BS Scheduler Interface

BSScheduler class implements the *BSSchedulerInterface* interface:

public Vector<IE> getNextDLMAP()
Determines and returns the DLMAP for the next frame. To determine the size of the MAP, the next ULMAP skeleton/template must also be set up here. Note that the simulator will invoke the *getNextULMAP*() method next, which will detail the IEs of the next ULMAP.

public Vector<IE> getNextULMAP()
Determines and returns the ULMAP for the next frame. Note that *getNextDLMAP*() was invoked just before this method and the ULMAP skeleton were created. This method detail the IEs of the next ULMAP.

public int getMAPSize()
Determines and returns the size of the current MAP in bits.

public void updateActivePool()
Updates the list of connections currently being served by the network with the BWRs initially placed in the delayed BWR list/pool.

public void updateDelayedPool(Vector<BWR> delayedPool)
Updates the delayed BWR pool with the BWRs in the delayedPool Vector.

public int getRate()
Gets the current DL transmission rate of the BS in bits/s.

public void setRate(int rate)
Sets the current DL transmission rate of the BS in bits/s.

public PDU takeOutOfService()
Takes the PDU currently being served out of service. The PDU is returned.

public void putInBuffers(PDU arrival)
Puts the PDU arrival in the appropriate waiting line and at the appropriate position.

public int serveNext()
Puts the next PDU into service and returns the length of the PDU in bits.

public boolean emptyWaitingLines()
Determines whether the waiting lines are empty (true) or not (false) and returns this boolean.

public Vector<Object> getQoSStatus()
Estimates the current network QoS performance and returns a Vector of Objects representation of this. This method must be defined in conjunction the CACInterface interface.

public Vector<IE> getDLMAP()
Gets the current DLMAP as a Vector of IE instances.

public Vector<IE> getULMAP()
Gets the current ULMAP as a Vector of IE instances.

SS Interface

The *SS* class implements the *SSInterface* interface:

public int getSSID()
Gets the SSID of the SS instance.

public void setSSID(int ssid)
Sets the SSID of the SS instance.

public int getRate()
Gets the current UL transmission rate of the SS instance in bits/s.

public void setRate(int rate)
Sets the current UL transmission rate of the SS instance in bits/s.

public PDU takeOutOfService()
Takes the PDU currently being served out of service. The PDU is returned.

public void putInBuffers(PDU arrival)
Puts the PDU arrival in the appropriate waiting line and at the appropriate position.

public int serveNext()
Puts the next PDU in the SS waiting lines into service and returns the length of the PDU in bits.

public boolean emptyWaitingLines()
Determines whether the waiting lines are empty (true) or not (false) and returns this boolean.

public BWR generateBWR()
Generates and returns the next BWR instance.

public PDU generatePDU()
Generates and returns the next PDU instance.

Workload Generation Interface

The *TM* class implements the *WLM* interface:

double getNextBWRIAT()
Determines and returns the time before the next BWR arrival (in seconds).

double getNextPDUIAT()
Determines and returns the time before the next PDU arrival (in seconds).

BWR getNextBWR()
Determines and returns the next BWR.

double getNextPDUSize()
Determines and returns the size of the next PDU (in bits).

int getNextPDUTC()
Determines and returns the TC of the next PDU.

A.2.4 Program Execution and File Formats

The simulator is started by executing the *BasicSim* program. 12 command-line parameters are required at execution time. Table A.2 details these parameters command-line string:

```
java BasicSim test_project 0 2 1 1 5 0 0.001 40000000 80000000 500 1
```

Parameter	Example value	Note
`experiment_name`	*test_project*	The name of the project, where all related files are stored in the *test_project* directory
`debug_mode`	*0*	Mechanism to aid testing (maximum 8 modes)[1]
`num_indep_vals`	*2*	The number of different traffic intensities, i.e. the different values that the independent variable takes on[2]
`min_SSs`	*1*	The number of SSs the network is initially populated with
`increment_SSs`	*1*	The number of SSs to increment the network by for each workload intensity increase
`sim_duration`	*5*	Duration of the simulation (simulation time) in seconds
`start_record`	*0*	Time in the simulation at which to start recording performance data in seconds
`frame_duration`	*0.001*	Frame duration in seconds
`DL_rate`	*40000000*	Basic DL data rate in bits per second
`UL_rate`	*80000000*	basic UL data rate in bits per second
`fragment_size`	*500*	Fragment size in bits
`WorkloadMultiplier`	*1*	Multiplier of packet-level workload IAT streams

Table A.2: Simulation program command-line input parameters

All other system parameters (such as the CAC, BS UL scheduler, etc. parameters) are hard-coded into the simulator program. When developing a RaCM component, the implementer must decide on the component parameter.

On completion of the program execution, trace files are generated for the simulation experiment. Connection-level information is stored in the BWR trace file (with file extension `.bwrtrace`) while packet-level information is stored in the PDU trace file (with file extension `.pdutrace`). In these trace files, comment lines, which are ignored at time of post-processing, start with the hash symbol (#). BWR trace file entries are single-lined entries, each entry consisting of the attributes described by Table A.3. PDU trace file entries are single-lined entries, each entry consisting of the attributes described by Table A.4

A.2.5 Output Data Processing

The BWR and PDU trace files are processed in turn to determine both connection– and packet-level performance statistics. 2 separate trace parsing programs were written in Java to parse and collect statistics from the BWR and PDU traces, namely *ConnTP.java* and *PacketTP.java*, respectively.

[1] This debug facility has been provided so that the programmer can add *debug hooks* into the simulation source when modifying the source in the future.

[2] In the case where the independent variable is the number of SSs in the system, this parameter is the smallest number of SSs in the system experimented with.

	Content	Value
1	Request type	`0=DSA, 1=DSD`
2	TC value	`0=UGS, 1=RTPS, 2=NRTPS, 3=BE`
3	CID	`INTEGER`
4	Source ID	`INTEGER`
5	Destination ID	`INTEGER`
6	Current simulation time	`DOUBLE`
7	Admission result	`true=admitted, false=denied`

Table A.3: BWR trace file entry attributes

	Content	Value
1	CID	`INTEGER`
2	Source ID	`INTEGER`
3	TC value	`0=UGS, 1=RTPS, 2=NRTPS, 3=BE`
4	Length (bits)	`INTEGER`
5	TimeInSSQueue	`DOUBLE`
6	TimeInBSQueue	`DOUBLE`
7	TimeInBSEndService	`DOUBLE`

Table A.4: PDU trace file entry attributes

Connection-level trace processing

ConnTP parses the trace file `sample1.bwrtrace`, generated by the simulator. Both the mean number of connections in the system and mean blocking probability are sampled over successive 5s observation windows per UGS and rtPS TCs. These samples are recorded in the comma separated variable (CSV) files `connParse.csv` and `bpParse.csv`, respectively.

Packet-level trace processing

PacketTP parses the trace file `sample1.pdutrace`, generated by the simulator. The mean throughput is sampled over successive 5s observation windows per UGS, rtPS and BE TCs. The PDU delays are recorded per connection over a cross-section of simulation time, i.e. the comparable time section; the jitter samples are recorded in the same way. The parsing program does not include the delay samples of those connections that have delays greater than a specified maximum user-tolerated delay, after which the user is likely to close the connection; these samples are residue data of connections that were connected via a deadlocked SS and therefore are no longer relevant. The mean delay and jitter are calculated by computing the arithmetic means of the mean delay and jitter experienced per connection in each TC. The mean delay and jitter are recorded in CSV files `delayParse.csv` and `jitterParse.csv`, respectively, while the throughput samples are recorded in the CSV file `tauParse.csv`.

A.3 TESTING

A.3.1 Simulation Engine Tests

Four test cases were devised to test the basic simulation engine:

Test case B1: The correct next event is selected for execution.

Test case B2: The clock is advanced correctly. (This is tested by reviewing the program source. The clock variable must be assigned the time of the Event instance.)

Test case B3: The associated process flows are called correctly. (This is tested by reviewing the program source, ensuring the appropriate method calls are made.)

Test case B4: Simultaneous events are executed in a top-down fashion.

A.3.2 **Simulator Event/Integration Tests**

For the network to operate in the way intended, as described by the design artefacts, it is necessary to ensure that the event PFCs are implemented correctly. Therefore, integration testing involved inspecting the event routines, ensuring they invoke the methods necessary to achieve what was expected. Moreover, these methods must be invoked in the correct order specified by the PFCs. If these methods did not directly affect the intended system state variables, the methods that they invoked in turn, were traced and inspected until all intended variables were affected correctly. The program source was inspected manually and thus no additional test case artefacts resulted for integration testing.

APPENDIX B

TESTING ARTIFACTS

Selected test cases (and their respective debug output) are provided in this appendix for the relevant system components to illustrate how testing was conducted. The other test case artefacts are available on request.

B.1 SCHEDULERS

B.1.1 SS UL and BS DL Schedulers

Field	Content
ID	S1.1
Description	A BE PDU arrival.
Preconditions	System state variables are as after test case S1.13. BE PDU arrives.
Expected results	BE buffer size must increase by 1 PDU.
Outcome	PASS

```
Invoking putInBuffers(BE PDU) on an empty buffer;
  * SS STATUS
  isqp: -1
  ispp: 0
  lastInService: -1
  quotas[UGS]: 6.0
  quotas[RTPS]: 6.999999999999999
  quotas[NRTPS]: 7.199999999999999
  queues[UGS]: 0
  queues[RTPS]: 0
  queues[NRTPS]: 0
  queues[BE]: 1

  * SS STATUS
  isqp: 1
  ispp: 0
  lastInService: -1
  quotas[UGS]: 6.0
  quotas[RTPS]: 6.999999999999999
  quotas[NRTPS]: 7.199999999999999
  queues[UGS]: 1
  queues[RTPS]: 1
  queues[NRTPS]: 0
  queues[BE]: 1
```

B.1.2 BS UL Scheduler

```
Test case S2.2:
DLMAP size (in IEs): 1; ULMAP size (in IEs): 4
MAP size (in bits): 268
```

Field	Content
ID	S2.2
Description	Calculate and generate the DLMAP and thereafter the ULMAP.
Preconditions	S = 3.
Expected results	Since S=3, there must be (as always) 1 DL IE and 4 UL IE(S SS IEs + 1 connection contention IE). Testing for test case S2.1 again, the MAP size should be (120 + 1x20 + 4x32) = 268 bits.
Outcome	PASS

B.2 CAC

Field	Content
ID	C3
Description	Tests whether the CAC can evaluate a DSD request correctly.
Preconditions	A DSD request must be in the request queue. It must have the same ID and TC as an already admitted connection.
Expected results	The request must be admitted and the number of already admitted connections for the Source and TC must decrease by 1.
Outcome	PASS

```
Test C3:
BEFORE: System Variable Status:
Request queue size: 1
Total QoS requests already admitted per Source per TC [SOURCE ID,UGS,RTPS,NRTPS]: [0,0,1,0]
Total QoS requests already admitted per Source per TC [SOURCE ID,UGS,RTPS,NRTPS]: [1,0,0,0]
Total QoS requests already admitted per Source per TC [SOURCE ID,UGS,RTPS,NRTPS]: [2,0,0,0]
Total QoS requests already admitted per Source per TC [SOURCE ID,UGS,RTPS,NRTPS]: [3,1,0,0]
Admitted: 1
AFTER: System Variable Status:
Request queue size: 0
Total QoS requests already admitted per Source per TC [SOURCE ID,UGS,RTPS,NRTPS]: [0,0,1,0]
Total QoS requests already admitted per Source per TC [SOURCE ID,UGS,RTPS,NRTPS]: [1,0,0,0]
Total QoS requests already admitted per Source per TC [SOURCE ID,UGS,RTPS,NRTPS]: [2,0,0,0]
Total QoS requests already admitted per Source per TC [SOURCE ID,UGS,RTPS,NRTPS]: [3,0,0,0]
```

B.3 WORKLOAD GENERATOR

Field	Content
ID	W2
Description	MMAP WLG generates PDUs in the correct volume.
Preconditions	A TM vector containing 1 of each TM must be initialised. The associated transition probability matrix (TPM) must also be initialised.
Expected results	A count of the amount of PDUs generated by a long run of PDU generation must result in the TPM being matched.
Outcome	PASS

```
Volumes specified(TPM):
TC 0: 15%
TC 1: 10%
TC 2: 0%
TC 3: (20+55)=75%
Volumes measured:
TC 0: 0.15%
TC 1: 0.0982%
TC 2: 0.0%
TC 3: 0.7518%
```

Field	Content
ID	B1
Description	The correct next event is selected for execution.
Preconditions	1 event must be scheduled at a time earlier than all others.
Expected results	That event must be identified as the earliest next event.
Outcome	PASS

B.4 SIMULATION ENGINE

```
Make sure all events initialised to INF (Double.MAX_VALUE)
Set NEXTUIUC to 100 [event: 4]
Set WARR to 150 [event: 0]
Set FARR to 20 [event: 7]
Set NEXTDIUC to 100 [event: 5]

Get next event and disable it:
Next Event :7 at time 20.0
```

APPENDIX C

PERFORMANCE DATA FILTERING

This appendix presents the initial transient and rare event activity-region identification. The metric used was the mean number of connections in the system for both UGS and rtPS separately.

C.1 INITIAL TRANSIENT

Two graphs were selected to illustrate identification of the initial transient. These suffice to illustrate the approach undertaken. Figures C.1 and C.2 show the mean number of connections in the system, sampled every 5 seconds, for CAC2-SCHED2 RaCM configuration and a workload intensity of 2.8 Mbps for UGS and rtPS, respectively. The assumed end of the initial transient is indicated on the graphs.

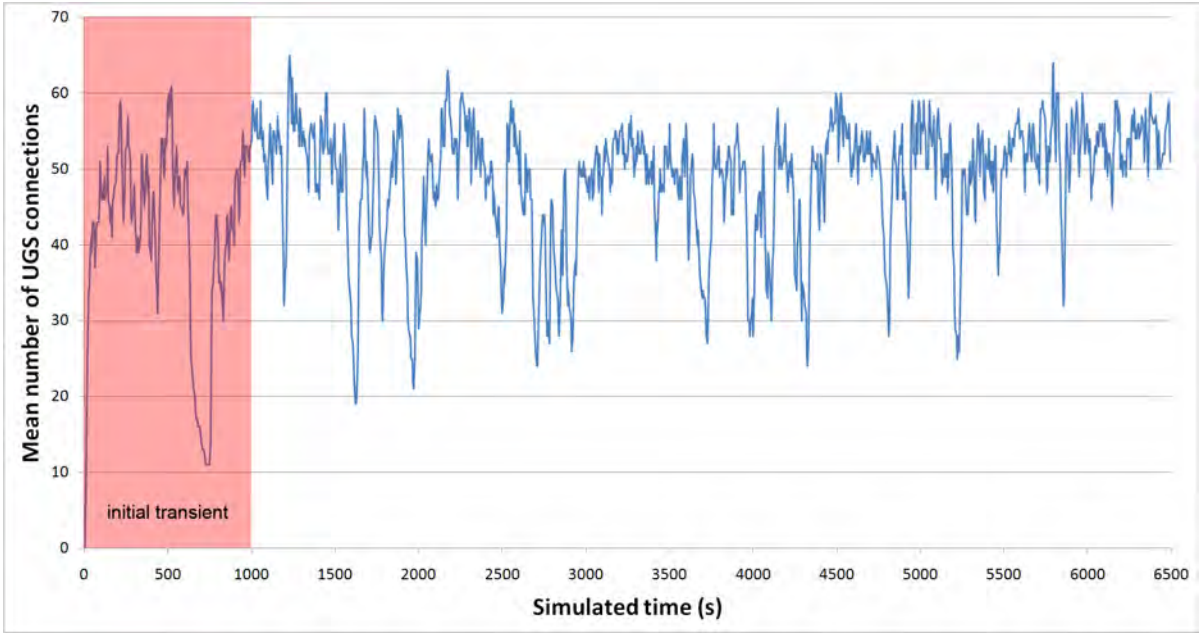

Figure C.1: Mean number of UGS connections in the system versus simulated time, showing the assumed initial transient

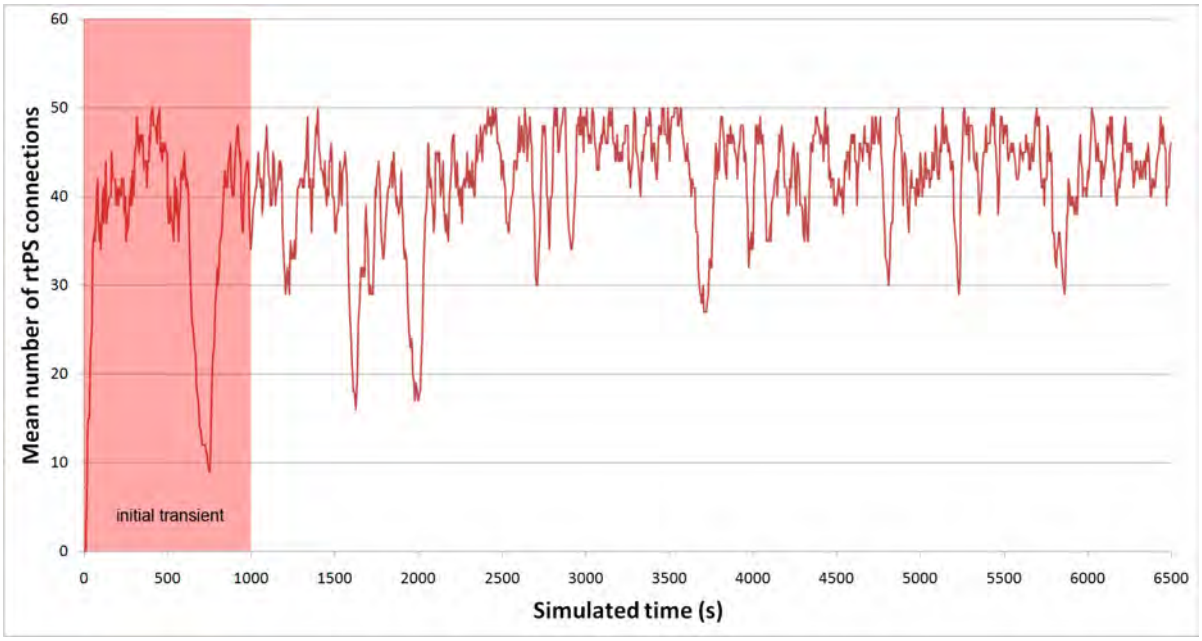

Figure C.2: Mean number of rtPS connections in the system versus simulated time, showing the assumed initial transient

C.2 RARE EVENT ACTIVITY-REGION

Two graphs were selected to illustrate identification of rare event activity-regions. These suffice to illustrate the approach undertaken. Figures C.3 and C.4 show the mean number of connections in the system, sampled every 5 seconds, for CAC2-SCHED1 RaCM configuration and a workload intensity of 3.6 Mbps for UGS and rtPS, respectively. Rare/infrequent event regions are indicated on the graphs.

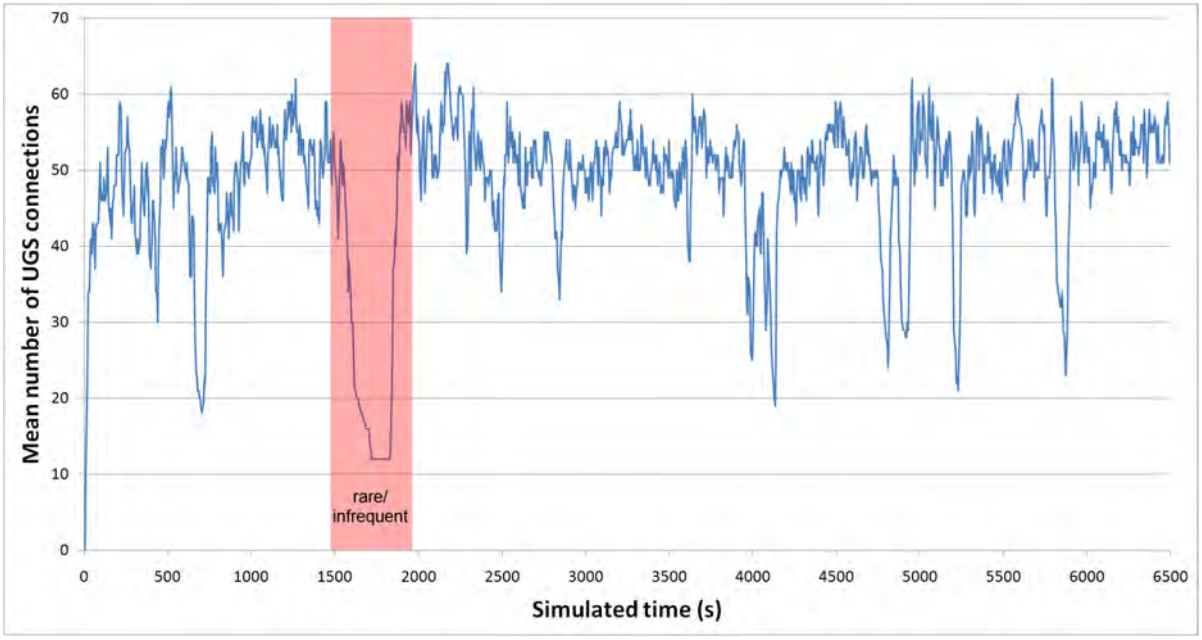

Figure C.3: Mean number of UGS connections in the system versus simulated time, showing the assumed rare event activity-regions

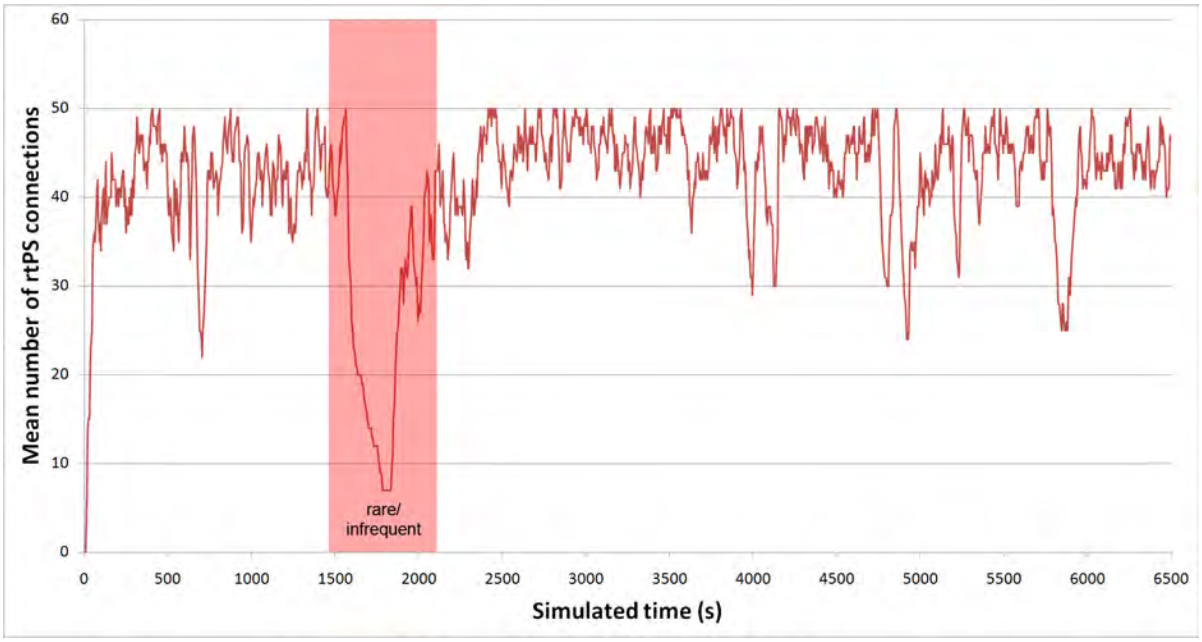

Figure C.4: Mean number of rtPS connections in the system versus simulated time, showing the assumed rare event activity-regions