Design and performance analysis of scheduling algorithms for cable and optical access networks

Universiteit Antwerpen

Faculty of Science
Department of Mathematics and Computer Science

Design and Performance Analysis of Scheduling Algorithms for Cable and Optical Access Networks

Ontwerp en Prestatie Analyse van Scheduling Algoritmes voor Kabel en Optische Access Netwerken

---

Proefschrift tot het bekomen van de graad van
Doctor in de Wetenschappen
aan de Universiteit Antwerpen, te verdedigen door

Dessislava Nikolaeva Nikolova

Promotor: Prof. Dr. Chris Blondia

Antwerp, 2009

Contents

Acknowledgments

First of all, I would like to thank my promoter, Prof. Dr. Chris Blonda, for giving me the opportunity to conduct the extensive research that led to this thesis. I am also very grateful for the freedom I had in the last years to choose research topics, which made me more self assure as a scientist.

I want to pay special thanks to Prof. Dr. Benny Van Houdt for his guidance in my first years as a PhD student, for collaborating with me on several papers that form the basis of Chapter 2 and for the many other remarks and suggestions, which improved the quality of this thesis. Furthermore, I would also like to thank him for the daily serious discussions about gaming strategy, results, life and other important and less important issues. I also want to thank Dr. Gino Peeters, who had to witness all these discussions and was always a very helpful and knowledgeable roommate. My gratitude goes also to Dr. Kathleen Spaey for the careful reading of this thesis and for the various useful suggestions for improvements.

I want to thank all my colleagues from the third floor and the PATS group for the pleasant and relaxed atmosphere. I am especially grateful to Juan F. Pérez, Jeroen Avonts and Dr. Jeroen Van Velthoven, who made it feel a little bit like home by learning few Bulgarian words and showing genuine interest in tasting exotic beverages. I would also like to acknowledge the software and hardware support I received from Dr. Peter De Cleyn, Dr. Nik Van den Wijnagaert, Nico Letor, Johan Bergs and Bart Braem.

Finally, my thanks and gratitude go to my husband, Pascal, for the fun times, for his support, care and patience during all these years.

Summary and Organization

Wireline point-to-multipoint access networks connect a telecom provider via a shared coax and/or optical link with its subscribers. Such networks rely on traffic scheduling algorithms to efficiently utilize the available bandwidth and to provide quality-of-service (QoS) guarantees, which makes such algorithms an important design issue. In the upstream, often the users first have to request bandwidth before being granted any. It is the task of a scheduling algorithm to allocate bandwidth to the subscribers and their packet flows. In the downstream a scheduling algorithm would select the packet to be transmitted next on the shared link. The challenges in the design of these algorithms are ensuring fairness, flow isolation and low complexity in the downstream and efficient bandwidth utilization in the upstream. This thesis presents and analyses different algorithms designed for upstream and downstream scheduling in high-speed cable and optical subscriber access networks.

Chapter 1 provides an introduction to the subject and identifies the requirements for the design of scheduling algorithms. It gives an overview of the considered subscriber access technologies. The optical access network studied is the Ethernet Passive Optical Network. For Hybrid Fiber Coax Networks (HFC) it is the technology standardized in the Data-Over-Cable Service Interface Specifications (DOCSIS) versions 2.0 and 3.0. The chapter provides also a review of existing packet scheduling algorithms and discusses methods for their evaluation. Such methods are software simulations and theoretical analysis using the Latency Rate concept. The latency gives a measure to compare different scheduling disciplines and provides a bound on the maximum packet delay of a leaky-bucket shaped traffic.

An Ethernet Passive Optical Network (EPON) is a subscriber access network technology offering high-speed access over optical fiber. An EPON has tree topology with an Optical Line Terminal (OLT) connected at the trunk and Optical Network Units (ONU) residing at the subscribers premises. The upstream bandwidth allocation algorithm is responsible for assigning the upstream channel bandwidth to the traffic of the different

users. The design choices for the algorithm determine the fairness and efficiency of the network, the different QoS parameters like the packet delay, delay variation and bandwidth guarantees. While several upstream scheduling algorithms for EPON have been proposed in the literature, *Chapter 2* introduces the novel threshold reporting mechanism in the scheduling. For the purpose, methods to generate reports with thresholds at the ONU and to process them at the OLT, in a way suitable to base the scheduling decision on these reports, are designed. An upstream scheduling algorithm for EPON is defined by the scheduling mechanisms in both the ONU and the OLT. Two types of scheduling at the ONU are considered, namely full priority scheduling, which is the common strict priority scheduling and interval priority scheduling, where higher priority traffic is transmitted before lower priority only if it was already reported to the OLT. The types of algorithms proposed for the OLT differ on the threshold level on which they base their scheduling algorithm - full and single report threshold level - and on their scheduling policy for constant bit-rate traffic . The algorithms are analyzed by means of a detailed simulation program, regarding average packet delay, delay variation and bandwidth utilization. It is shown that by using threshold reporting and interval priority scheduling at the ONU the bandwidth can be fully utilized. Combining it with rate based scheduling at the OLT provides an interesting tradeoff between the efficiency, which is still near to the optimal, and the delay characteristics of time critical applications. The algorithms and results were published in (4), (5), (6) and (8) and have received more than 20 citations.

One of the currently most deployed wireline broadband access network solution is the HFC network. It uses the legacy community antenna television cables and the DOCSIS family of specifications, which standardize the physical and the Medium Access Control (MAC) layers and also the QoS mechanisms. The DOCSIS 2.0 specification offers a wealth of possibilities for the cable operators to provide high speed quality access to their subscriber. It is thus important to have a real-system simulator which allows to study the influence of the different standardized system parameters and scheduling types. *Chapter 3* describes a simulator of the DOCSIS 2.0 based HFC networks implemented in C++ using the open-source network simulator package OMNET++. Unlike other free DOCSIS simulators, it models in detail the physical medium dependent layer and many features of the Medium Access Control (MAC) layer. Various QoS mechanisms from the DOCSIS MAC protocol are implemented. These include the upstream scheduling services: unsolicited grant service, real time polling service (rtPS) and best effort (BE) service. The different polling mechanisms implemented are unicast request, bandwidth request in contention slots and piggybacking. The simulator has been extensively val-

idated and some simple scenarios, which constitute good examples of the operation of the DOCSIS protocol and of the implemented model, are presented in this chapter. The performance of the rtPS and BE scheduling services is evaluated via simulations and the results are described in the chapter. Improvement to the scheduling is proposed and optimal values for some contention channel and MAC protocol parameters are obtained.

The simulator can be used to evaluate the performance of a DOCSIS system in different scenarios, to fine tune the values of some parameters and to evaluate different scheduling algorithms, which is of great use for cable operators. It has already been used to simulate and evaluate the performance of the actual scheduler implemented in Motorola BSR 64000 CMTS/Edge routers, which are commercially deployed by a cable operator in Belgium. The results are reported in (7) but as the study remains confidential they are not disclosed here.

In subscriber access networks, typically high-speed routers are responsible for the downstream scheduling. The major requirements for downstream packet scheduling algorithms in high-speed networks are the low complexity and guaranteeing flow isolation, fairness, low end-to-end delay and bandwidth utilization. In *Chapter 4* a novel scheduling discipline called *Last Backlogged First Served* is proposed. It is combined with the Deficit Round Robin (DRR) algorithm resulting in a LBFS-DRR scheduler. In comparison with DRR, the new scheduler provides lower average packet delay, while preserving the advantageous features like O(1) complexity, fairness and bandwidth guarantees. The lower mean delay is realized by giving service in a round first to flows transmitting bellow their (weighted) fair share. The algorithm exploits the high variability in the typical user traffic pattern resulting in lower mean file transfer delay. The implementation for the known *Surplus Round Robin* algorithm proposed in this thesis has also O(1) complexity and succeeds in guaranteeing fairness and delay bounds for leaky-bucket shaped traffic. The fairness and latency bounds of the algorithms are derived analytically. Both algorithms are implemented in software and further analyzed and compared by simulations. Some of the results on packet scheduling algorithms are reported in (2) and (3). The second paper was awarded *Best paper for Telecommunication System* at the conference where it was presented.

In order to increase the available capacity some systems aggregate several channels. Channel bonding or the bonding of several channels together to create a larger bandwidth pipe, is defined in DOCSIS 3.0 for HFC access networks. This new technology offers new challenges for the design and analysis of the scheduling algorithms. To tackle the downstream scheduling problem two algorithms with distributed architecture are designed, which are described in *Chapter 5* . They make use of two different queueing

mechanisms. The Bonded Deficit Round Robin (BDRR) uses input queuing and can fully utilize the available downstream bandwidth. It is shown that the BDRR scheduler is a Latency-Rate scheduler and that it results in a bounded packet reordering. Thus it can bound the maximum delay for leaky bucket shaped traffic, which is derived theoretically. This is not the case for the output queueing algorithm - OutQ-DRR. Besides that it does not bound the delay, it is also unable to fully utilize the available bandwidth. However, it is more straightforward to apply as the channels are served independently from each other. The performance of both algorithms is further analyzed via simulations for a variety of traffic and load scenarios. Parts of the study on channel bonding for HFC networks is reported in (1).

Chapter 6 concludes the thesis and discusses future research directions.

While Chapters 2 and 3 describe simulation studies of the designed algorithms, Chapters 4 and 5 provide also theoretical analysis along with the results from the simulations. The most novel results on upstream scheduling are in Chapter 2 and on downstream in Chapters 4 and 5.

List of Publications

(1) Bonded Deficit Round Robin: packet scheduling algorithms for networks with channel bonding, D. Nikolova and C. Blondia; IEEE Transaction on Parallel and Distributed Systems, revised version submitted, 2009

(2) Last-Backlogged First-Served Deficit Round Robin (LBFS-DRR) Packet Scheduling Algorithm , D. Nikolova & C. Blondia in Proceedings of the International Conference on Networks (ICON 2007), Adelaide, Australia, 2007

(3) Evaluation of Surplus Round Robin Scheduling Algorithm, *(Best Paper Award for Telecommunication Systems)*, D. Nikolova & C. Blondia, in Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 06), Calgary, Alberta, Canada, 2006.

(4) Dynamic Bandwidth Allocation for Ethernet Passive Optical Networks (EPON) with Threshold Reporting, D. Nikolova, B. Van Houdt & C. Blondia, Telecommunication Systems, Vol. 28, No. 1, pp. 31-52, 2005.

(5) Dynamic Bandwidth Allocation Algorithms in EPON: a Simulation Study, D. Nikolova, B. Van Houdt & C. Blondia, Proceedings of Optical Networking and Communications (OptiComm 2003), Dallas, USA, 2003.

(6) Quality-of-Service Issues in Ethernet Passive Optical Networks *(Invited Paper)*, D. Nikolova, B. Van Houdt & C. Blondia, Proceedings of Community Nets and FTTH/P/x Workshop, Dallas, USA, 2003.

Technical Reports

(7) Study of Hybrid Fiber Coax Networks - Technical report UA, 2004

(8) Ethernet Passive Optical Networks - Technical report, UA, 2003

List of Abbreviations

BDRR	Bonded DRR
BE	Best Effort
BG	Bonding Group
CBR	Constant Bit Rate
CM	Cable Modem
CMTS	Cable Modem Termination System
DOCSIS	Data-Over-Cable Service Interface Specifications
DRR	Deficit Round Robin
DS	Downstream
EPON	Ethernet Passive Optical Network
FP	Full Priority (strict priority)
FTL	Full Threshold Level
Gb/s	Gigabits per second
HFC	Hybrid Fiber Coax
IP	Interval Priority
IPG	Inter-packet Gap
Kb/s	Kilobits per second
LBFS-DRR	Last Backlogged First Served DRR
LR	Latency-Rate
MAC	Medium Access Control
MAP	Bandwidth Allocation Map
Mb/s	Megabits per second
MPCP	Multi-Point Control Protocol
nrtPS	non-real time Polling Service
OLT	Optical Line Terminal
ONU	Optical Network Unit
OutQ-DRR	Output Queuing - DRR
PON	Passive Optical Network
QoS	Quality-of-Service
rtPS	real time Polling Service

SF	Service Flow
SRR	Surplus Round Robin
SRTL	Single Report Threshold Level
TW	Transmission Window
US	Upstream
USG	Unsolicited Grants Service
WF^2Q (WF2Q)	Worst-case Fair Weighted Fair Queuing

Chapter 1

Introduction

Subscriber access networks provide telecommunication services using a variety of broadband, packet-based, Quality-of-Service (QoS)-enabled technologies. These technologies differ in their topology, physical and link layer. Wireline networks use optical, twisted pairs, power line or coaxial cables to provide subscriber access. The optical and coaxial ones can have point-to-multipoint topology. This makes them an attractive choice for subscriber access because it allows part of the equipment and infrastructure to be shared amongst many users. Figure 1.1 shows a typical architecture of a point-to-multipoint access network, where subscribers are connected via a central office to the Internet. Such a broadband network supports diverse traffic as data, video, gaming, voice.

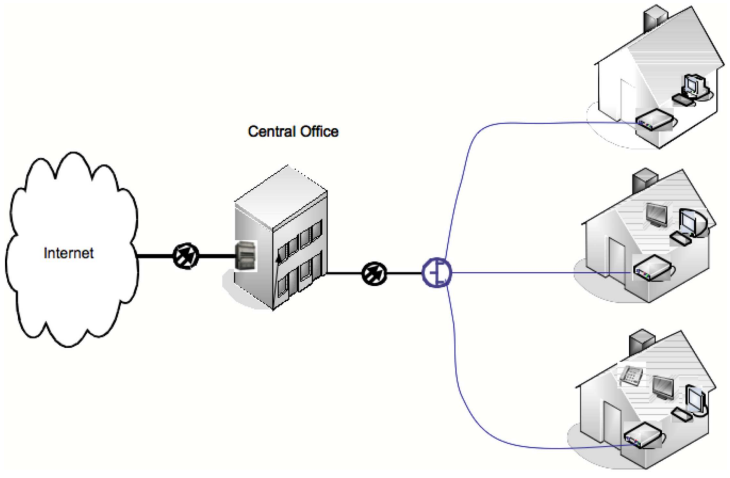


Figure 1.1: Point-to-Multipoint Fixed Subscriber Access Network

Currently a widespread, fixed subscriber access network technology with point-to-multipoint topology is the hybrid fiber coax (HFC) access network. There is a family of

Data-Over-Cable Service Interface Specifications (DOCSIS) [1], [2], [3], [4] which standardizes the physical, the Medium Access Control (MAC) layers and also, in the later versions, the QoS support. For example, the DOCSIS 2.0 standard offers a maximum downstream speed (from the central office to the subscriber homes) of 40Mb/s shared amongst all users, the number of which is typically in the range 300 to 1000 different subscribers. The increasing demand for more throughput has lead to the latest version of the DOCSIS, namely 3.0 [5], where a new technology is introduced. The physical link carries in one direction not one but several channels with different frequencies. The DOCSIS 3.0 technology bonds on MAC level the channels in order to obtain a larger bandwidth pipe. The bandwidth increases linearly with the number of channels bonded, thus for 4 channels it offers a downstream capacity of 160 Mb/s.

The passive optical network (PON) is an emerging subscriber access network technology that provides high bandwidth capacity over optical cable. Ethernet PON (EPON) [6] has also point-to-multipoint topology and uses the widespread Ethernet protocol [7]. An EPON supports a nominal bit rate of 1Gb/s, shared amongst maximum of 32 subscribers, which can be at a maximum distance of 20 km.

Both EPON and HFC networks have Ethernet as the Data Link layer but have a different MAC sub-layer. The MAC sub-layer is the interface between the physical and the data link layers and provides addressing and channel access control mechanisms that make it possible for the network nodes to communicate within such a multipoint network. The mechanism within the MAC sub-layer, which arbitrates the transmissions to and from the users, is referred to as a traffic scheduling algorithm.

The HFC and PONs serve different types of customers ranging from individual homes to corporate campuses and curbs. The traffic on the network can pertain to thousands of different applications and/or users. Applications may pose largely different requirements to the network as to the type of service they require. For example file transfers may require a minimum transfer rate while voice and video would require other quality of service (QoS) parameters like a guaranteed allocation of network resources and predictable bounds on end-to-end delay and/or delay variation. Other applications are assumed to be best-effort, in the sense that they would just try to use a fair share of the network resources. The applications may transmit the data in bursts which ultimately may cause congestion on the links. Users may also misbehave or be greedy trying to use more than a fair share of the network resource or more than what they have reserved.

In DOCSIS the QoS is supported by the mapping of the traffic to service flows, which can be from five different standards scheduling service types. The QoS features, defined in the standard MAC protocol, largely define the transmission ordering and scheduling in

the upstream, although they often have to work in conjunction with mechanisms beyond the standardized.

EPON relies on the inherent Ethernet QoS [8], expressed in the 8 possible different packet priorities and smart implementation dependent scheduling mechanisms to provide QoS. Also in the downstream for both point-to-multipoint networks, the scheduling algorithm is implementation dependent and is not standardized. Thus it is a very important design issue, which makes the difference between the different manufacturers equipment.

The major requirements for a packet scheduler applicable for high -speed networks are:

- Low complexity:
  The speed of the networks is progressively increasing. Such high-speeds require that the packet scheduling decision is made in less than a nanosecond. Thus low and constant time complexity and simplicity of the implementation is a must for the algorithm. An algorithm is said to have a constant or O(1) complexity if the worst-case number of operations needed to select the next packet is constant with respect to the number of flows.

- Isolation of flows:
  The algorithm must isolate an application session from the undesirable effects of other (possibly misbehaving) flows. That is, the algorithm should be able to ensure that a misbehaving flow does not affect other well-behaived ones.

- Low end-to-end delays:
  The end-to-end delay is the most assessed measure for the user experience. Moreover in a work-conserving system low packet delays imply low buffer requirements. Thus, the choice of the scheduler may directly affect the cost of the implementation in terms of the required memory. A bounded delay would allow to dimension the buffer requirements for a specified QoS.

- Fairness:
  The available bandwidth must be divided amongst the users in a fair manner. An unfair algorithm might offer widely different service rates to connections with the same reserved rate and service level agreement.

- Utilization:
  The algorithm should utilize the link(s) bandwidth(s) efficiently and attempt to

achieve the maximum link throughput. This requirement is vital for subscriber access networks which have lower capacity than the core and metropolitan networks.

Scheduling algorithms have been a topic of intense research throughout the years and there is a significant number proposed in the literature. Not all of them however satisfy all the requirements. Usually there are trade-offs between the complexity, delay, fairness and achieved utilization. During the years also the weight of the different requirements has varied. Currently for schedulers applicable for high-speed networks the low complexity is the determining factor. For upstream scheduling algorithms typically the complexity is not an issue but the utilization and delay guarantees are the dominating factors. Moreover the different subscriber access network technologies impose different challenges on the design of the scheduling algorithms. They operate according to various MAC protocols, which have to be incorporated in the algorithm. To evaluate the performance of the protocols and algorithms analytical methods from the queuing theory and the latency-rate server concept are used. However due to the complexity of the task very often also simulations are used for performance evaluation.

This dissertation addresses several of the issues involved in the design and analysis of scheduling algorithms for the fixed point-to-multipoint packet subscriber access networks. Algorithms for down- and up-stream scheduling in EPON and in DOCSIS 2.0 and DOCSIS 3.0 based HFC access networks are considered. For EPON, the novel threshold reporting is introduced in the upstream scheduling. In combination with rate-based scheduling it significantly improves several performance characteristics. For DOCSIS 2.0 based HFC networks the performance of several scheduling algorithms and of several standard scheduling service types is evaluated. For the purpose a very detailed simulation tool is implemented in OMNET++. The same tool is used to simulate also the packet scheduling algorithms proposed for downstream scheduling. In the downstream direction typically a high-speed router is responsible for the scheduling. Therefore two packet scheduling algorithms with O(1) complexity are implemented and simulated. It is shown by theoretical analysis that they are fair and have bounded delay. Finally, channel bonding systems are considered. This emerging technology, defined in DOCSIS 3.0 has not yet been addressed. Two scheduling algorithms differing by the packet queuing they use are proposed and evaluated.

Next in the introduction the different point-to-multipoint subscriber access technologies are described with emphasis on their MAC protocol and already available upstream scheduling mechanisms are outlined. A detailed overview of the existing packet scheduling algorithms applicable for downstream scheduling, is given afterwards, followed by an

outline of the different methods for their analysis. Before providing the thesis overview the contributions of the dissertation are summarized.

1.1 High-Speed Point-to-Multipoint Subscriber Access Networks Overview

1.1.1 DOCSIS 2.0

One of the currently most deployed wired broadband access network solution is the hybrid fibre coax (HFC) network. It uses the legacy community antenna television (CATV) cables and is a point-to-multipoint network with tree topology. The logical topology of the network is visualized in Figure 1.2. The terminal equipment placed at the root (head-end) of a HFC network is referred to as a Cable Modem Termination System (CMTS). It is connected via coaxial or both optical and coaxial cable to Cable Modems (CM) situated at the customer premises. There is a family of Data-Over-Cable Service Interface Specifications (DOCSIS) which standardize the physical and the Medium Access Control (MAC) layers and also in the later two versions the Quality-of-Service (QoS) support [1], [2], [3]. There is a EuroDOCSIS protocol [4], which is applied in Europe and diverse from the DOCSIS only at the physical layer. Further on in the thesis only the term DOCSIS will be used.

The medium between the CMTS and the different CMs is a two-way shared medium, in which the downstream channel carries signals from the head-end to users and upstream channels carry signals from users to head-end. A user or a subscriber further on refers to a single CM. Upstream and downstream channels are separated using Frequency Division Duplex (FDD). A CM is normally tuned to one upstream channel and one downstream channel. The upstream channel is an inherently shared medium while the downstream is a broadcast dedicated link from the CMTS to the CMs. The capacity of the channels depends on the modulation used on the physical layer. It determines how many symbols are necessary to transmit one byte. For example when 256 Quadrature Amplitude Modulation (QAM) is used in the downstream the channel capacity is 40Mb/s. Details on the DOCSIS physical layer including the forward error correction overhead are further given in Section 3.

The upstream channel capacity is a lot smaller and at 64 QAM is $\approx 5120Kb/s$. As a consequence the major features of the upstream scheduling mechanisms are standardized as part of the MAC layer. DOCSIS 2.0 specifies also the QoS mechanisms and algorithms at the MAC layer. To support QoS the notion of Service Flows (SF) is introduced. Each

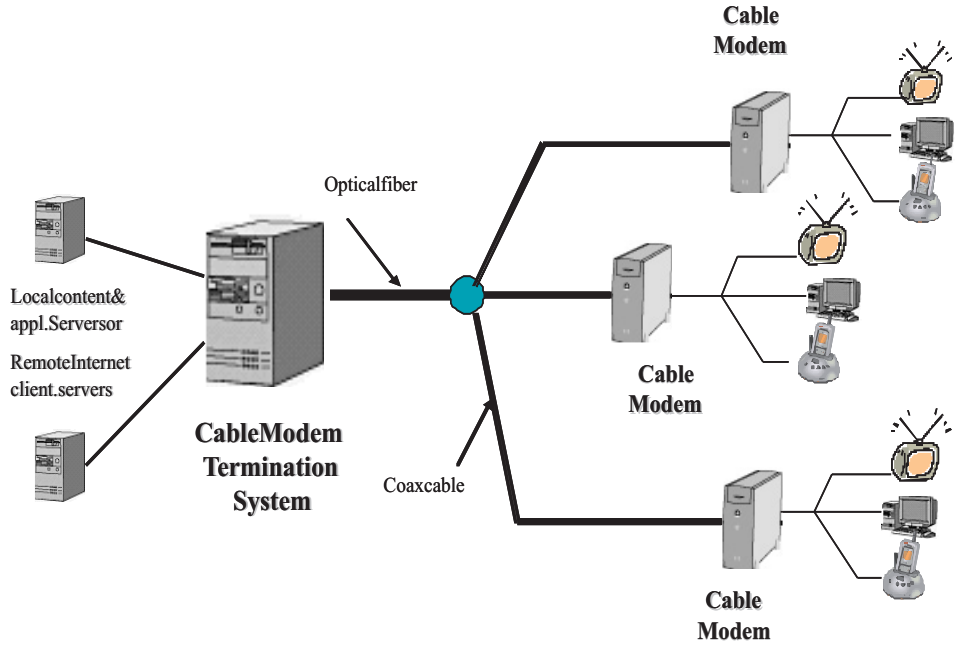


Figure 1.2: The logical topology of a HFC network

SF is unique and defines a particular service class or flow mapping between a CM and the CMTS. The CMTS may assign one or more SFs to a particular CM. A CM has at least two service flows one for the upstream and one for the downstream. Each SF is assigned a unique ID and the CM knows the IDs of its service flows. In this way it can filter out the downstream packets destined to any of its service flows.

The scheduler at the CMTS arbitrates the transmission opportunities of the SFs on the upstream channel. It transmits periodic messages called Bandwidth Allocation Map (MAP) where it informs the SFs in which period they can send data or requests on the upstream channel. The SF can send requests via a contention transmission opportunity, via a unicast request opportunity or using piggybacking. The scheduler allocates bandwidth based on requests or schedules unsolicited transmission opportunities or assigns portions of the bandwidth for contention transmit opportunities. The details of the mechanism are described in Section 3.1.2.

An upstream SF may be one of the following

- Unsolicited Grant Service (UGS)
  The UGS scheduling service type is used to support real-time SFs that generate fixed size data packets on a periodic basis such as VoIP. The service offers fixed

size unsolicited grants on a real-time periodic basis, which eliminates the overhead and latency of SF requests and assures that grants will be available to meet the flow's real-time needs.

- Real-time Polling Service (rtPS)
  The rtPS is designed to support real-time service flows that generate variable size data packets on a periodic basis such as MPEG video. The service offers real-time, periodic, unicast request opportunities, which meet the flow's real-time needs and allow the SF to specify the size of the desired grant. The service requires more overhead than the UGS but supports variable grant sizes for optimal data transport efficiency.

- Unsolicited Grant Service with Activity Detection (UGS/AD)
  The UGS/AD is designed to support UGS flows that may become inactive for substantial portions of time, such as VoIP with silence suppression. The service provides Unsolicited Grants when the flow is active and unicast polls when the flow is inactive. This combines the low overhead and low latency of UGS with the efficiency of rtPS. Though UGS/AD combines UGS and rtPS, only one of these scheduling services should be active at a time.

- Non-Real-time Polling Service (nrtPS)
  The nrtPS is designed to support non real-time service flows that require variable size data grants on a regular basis, such as high bandwidth FTP. The service offers unicast polls on a regular basis but using more spaced intervals than rtPS. This assures that the flow receives request opportunities even during congestion. The CM is also allowed to use contention and piggyback opportunities.

- Best Effort (BE)
  The intent of the BE service is to provide efficient service to the best effort traffic. The SF is allowed to use contention and piggyback request opportunities.

Different aspects from the performance of DOCSIS based HFC networks are studied in the literature covering issues for the TCP performance, the performance of the contention channel, the MAC layer and also different scheduling algorithms and mechanisms. Most of these topics have been tackled by both simulations and theoretical analysis. In [9] and [10] an extensive set of simulation results of downstream and upstream channel utilization, the average upstream delay and web response time is reported. Using theoretical analysis and simulations the performance of web-browsing was studied

in [11]. It was shown that the number of Web users that can be supported is about 85% of the ratio of the network rate to the intended average user rate. The performance of the DOCSIS MAC protocol according to the structure of the Bandwidth Allocation Map (MAP) message is evaluated in [12]. According to the study, the throughput and delay show best performance when the MAP size is 2 msec. The goodput for the upstream channel capacity is found to be about 58%.

In DOCSIS the Ethernet frames can be fragmented and concatenated in one transmission. In [13] a detailed study of the influence of the fragmentation and concatenation, the piggybacking and the effect of the priorities on the delay is conducted. Conclusions are as expected: the higher priority traffic has lower upstream delay than the lower priority one; a system where concatenation and fragmentation are enabled shows significant improvement in the achieved throughput (40% vs. 70 %); and finally the use of piggybacking improves the delay. The latter issue is shown to be important for the TCP traffic.

The fragmentation can significantly improve the throughput especially when there are 'Unsolicited Grants Service' (UGS) grants with strict jitter requirements. However packet fragmentation comes with extra overhead. In [14] is concluded that the scheduling scheme should try to fragment as little as possible. To do so the algorithm must try to group unsolicited grants together and schedule them over consecutive slots.

In [15] a detailed study of the performance of TCP over DOCSIS is reported. It is shown that due to the asymmetry in the network's upstream and downstream bandwidth, the maximum throughput is limited in both directions when the number of simultaneous transfers is small. Analytical expressions for the asymmetry ratio and the throughput are derived. Testbed experiments reported in [16] demonstrate the limitation on the TCP throughput. In [17], [18] and [19] changes in the MAC layer and a scheduling mechanism are proposed which improve the TCP behavior over DOCSIS. The "Fast Request Transmission" changes the MAC layer to allow more than one outstanding request per SF at the CMTS. The "Long Packet Deferment" mechanism treats ACK and data packets differently giving precedence to the former. Simulations of the impact of the DOCSIS MAC on TCP reported in [16] confirm the limit on the throughput.

In [20], [21] a theoretical model is developed in order to dimension the contention channel in cable networks. It is proved that the optimal size of the contention channel, when it is fixed in size, is 10-15% of the upstream bandwidth.

A number of upstream scheduling algorithms for DOCSIS 1.1/2.0 are proposed in the literature. In [22] the authors propose and study a dynamic bandwidth allocation for the HFC DOCSIS MAC protocol which explores an implementation of SCFQ (self clocked

fair queueing, [23]). They also present an implementation of UGS scheduling based on the Shaped Virtual Clock algorithm. In [24] the authors present a scheduling architecture which supports QoS. The scheduler consists of a plurality of FIFO queues. One is designated for the unsolicited request ,which have to be generated from the scheduler and have stringent delay and jitter requirements. A set of the FIFO queues is designated for data grant for flows with minimum bandwidth guarantee. These grants are generated based on bandwidth requests from these flows. Finally there is a queue for the grants for flows without minimum guaranteed rate. A fair algorithm like Weighted Fair Queuing (WFQ) [25] or Start-Time Fair Queuing (STFQ) [26] schedules amongst the solicited grants queues. In [27] a formal theoretical analysis and formulation of the optimal upstream scheduling problem is given. The authors use it to identify important relations amongst the scheduling parameters like MAP frequency and channel utilization. More sophisticated and bandwidth allocation algorithms are proposed in [28] and in [29]. The first one is traffic based and requires predefined knowledge of the traffic to be scheduled. The latter uses high complexity calculations for the assignment of each mini-slot.

In the downstream direction, i.e., from the CMTS to the CMs any packet scheduling algorithm like the ones discussed in Section 1.2 or in Chapter 4, can be deployed.

1.1.2 Ethernet Passive Optical Networks

A passive optical network (PON) is a subscriber access network technology that provides high bandwidth capacity over fiber. It is also a point to multipoint network with a tree topology. The terminal equipment connected at the trunk of the tree is referred to as an Optical Line Terminal (OLT) and typically resides at the service provider's facility. The OLT is connected to a passive optical splitter using an optical trunk fiber, which fans out at the splitter to multiple optical drop fibers to which Optical Network Units (ONUs) are connected. ONUs typically reside at the subscriber premises, which can be end-user locations or curbs resulting in different fiber-to-the home, business or curb (FTTx) architectures (Fiber-To-The-Home, Fiber-To-The-Business or Fiber-To-The-Curb).

One of the most beneficial features of a PON is that the same fiber infrastructure allows the transporting of different formats - WDM, ATM or Ethernet [30]. The Ethernet protocol is highly deployed in local area networks (LANs) and it is also becoming an emerging technology for metropolitan and wide area networks with the standardisation of 10 gigabit Ethernet. Thus, Ethernet is an attractive protocol choice for the access network with its technological simplicity and customer familiarity. A PON, which provides transport of Ethernet frames is called Ethernet PON (EPON) [6]. EPON is a part

of the IEEE802 standards family and therefore an implementation may make use of the wide spread Ethernet equipment. A detailed description of the technology can be found in for example [31] and [32]. Hereafter an overview is provided.

An EPON supports a nominal bit rate of 1Gb/s, shared amongst the ONUs, which can be at a maximum distance of 20 km. There are two wavelengths – one for the down- and one for the upstream direction. In an EPON, all downstream (from the OLT to the ONU) Ethernet frames transmitted by the OLT, reach all ONUs. ONUs will discard frames that are not addressed to them. In the upstream direction (from the ONU to the OLT) the signal transmitted from the ONU is received only by the OLT. The scheduler at the OLT arbitrates the upstream transmissions from the ONUs by granting Transmission Windows (TWs), which can have variable lengths. An ONU is only allowed to transmit during the TWs allocated to it. In order to inform the scheduler about its bandwidth requirements, ONUs use REPORT messages that are also transmitted (along with the data) in the TW.

In an EPON the upstream bandwidth allocation algorithm is responsible for assigning the upstream channel bandwidth amongst the traffic of the different users and priorities. Unlike DOCSIS, in EPON it is relied on the inherent Ethernet QoS expressed in the 8 possible packet priorities and smart implementation dependent scheduling mechanisms to provide QoS . Thus, the design choices for the algorithm determine the fairness and efficiency of the network, the different QoS parameters like the packet delay, delay variation and bandwidth guarantees.

There are several upstream bandwidth allocation algorithms proposed in the literature. In [33] an algorithm was proposed where the length of the TWs depends on the ONU's current bandwidth requirements. This algorithm was extensively studied and improved in order to support differentiated services in [34]. Roughly speaking, this algorithm works as follows: All ONUs get a TW in a cyclic order, during each TW an ONU will transmit some data as well as a REPORT message to update the OLT's knowledge about this ONUs bandwidth requirements. The length of a TW of ONU i is completely determined by the contents of the REPORT transmitted in the previous TW of ONU i, that is, the OLT grants a TW with a length equal to the minimum of the requested amount of bandwidth and a predefined maximum (plus the size of a REPORT message).

Other types of algorithms are cyclic-based. Such algorithms allocate bandwidth based not only on the ONU's request but on the total bandwidth requested (by all ONUs) during a cycle. Such algorithms provide better fairness and minimum bandwidth guarantee. The cycle can have fixed [35], [36] or variable length [37], [38], [39], [40], [41]. In [32] the authors study a hierarchical scheduling algorithm, applicable for bandwidth

allocation in EPON, which uses service envelops. A service envelope according to the authors is the amount of service given to a node as a function of some nonnegative value called satisfiability parameter (SP). SP is a measure of how much demand for bandwidth can be satisfied for a given node.

Frames are never fragmented in EPON, therefore, the IEEE working group introduced the concept of *threshold reporting* in order to achieve a higher bandwidth efficiency. The mechanism is crucial for the utilization of the upstream bandwidth and is described in detail in Chapter 2, where also several novel algorithms are described.

Again, in the downstream direction, i.e., from the OLT to the ONUs, any packet scheduling algorithm like the ones discussed in Section 1.2 or in Chapter 4, can be deployed.

1.1.3 DOCSIS 3.0

In order to be able to provide higher bandwidths and to compete with the optical access networks offerings a new technology for hybrid fiber coax (HFC) networks is specified in the DOCSIS 3.0 specification [5]. Particularly DOCSIS 3.0 defines a mechanism of forwarding upstream and downstream packets between the CMTS and the CM by utilizing the bandwidth of multiple physical layer channels. In this way the throughput can be significantly increased without increasing the modulation and still use the legacy cable. While in the pre DOCSIS 3.0 specifications, the communication between the CMTS and the CMs is realized on two channels: one for the downstream and one for the upstream, in DOCSIS 3.0, in each direction there are multiple channels. The CMTS can transmit or receive simultaneously on all of them. The mechanism is termed channel bonding and is realized on MAC level. By balancing the CMs amongst the channels, the full capacity of a channel bonded system can be utilized. However DOCSIS 3.0 significantly expands the downstream service offering by requiring the DOCSIS 3.0 CM to be capable of receiving on multiple channels simultaneously.

The individual CMs can have different capabilities in terms of the number of channels they can receive simultaneously. The legacy CMs can receive on only one channel, while the DOCSIS 3.0 CMs on 2,3 and more channels. An example of a system with 4 channels is given on Figure 1.3. There are 4 modems reached by these channels. However they have different capabilities and thus are balanced amongst the channels. CM1 has 4 receivers and its SF is assigned to all channels. CM2 and CM4 have each 2 receivers and are assigned correspondingly to the bonding groups (BG): $BG1$ consisting of frequencies $f1$ and $f2$ and $BG2$ consisting of $f2$ and $f3$. Finally, CM 3 has only one receiver and is

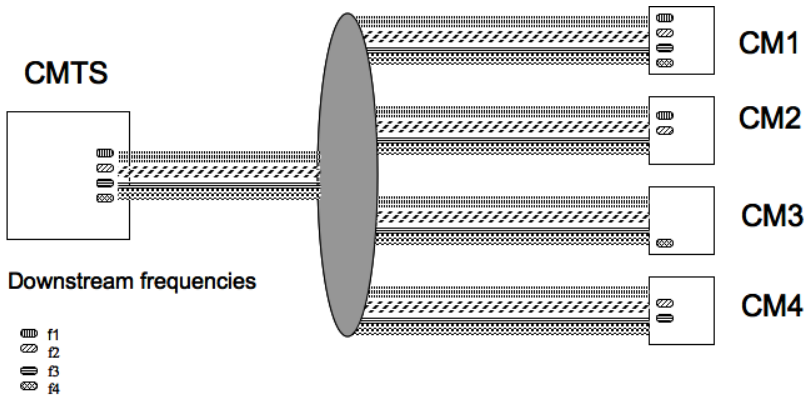


Figure 1.3: Frequency Assignment

assigned to $f4$. Thus a packet transmitted on channel $f1$ reaches CM1 and CM2, while a packet transmitted on channel $f2$ reaches CM1, CM2 and CM4. Similarly a packet transmitted on channel $f3$ would reach CM1 and CM4, and a packet transmitted on $f4$ will reach CM1 and CM3.

The scheduler at the CMTS has to distribute the packets over the set of DownStream (DS) channels for delivery to a single CM. Thus the schedulers described in Section 1.2 or in Chapter 4 are not readily applicable.

Each complete packet is transmitted on a single channel. The channels can have different modulation and thus different bit rates. The packets are tagged with a sequence number. In this way the proper packet sequencing is not lost if there are different latencies on the channels. The CM restores the original sequence before forwarding the packets to the user devices.

DOCSIS 3.0 uses the same mechanism as specified in DOCSIS 2.0 to provide QoS. The principal mechanism is the classification of the packets into Service Flows and scheduling those SFs according to a set of QoS parameters.

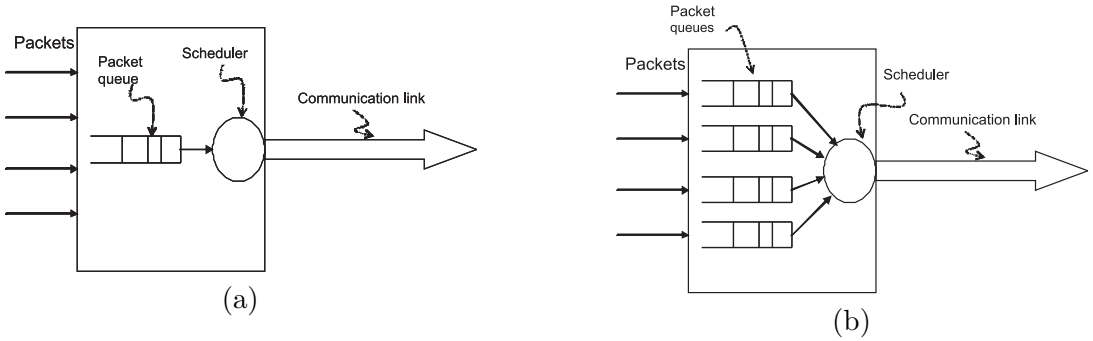


Figure 1.4: Packet scheduling architecture (a) single queue (b) per-flow queuing.

1.2 Packet Scheduling Algorithms

A wealth of scheduling disciplines are available in the literature with the most simple one being the First-Come-First Served (FCFS), known also as First-In First-Out (FIFO). The FCFS discipline, as the name says, serves the packets in order of their arrival at the system. There is also the Last-Come First-Serve discipline (LCFS) which transmits the packets in the opposite order, i.e., the last arrived is the first one to be served. These disciplines have constant time low complexity. In high-speed packet routers the algorithms are still mainly based on FCFS having only one queue where the packets from all the flows are queued. However, in order to guarantee QoS, to isolate the flows and to provide fairness, the network nodes should support per flow queuing. Figure 1.4 shows a schematic view of the two queuing types. In a per flow queuing algorithm there is a separate queue for each flow, hence the names queue and flow are used interchangeably. Depending on the network, packets are classified as belonging to a flow, for example, based on the destination or source address, the TCP port, or a combination of these or other classifiers.

The Generalized Processor Sharing (GPS) scheduling discipline [42], a natural generalization of processor sharing [43], is a flow-based multiplexing discipline. GPS is a fluid server[1], which treats the traffic as infinitely divisible and it serves multiple flows simultaneously. A GPS server is work conserving i.e. it is busy if there are packets waiting in the system. It operates at a fixed rate r. A GPS server with N queues is characterized by a set of N positive numbers $\{\phi_i\}$. Let $W_i(\tau, t)$ be the amount of flow's i traffic served in the interval $(\tau, t]$. A flow is backlogged at time t if a positive amount

[1]In the dissertation server, scheduling algorithm, discipline and scheduler are used as synonyms

of the flow's traffic is queued at time t. Then a GPS server is defined as one for which

$$\frac{W_i(\tau,t)}{W_j(\tau,t)} \geq \frac{\phi_i}{\phi_j}, j = 1, 2, ..., N \tag{1.1}$$

for any flow i that is continuously backlogged in the interval $(\tau, t]$.

Let $B(\tau, t)$ be the set of all backlogged flows in the time interval (τ, t). The minimum traffic transmitted from flow $i \in B(\tau, t)$ is determined from ϕ_i i.e.

$$W_i(\tau,t) \geq \frac{\phi_i}{\sum_{j \in B(\tau,t)} \phi_j} * r(t-\tau). \tag{1.2}$$

Thus a flow is guaranteed a rate of

$$g_i \geq \frac{\phi_i}{\sum_j \phi_j} * r. \tag{1.3}$$

Furthermore the excess bandwidth available from flows not using their minimum bandwidth, is distributed amongst the backlogged sessions at each instant in proportion to their guaranteed bandwidth portion. This results in a work conserving discipline with perfect isolation of the flows, ideal fairness and full link utilization. Moreover the delay of flow i can be bounded as a function of its queue length and it is possible to make worst-case queuing delay guarantees when the flows are leaky bucket constrained [42]. Thus the GPS possess almost all required properties for a scheduling algorithm. However, being a fluid server it presumes that the traffic is infinitely divisible which in real networks is not the case. There are a significant number of packet scheduling algorithms which attempt to approximate the GPS discipline from Equation (1.1), the GPS packet order or the rate allocation from Equation (1.2) and (1.3). They are discussed in the following subsections where they are classified based on their internal structure as *sorted-priority, round robin or hybrid*.

1.2.1 Sorted-Priority Based Schedulers

The sorted priority algorithms try to determine an order in which the packets will be served, which closely approximates the order in a GPS server. To achieve this, they typically simulate in parallel a GPS system. To do so at least one timestamp is associated with each packet or to save on space and computational time, with the first packet in the queue of each backlogged flow. The packets are then served in increasing order of the timestamp.

The finishing time of packet k from flow i with length L_i^k is given by

$$F_i^k = S_i^k + \frac{L_i^k}{\phi_i}. \tag{1.4}$$

Here S_i^k is the packet start time and is calculate from

$$S_i^k = max(F_i^{k-1}, V(t)). \tag{1.5}$$

The function V(t) is referred to as virtual time and is typically different for the different approximations of GPS. In Weighted Fair Queuing (WFQ) [25] or Packet Generalized Processor Sharing (PGPS) [42] it is the normalized amount of service that all backlogged sessions will receive by time t

$$V_{PGPS}(\tau + t) = V_{PGPS}(\tau) + \frac{t}{\sum_{i \in B(\tau, \tau+t)} \phi_i}. \tag{1.6}$$

So at each moment of time it depends on the number of the backlogged flows and their respective shares ϕ_i. Therefore the algorithm has O(N) complexity, where N is the number of active flows.

In Virtual Clock (VC) [44] scheduling algorithm the virtual time function is the current system time. This choice reduces the complexity but increases the unfairness as will be discussed in detail in Section 1.3.3. In WFQ and VC algorithms the timestamp is the finishing time of packet F_i^k. Some algorithms, like Start-Time Fair Queuing (STFQ) [26] and Self-Clocked Fair Queuing [23] use as a timestamp the starting time of the packets. An optimal packet approximation of GPS is proposed in [45], termed Worst-case Fair Weighted Fair Queuing WF^2Q. Conceptually, the algorithm works by combining both criteria, start and finish time. In a first shaping step all eligible packets are selected, that is, all packets with starting time not later than the current system virtual time. From all these packets, the one with the smallest finish time is sent next. This packet selection policy, termed Smallest Eligible Virtual Finish-time First (SEFF) ensures tight bounds for all quality indices.

The SEFF selection policy has attracted significant research efforts to propose different implementations with low complexity. The authors of [46] independently arrived to the same conclusion that the combination of traffic shaping and finish-time service results in optimal scheduling characteristics and proposed an algorithm termed Shaped Framed Fair Queuing (SFFQ). In [47] the authors propose an implementation of their SFFQ algorithm with $O(logN)$ complexity for ATM networks. In [48] an implementation

of an algorithm with the SEFF policy is proposed called Leap Forward Virtual Clock (LFVC) that has O(1) complexity, though with high constant overhead, for most of the operations but which on occasions can reach $O(N)$. The algorithm in [49] builds upon LFVC to achieve smaller constant execution overhead.

WFQ and WF^2Q are based on real-time simulation of a GPS server, which under straightforward implementation has $O(N)$ complexity. Efficient implementations of WFQ are discussed in [50] where it is shown that the complexity can be minimized to $O(logN)$. Different implementations and approximations of the algorithm WF^2Q are studied in [51] and [52]. However one should note that any approximation of the GPS virtual time has an effect on the delay bound and the tradeoffs are discussed in [53]. The complexity of the algorithms is due to *a*) the calculation of the virtual time function and *b*) the ordering of the packets. A typical value for the complexity for the packet ordering is O(logN) (see [54]). However the lowest complexity that can be achieved for sorting (ordering) the packets is via pipelined heap (P-Heap) [55] and Emde-Boas priority queue [56] and is O(loglogN). A precise simulation of GPS virtual time has been considered as being an operation with linear complexity, since it needs to keep track of all changes in the set of flows backlogged in the GPS reference system. A proposal in [57] allows for exact simulation of GPS virtual time with $O(logN)$ algorithmic complexity in the number of flows. Independent analysis in [58] shows $O(logN)$ to be a lower bound.

In [59] the authors propose WF^2Q+ algorithm which uses an approximation of the virtual time function. With WF^2Q+, the virtual time function is defined as

$$V_{WF^2Q+}(t+\tau) = max(V_{WF^2Q+}(\tau) + W(\tau, \tau+t), min_{i \in B(\tau)}(S_i^{h_i(\tau)})), \tag{1.7}$$

where $W(\tau, \tau+t)$ is the total amount of service provided by the server during the period$[\tau, \tau+t]$, $B(\tau)$ is the set of backlogged flows in the system at time τ, $h_i(\tau)$ is the sequence number of the packet at the head of the flow i's queue, and $S_i^{h_i(\tau)}$ is the virtual start time of the packet. The WF^2Q+ algorithm is the closest approximation of GPS server, with the major drawback being the high complexity.

1.2.2 Round Robin Schedulers

In round robin schedulers the packets or the flows to be served have already some predefined order (round robin) and no sorting is needed, which reduces significantly the complexity of the algorithms. While a simple cell-by-cell round robin algorithm will be sufficient for ATM networks, the variable packet size should be taken into account for packet networks, in order to satisfy the requirements for a suitable scheduling algorithm.

The first round robin packet scheduling algorithm proposed in the literature is the Deficit Round Robin (DRR). It is also the most implemented per-flow queuing algorithm in high-speed packet routers [60], [61].

In the following the DRR algorithm is outlined, a detailed description of which is presented in [62]. Consider a number of flows contending for a link. The packets of each flow i are stored in a corresponding queue i. To each flow is assigned a weight w_i and a quantum Q_i proportional to that weight. The quantum indicates the portion of the resources a flow should get in a round robin cycle. Also to each flow i is associated a counter called deficit counter DC_i, which indicates the amount of service the flow can still receive in a round. The flows are serviced in a round robin order. Each round a flow is visited once. Upon a visit to a flow its deficit counter is increased with its quantum. After a packet is sent it is decreased with the size of the packet. In this way if a part of the deficit counter remained unused in a round it will be still available for the next one. Only backlogged flows are serviced. To realise this the DRR scheduler maintains a list of all backlogged flows. When a flow is no longer backlogged it is removed from the list and its deficit counter is set to 0. When a flow becomes backlogged it is inserted at the tail of the list.

If the minimum quantum, say Q_{min}, is chosen not less than the maximum packet length in the network, each time the scheduler visits a queue it will be able to serve at least one packet thus the algorithm would have an $O(1)$ complexity. The quantums of the flows can be obtained from $Q_i = w_i Q_{min}$ with $w_i \geq 1$.

Several other round robin algorithms are proposed in the literature. The Nested DRR [63] divides the regular DRR round in several sub-rounds. In each sub-round a flow is served at most the minimum quantum for the network. The scheduler remains in the same round as long as there are backlogged flows, which have deficit counters not less than the length of the next packet in their queue. Thus the flow with the maximum weight determines the number of sub-rounds in a round. According to the implementation proposed in [63] flows which have received their quantum for a round are moved to a second list. When the first list is empty the scheduler swaps the lists and this marks the start of a new round. In this way the scheduler doesn't have to visit flows, which have received their quantums for the round. However consider a flow, which becomes backlogged and has fewer packets than its quantum to send. After it is no longer backlogged its state is lost and if it becomes backlogged again in the same round it will be able to transmit again the full quantum. Thus the proposed algorithm for the nested DRR cannot guarantee neither fairness nor isolation of the flows nor delay bound. Moreover the fact that it cannot bound the traffic sent from a flow implies that

it also can not bound the end-to-end delay.

The Elastic round robin scheduler proposed in [64] and further developed in [65] exploits the different packet sizes in the network and potentially can achieve a lower delay bound than DRR. However, as pointed out in [66], the algorithm has the same drawback as Nested DRR and in fact can not guarantee fairness or delay.

The Surplus Round Robin (SRR) proposed in the context of a stripping algorithm [67] keeps track of the surplus from the previous round. More precisely, in the beginning of each round the flow's surplus counter is increased with its quantum and in a round, a flow can transmit packets as long as its surplus counter is positive. This algorithm has the advantage over DRR that it doesn't need the size of the next packet in order to make the scheduling decision. However a straightforward implementation, like the one discussed in [68], would fail to achieve fairness, isolate the flows and guarantee end-to-end delay for the same reasons Nested DRR and ERR do not.

In [69] and [70] the authors propose an algorithm Priority DRR, which combines FCFS queuing with DRR. They show that it is scalable and it provides very low average latency for flows transmitting at rates lower than their estimated fair share of the link rate. However the proposed implementation does not have O(1) complexity as the authors acknowledge in [69].

1.2.3 Hybrid schedulers

Under hybrid schedulers are classified algorithms, which combine more than one scheduling discipline. In Bin-Sort Fair Queuing (BSFQ) [71] the virtual time is divided into slices of equal length called bins. As in sorted-priority schemes, each arriving packet is stamped with a value, which represents the virtual departure time. The packet is then distributed in the bin that corresponds to the time slice that contains the virtual departure time of that packet. The packets sorted in the same bin are queued in FIFO order. The complexity depends on the number of empty bins the scheduler has to check before transmitting a packet from a non-empty queue. If there are in total V bins then in the worst case the complexity of the algorithm is O(V). Note that the less bins there are the worst the fairness and delay guarantees with the limit being only one bin, which corresponds to FIFO scheduling.

Eligibility-Based Round Robin (EBRR) [66] extends on the ideas outlined in [72] and [73], where a round is divided into Z sub-rounds with equal size. For each sub-round the flows are added to an active list and served according to a packet round robin scheduling discipline (DRR for example). Each flow has a quantum for a sub-round.

When a packet is served, the flow will be added to the sub-round, which is further away from the current one by a number corresponding to the packet size expressed in quantums. The complexity of the algorithm depends on the number of empty lists for the sub-rounds, which the scheduler has to check. Thus in the worst-case the complexity is O(Z).

In Pre-order DRR [74] a priority queue module is added to the original DRR scheduler which reorders the packet transmission sequence in DRR to distribute the output more evenly amongst the flows. The packets, which can be transmitted in a round from all backlogged flows are first classified in the priority queues according to the flows' quantums and then the priority queues are served according to a simple FIFO. Even though the authors claim that the Pre-order DRR scheduler has O(p) complexity where p is the number of the priority queues it is obvious that the classification of the packets in the queue will invoke additional complexity of the number of packets that can be sent in a round. In [66], it was pointed out that the effective complexity is $O(N + p\log p)$ where N is the number of backlogged flows.

In Stratified Round Robin (Stratified RR) [75], Fair Round Robin (FRR) [76] and Group Ratio Round Robin(GR3) [77] the flows are divided in groups based on their reserved rate. The intra-group scheduling is (deficit) round robin. The inter-group scheduling in GR3 is simple weighted round robin while Stratified RR and FRR use a sorted priority algorithm.

Smoothed Round Robin [78] and Recursive Round Robin [79] use the fact that the rates reserved for the flows are typically a power of 2 and code them binary. They make the scheduling decision by using this code and a weight spread sequence (WSS), which can be generated via a tree structure [79]. They still need to use deficit counters in order to be applicable to packet networks.

Although in most of the cases the scheduling decision for the hybrid schedulers can be made in constant time none of them has a worst-case O(1) scheduling complexity. They all explore the trade-offs between complexity, fairness and delay guarantees.

1.3 Performance Analysis of Scheduling Algorithms

1.3.1 Simulation Models

Complex systems serving hundreds or thousands of flows or with complicated MAC layers cannot always be modelled analytically. Instead simulations are required in order to study the system behaviour.

There are a number of software tools available like OPNET [80] - commercial, $ns2$ [81], OMNET++ [82] - free software, suitable to model telecommunication systems. Some of them have already detailed models of some of the access networks. Depending on the desired depth of the model it can include part or all of the network topology, the different nodes in the network and all or a number of the layers in the protocol stack. For network technologies, which are not included as a model it is an overkill to use these tools as they add useless overhead. Instead a simple tailor made simulation tool can be developed.

OMNET++ is an object-oriented modular discrete events simulator with graphical user interface [82]. An OMNET++ model consists of hierarchically nested modules. Modules communicate with message passing. Messages can contain arbitrarily complex data structures. Modules can send messages either directly to their destination or along a predefined path, through gates and connections. Modules at the lowest level of the module hierarchy (simple modules) are provided by the user and they contain the algorithms in the model. During simulation execution simple modules appear to run simultaneously. The programming language of the algorithms in the simple modules is C++ [83].

1.3.2 Delay Bounds and Latency-rate (LR) Schedulers

For some simple scheduling disciplines it is possible to apply queuing theory methods [84], [85] to analyze them and to predict delay and buffer sizes. For example, in [86] was demonstrated how the GPS model can be used to describe the flow level characteristics of the traffic of the subscribers of a high-speed access network. In [87], queuing theory is used to compare two scheduling disciplines. However, due to the big number of users and states, the real schedulers are too complex to apply queuing theory methods to analyze them. In order to characterize and compare scheduling algorithms different metrics are used.

The use of the worst-case performance of a scheduling discipline as a metrics has first been considered in [88]. In [42] the authors have first proposed to evaluate the worst-case performance guarantees for leaky bucket traffic, which they did for the PGPS/WFQ scheduler. They used the same idea to evaluate the worst-case performance of networks with PGPS scheduling nodes [89].

Further developing the idea to classify schedulers based on their worst-case bounds of the service received by a flow, in [90] and [91] a common framework, called $Latency-Rate$ (LR) schedulers, has been developed for the analysis of traffic scheduling algorithms.

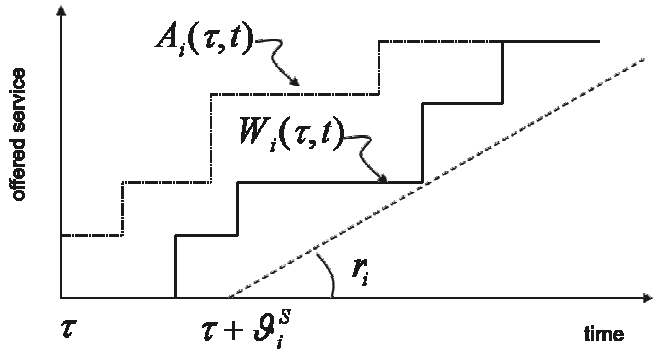


Figure 1.5: An example of the latency θ for an LR scheduler

Service refers to the amount of traffic transmitted. Besides that it is useful as a metric, from such a service bound, one can easily obtain a delay bound provided the arrival process is bounded like leaky-bucket shaped traffic. Hereafter a short description of the latency-rate server concept is given. It is going to be used further in the dissertation to evaluate schedulers and to establish delay and service bounds.

The LR scheduler model introduces the notion of *latency*, which bounds the time that a flow has to wait until it begins receiving service at its guaranteed rate. From the latency one can obtain upper bounds on the delay, backlog in the queue and burstiness of the output traffic.

Consider a flow i, having reserved rate r_i and let $W_i(\tau, t)$ be the amount of service received by the flow in the time interval (τ, t). The reserved rate is in fact the rate which the scheduler promises to guarantee for the flow. The latency of a scheduling algorithm S is defined in [91] as the minimum non-negative constant θ_i^S that satisfies

$$W_i(\tau, t) \geq max(0, r_i(t - \tau - \theta_i^S)) \tag{1.8}$$

where τ is a start of a busy period and t is any time instance within this busy period. A busy period is defined as the maximal interval of time (τ, τ_1) such that for any time $t \in (\tau, \tau_1)$, packets arrive for flow i with rate greater than or equal to its reserved rate r_i, or,

$$A_i(\tau, t) \geq r_i(t - \tau). \tag{1.9}$$

The concept is illustrated on Figure 1.5. The latency of a guaranteed rate scheduler can be considered as the cumulative time that a flow has to wait until it receives service at its guaranteed rate. In [91] is shown that if the service can be bounded with Equation

(1.8) for an interval of time within flow's backlogged period with some latency $\tilde{\theta}_i^S$ than this is an upper bound to the scheduler's latency ,i.e., $\theta_i^S \leq \tilde{\theta}_i^S$.

The arrival traffic from a leaky-bucket shaped traffic source with parameters (σ_i, r_i) is bounded by

$$A_i(\tau, t) \leq r_i(t - \tau) + \sigma_i. \tag{1.10}$$

From the latency of a scheduler given by Equation (1.8) and considering Equation (1.10) one can easily obtain a bound on the maximum delay D_{max} for a leaky-bucket shaped traffic source. This is given by

$$D_{max} \leq \frac{\sigma_i}{r_i} + \theta_i^S \tag{1.11}$$

1.3.3 Fairness

A common metric used to evaluate the fairness of the algorithm is defined in [91] and [23]. Lets assume that $W_i(\tau, t)$ is the service offered to flow i in the interval (τ, t), r_i is the rate allocated to flow i and $\frac{W_i(\tau,t)}{r_i}$ is the normalized service offered to the flow in the period. The fairness Φ^S is defined as

$$\left| \frac{W_i(\tau, t)}{r_i} - \frac{W_j(\tau, t)}{r_j} \right| \leq \Phi^S, \tag{1.12}$$

for all flows i and j, which have an infinite supply of packets in the time interval (τ, t). This forces them to be continuously backlogged, regardless the scheduling algorithm used. A scheduler is considered fair if its metric is bounded i.e. $\Phi^S < \inf$. The ideal GPS is perfectly fair with $\Phi^{GPS} = 0$. Further on in Section 1.3.4 we give the fairness metric for some of the major schedulers.

1.3.4 Comparison Between Scheduling Algorithms

In Table 1.1 the values of the complexity, latency and fairness for the major LR scheduling algorithms, which are known up to date, are summarized. N is the number of flows, r is the bandwidth of the shared output link, r_i, r_j is the reserved rate of flow i, j. The maximum packet size for flow i is given by $L_{i,max}$, while the overall maximum packet size is L_{max}. L_i, L_j is the size of a packet from flow i, j. In the expressions for the latency and fairness of the BSFQ scheduler Δ is the bin size. The number of bins should be bounded by $V \geq \frac{B}{\Delta r}$, where B is the buffer size.

In the table WF^2Q+ algorithm is not included because there is no formal proof that it is an LR scheduler. However a delay bound for leaky-bucket constraint traffic was derived in [59] being $\frac{\sigma_i}{r_i} + \frac{L_{max}}{r}$. This bound makes WF^2Q+ the algorithm with lowest delay bound . It has $O(logN)$ complexity.

Table 1.1: Comparison of LR Schedulers

Scheduler	Complexity	Fairness	Latency
GPS [91]	-	0	0
VC [91]	O(logN)	∞	$\frac{L_i}{r_i} + \frac{L_{max}}{r}$
WFQ/PGPS [91]	O(logN)	O(N)	$\frac{L_i}{r_i} + \frac{L_{max}}{r}$
SCFQ [91]	O(logN)	$\frac{L_{i,max}}{r_i} + \frac{L_{j,max}}{r_j}$	$\frac{L_i}{r_i} + \frac{L_{max}}{r}(N-1)$
FFQ [91]	O(logN)	$\frac{2F}{r} + max(\frac{L_i}{r_i} + \frac{L_j}{r_j})$	$\frac{L_i}{r_i} + \frac{L_{max}}{r}$
Pre-order DRR [74] [92], [66]	O(N+plogp), p-the number of priorities	$\frac{2F+\frac{F}{p}}{r}$	$\frac{\frac{F-Q_i}{p}+(N-2)(L_{max}-1)}{r} + \frac{L_{max}-1}{r_i}$
BSFQ [71]	O(V), V is the number of bins.	$2(\frac{L_{i,max}}{r_i} + \frac{L_{j,max}}{r_j} + \Delta)$	$\frac{\Delta(r-r_i)-(N-2)L_{max}}{r} + \frac{L_{max}}{r_i}$
EBRR [66]	O(Z), Z i s the number of sub-rounds	$\frac{F}{Zr} + \frac{L_{max}-1}{r_i} + \frac{L_{max}-1}{r_j}$	$\frac{\frac{F-Q_i}{Z}+(N-1)(L_{max}-1)}{r}$
DRR [93], [94]	O(1)	$\frac{F}{r} + \frac{L_{max}-1}{r_i} + \frac{L_{max}-1}{r_j}$	$\frac{F-Q_i+(N-2)(L_{max}-1)}{r} + \frac{L_{max}-1}{r_i}$

Chapter 2

Upstream Bandwidth Allocation in Ethernet Passive Optical Networks (EPON)

This chapter describes several upstream scheduling algorithms for EPON, which make use of the Multi-Point Control Protocol (MPCP) with threshold reporting. It also describes the designed mechanisms to generate reports with thresholds at the Optical Network Units (ONUs) and to utilize this information at the Optical Line Terminal (OLT).

An upstream scheduling algorithm for EPON is defined by the scheduling mechanisms in both the ONU and in the OLT. Two types of scheduling at the ONU are considered, namely full priority scheduling, which is the common strict priority scheduling and interval priority scheduling, where higher priority traffic is transmitted before lower one only if it was already reported to the OLT. The types of algorithms proposed for the OLT differ by the threshold level, on which the scheduling is based - full and single report threshold level - and by their scheduling policy for constant bit-rate traffic.

The proposed upstream bandwidth allocation algorithms are compared by means of a detailed simulation program regarding average packet delay for several priorities, delay variation for constant bit rate (CBR) traffic and bandwidth utilization and the trade-offs between them are discussed. The algorithms and results have been published in [95], [96], [97] and [98].

2.1 Background and Overview

Recall from Section 1.1.2 that Ethernet Passive Optical Networks are subscriber point to multipoint access network with a tree topology. The terminal equipment connected at the trunk of the tree is referred to as an Optical Line Terminal (OLT) and typically resides at the service provider's facility. The subscriber equipment is referred to as the Optical Network Unit (ONU). Unlike HFC networks, EPON uses only optical fibres to connect the OLT to the ONUs thus, offering higher speeds and obviates the need of active elements in the field. An EPON supports a nominal bit rate of 1Gb/s, shared amongst the ONUs, which can be at a maximum distance of 20 km. There are two wavelengths – one for the down- and one for the upstream direction. In an EPON, all downstream (from the OLT to the ONU) Ethernet frames transmitted by the OLT reach all ONUs. ONUs will discard frames that are not addressed to them. In the upstream direction (from the ONU to the OLT) the signal transmitted from the ONU is received only by the OLT. The physical medium dependent layer in EPON also differs from the one specified in DOCSIS (recall Figure 3.4 from Section 3.1.1) The OLT and the ONUs transmit Ethernet frames at wire speed. In front of each frame there is a preamble of 8 bytes and between two frames there is at least a 12 byte inter-packet gap (IPG) [7]. The scheduler at the OLT arbitrates the upstream transmissions from the ONUs by granting Transmission Windows (TWs), which can have variable lengths. An ONU is only allowed to transmit during the TWs allocated to it. Between the Transmission Windows (TWs) of two ONUs there is a certain guard time g needed to account for the laser on (T_{xON}) and off (T_{xOFF}) times, receiver recovery times and other optics related issues (R_{xrec}) (see Figure 2.1). The FEC is not compulsory in EPON.

A multi-point control protocol (MPCP) standardized in [6] facilitates the implementation of the upstream bandwidth allocation algorithm. The MPCP defines the report-grant mechanism which allows the ONUs to communicate their bandwidth needs to the OLT allowing it to allocate TWs for the ONUs. In attempt to provide means to reduce the idle times in a TW, the standard introduced a novel reporting mechanism called *threshold reporting* . It allows the ONU to include in the report some information on the Ethernet frame boundaries. The exact mechanism will be further detailed in section 2.2.1.

In order to provide QoS, EPON relies on the 8 priorities defined in the Ethernet standard and on the implementation dependent upstream bandwidth allocation mechanism. The later is responsible for assigning the upstream channel bandwidth amongst the traffic of the different users and priorities. The design choices for the algorithm

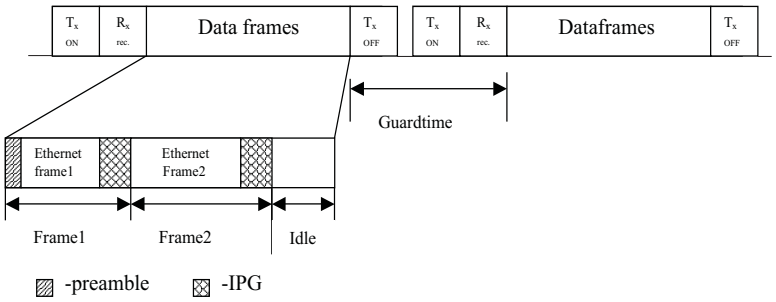


Figure 2.1: Transmission window and frame format

determine the fairness and efficiency of the network and the different QoS parameters like the packet delay, delay variation and bandwidth guarantees. In this chapter several upstream bandwidth allocation algorithms, which use threshold reporting are proposed and studied.

The rest of this chapter is organized as follows: The following section describes the report-grant mechanism of the MPCP protocol. In Section 2.3 a method for generating reports at the ONU with threshold information is proposed. A set of upstream bandwidth allocation algorithms are described in Section 2.4. Two types of intra-ONU scheduling algorithms are discussed and a full threshold level scheduling algorithm at the OLT is defined. The next section discusses a rate-based scheduling algorithm especially suitable for constant bit-rate applications. Section 2.6 discusses a mechanism for thresholds assignment. Section 2.7 describes an extensive set of simulation results evaluating the proposed algorithms. Based on the analysis in this section an algorithm which further improves the throughput of the system by applying a single report threshold level (SRTL) scheduling at the OLT is described in Section 2.8. The following section is devoted to comparing the performance of the two modifications of the bandwidth allocation algorithms. The performance of the SRTL based algorithms is further studied in Section 2.10. Finally the last section summarizes and concludes the chapter.

2.2 Multi-Point Control Protocol (MPCP)

The Multi-Point Control Protocol (MPCP) defines the messages used to control the data exchange between the OLT and the ONUs as well as the processing of these messages. The OLT assigns the TWs via GATE messages. Each ONU uses a set of queues to store its Ethernet frames and starts transmitting them as soon as its TW starts. An ONU can support up to 8 priority queues as defined in 802.1Q [8]. During a TW the ONU sends

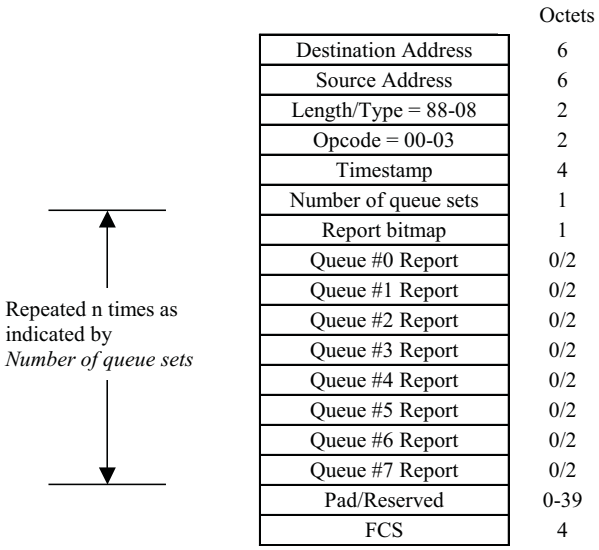


Figure 2.2: Format of a REPORT message

data and/or other management messages such as the REPORT message, the contents of which reflects the ONU's current bandwidth requirements. An ONU can also be forced to send a REPORT message within a TW. All MPCP messages are transmitted as Ethernet frames. During a TW, an ONU is free to transmit its Ethernet frames according to an internal scheduling algorithm. Ethernet frames are not fragmented, causing idle periods in the TWs. For example, if an ONU was granted a TW of 1000 bytes and it has 10 frames ready for transmission, each with a length of 101 bytes (including preamble and IPG), it will send only 9 frames in this TW, which leaves 1000-9*101=91 bytes unused. Also, as the order of Ethernet frames must be maintained, it is not possible to transmit another frame (from the same queue) that fits in the remainder of the TW. To deal with this drawback the threshold reporting concept will be introduced (see Section 2.2.1).

2.2.1 REPORT Messages

An ONU transmits its current bandwidth requirements to the OLT by means of a REPORT message. These requirements are indicated by the number of bytes waiting in each queue. The granularity of the reported queue value is 2 bytes, i.e., if there are x bytes in the queue the reported value is $\lceil \frac{x}{2} \rceil$. This would imply that there is only one value for each queue in a REPORT message, being the current queue length. However, within MPCP, there can be several Queue Reports (QRs) for one queue in a single REPORT message (allowing an ONU to provide some information on the frame bounds as

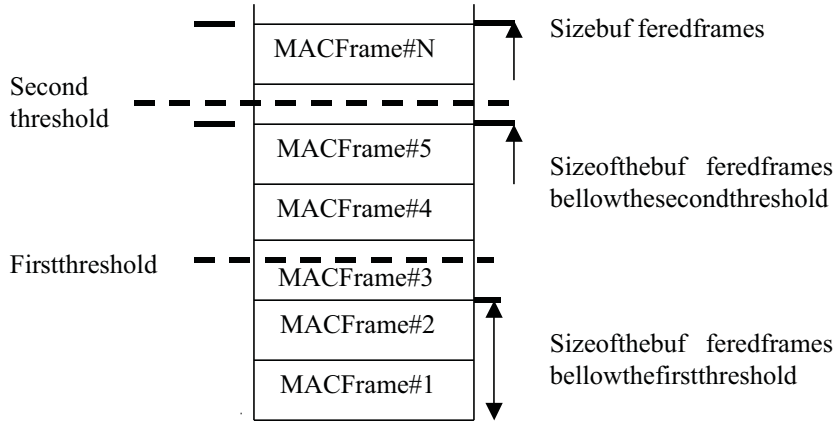


Figure 2.3: Threshold Reporting: the threshold values are used by the ONU to determine the values of the Queue Report (QR) fields

well). As stated earlier, REPORT messages are transmitted as Ethernet frames as can been seen from the format of the REPORT message (Figure 2.2).

The Report bitmap identifies the queues for which QRs follow, e.g., 10011000, indicates that 3 QRs, one for the priority $0, 3$ and 4 queue, follow the report bitmap. The time stamp indicates the local time when the message is transmitted by the ONU. In MPCP, ONU i can have several threshold values $\tau^i_{j,l}$ for queue j (with $l = 1, \ldots 13$). These thresholds are used by the ONU to determine the values of the QR fields (see Figure 2.3). Denote $\beta^i_j(n)$ as the total size of the first n packets waiting in queue j at ONU i. ONU i is said to use the threshold $\tau^i_{j,l}$ if it includes $\beta^i_j(n)$ as a QR in the REPORT message, where $\beta^i_j(n) \leq \tau^i_{j,l} < \beta^i_j(n+1)$. Infinity can also be a threshold value, meaning that an ONU will report all the bytes waiting in this queue. The length of the REPORT message is 64 bytes and when the IPG and the preamble is added the REPORT message has a length of 84 bytes. From these at most 40 bytes are used to report the bandwidth requirements of an ONU (see Figure 2.2).

2.2.2 GATE Messages

The GATE message contains the starting time and the length of a TW, taking the guard time into account. For example, if the OLT wants to grant a transmission window for 1000 bytes to ONU i it actually grants $1000 + g$ bytes where g is the guard time. GATE messages are transmitted as Ethernet frames of 64 bytes.

2.3 Reporting with Thresholds

The standard protocol defines only the format of the threshold report message. The way the thresholds are assigned, the reports filled at the ONU and how this information is used by the scheduler at the OLT is implementation dependent. In this section a mechanism for generating the threshold reports at the ONU is proposed.

Several thresholds, denoted as $\tau_{j,l}^i$ for $l = 1, \ldots 13$, are associated to each queue j of ONU i. It is assumed that the condition $\tau_{j,l}^i < \tau_{j,l+1}^i$ is satisfied. Recall that a REPORT message uses 39 bytes for the QRs and their associated bitmap fields, as a result there can be at most 13 QRs for the same queue j of ONU i in a REPORT message. This can happen if queue j of ONU i is the only non-empty queue. In this case the REPORT message would contain 13 bitmap fields with a single bit set to one, where each bitmap is followed by a single QR; hence, 39 bytes are used to report the state of queue j.

In the current algorithm the last threshold for each queue equals infinity (see Section 2.6), to allow reporting of the total number of bytes waiting in a queue. The ONU includes in each REPORT message at least one QR for each queue that has a length different from 0. Instead of describing the general procedure to generate REPORT messages, lets start with the example presented in Figure 2.4.

In this example there are 7 non-empty queues. The threshold values of queue 0 are multiples of 2160 bytes, while for all other queues the l-th threshold equals $1538l$. The values $v_{j,l}^i$ added below each queue represent the total number of bytes that can be transmitted in a window of length $\tau_{j,l}^i$ without fragmenting packets. The QRs for the queue with the highest priority (priority 0) are created first. To report the state of queue 0, 3 bitmap fields are needed, whose leftmost bit is set equal to 1, and 3 QRs. Thus, 9 bytes are needed to report the state of queue 0: 1 byte for each of the three bitmap fields and 2 bytes for each QR. The values of the 3 QRs equal 1080, 2160 and 2250. For instance, the total number of bytes that can be transmitted in a window of length $\tau_{0,1}^i = 2160$ without fragmenting packets is exactly 2160. Due to the 2 byte granularity of the reported values the QR value of $\lceil 2160/2 \rceil = 1080$.

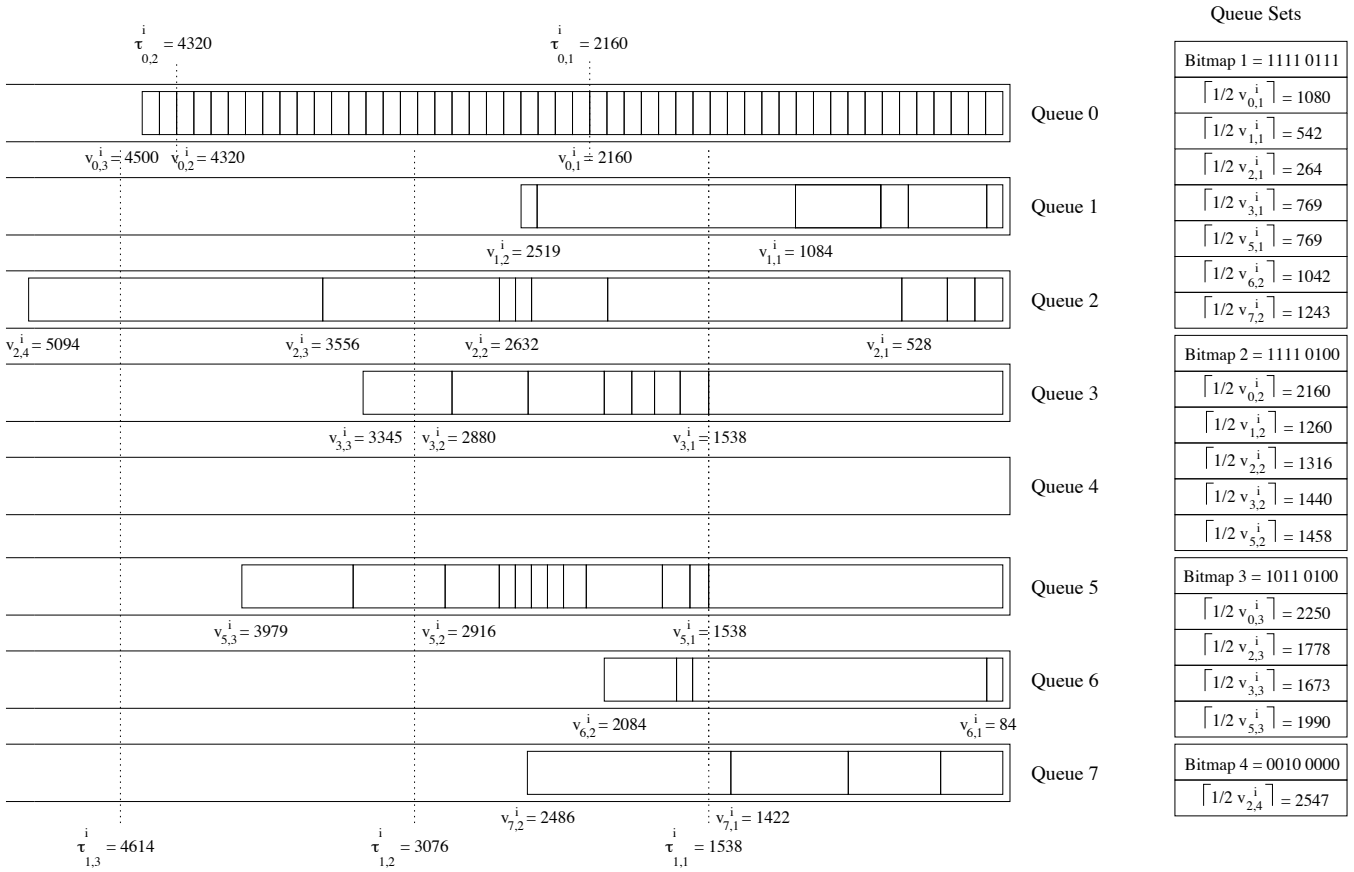


Figure 2.4: Example: Constructing a REPORT message, left: the contents of the priority queues residing at ONU i, right: the resulting Queue Sets and Queue Reports in the REPORT message

For queue 1 two QRs have to be added, whose values equal 542 and 1260 (a use is made of the first 2 bitmap fields, introduced for queue 0). Hence, a total of 13 bytes is used to report the values of queue 0 and 1. Similarly, adding the QRs for queue 2 requires 9 bytes, being 4 QRs and 1 additional bitmap field. The same procedure is applied to queues 3, 4 and 5. For queue 6, 2 QRs have to be added. However, a total of 34 bytes was used to report the contents of the first 6 queues, therefore, only 5 bytes are left (as the total size of all queue sets should be at most 39). Two of the 5 remaining bytes are reserved for queue 7, because there has to be at least one QR in the REPORT message for each non-empty queue. As a result, we can only add a single QR, whose value equals 1042 for queue 6. Finally, the remaining 3 bytes allow to include a single QR with a value equal to 1243 to represent the contents of queue 7.

The procedure used in the previous example can be generalized to the following 3 step algorithm to generate a REPORT messages:

- STEP 1: ONU i generates, for each queue j, a set of 13 values $v^i_{j,l} = \max_n\{\beta^i_j(n)|\ \beta^i_j(n) \leq \tau^i_{j,l}\}$. Next, define the set V^i_j as $\{v^i_{j,l}|\ 0 \neq v^i_{j,l} \neq v^i_{j,l-1}\}$ and let $\theta^i_j = 1$ if V^i_j is not empty and let $\theta^i_j = 0$ otherwise. For instance, in our example $V^i_2 = \{528, 2632, 3556, 5094\}$. Ideally, ONU i would like to include a QR for each value $v^i_{j,l}$ in V^i_j for all queues j. However, due to the limited size of a REPORT message, this is often impossible. As a result, a selection has to be made.

- STEP 2: Keeping in mind that ONU i has to include at least one QR for each queue j for which the set V^i_j is not empty, the ONU i includes at most $x = \lfloor(39 - 2\sum_{j>0}\theta^i_j)/3\rfloor$ QRs for queue 0 (2 bytes are used for the QR, 1 for the corresponding bitmap). Hence, ONU i will include $n^i_0 = \min(x, |V^i_0|)$ QRs for queue 0, where $|V|$ is used to denote the number of elements in a set V.

- STEP 3: Denote n^i_j as the number of QRs that are included for queue j (by ONU i). After creating the QRs for the queues $0, \ldots, j-1$ there are at most

$$y = 39 - 2\sum_{k<j} n^i_k - \max_{k<j} n^i_k - 2\sum_{k>j}\theta^i_k$$

  bytes left for queue j. Indeed, the first sum represents the size of all the QRs (for queue 0 to $j-1$), the third term is the maximum number of bitmaps used so far and the last term is the amount of bytes reserved for the remaining queues. For instance, in the example there are $y = 39 - 30 - 4 - 2 = 3$ bytes for queue $j = 6$. If $z = \min(|V^i_j|, \lfloor y/2\rfloor) \leq \max_{k<j} n^i_k$, meaning that there is no additional bitmap

required for the QRs of queue j, then ONU i will generate $n_j^i = z$ QRs for queue j. Otherwise, it generates

$$n_j^i = \max_{k<j} n_k^i + \min(|V_j^i| - \max_{k<j} n_k^i, \lfloor (y - 2 \max_{k<j} n_k^i)/3 \rfloor)$$

QRs for queue j (the minimum in this term reflects the number of new bitmap fields required).

Finally, the n_j^i QRs for queue j included by ONU i hold the $n_j^i - 1$ smallest values in the set V_j^i and the maximum value in this set. For instance, in the example the QR included the values 1042 for queue 6 and 1243 for queue 7 and not 42 and 711, respectively.

ONUs transmit their REPORT message using the last 84 bytes of the TW, thus the reported values represent the state of the queues at the end of the TW, i.e., the necessary time for the ONU to construct such a report is not taken into account. The ONU takes the IPG and the preamble for each packet into account when reporting (this is also the case in the example).

2.4 Upstream Bandwidth Allocation Algorithms

In the following sections upstream bandwidth allocation (UBA) algorithms are introduced, which utilize the threshold reporting information. In EPON the UBA algorithm is defined by the scheduling mechanisms at the OLT and the ONUs. The scheduling mechanism at the OLT is cycle based, where a cycle is defined as the time that elapses between 2 "executions" of the scheduling algorithm. A cycle has a variable length confined within certain lower and upper bounds, which are denoted by T_{min} and T_{max} (sec), meaning that the algorithm schedules between B_{min} and B_{max} (bytes) at a time, where B_i is found by multiplying T_i by the line rate. During each cycle each ONU is granted exactly one TW and each registered ONU is forced to send a REPORT message during its TW, thus, even if an ONU reported nothing to the OLT, it is granted a TW by the OLT that is sufficiently large for one REPORT message. Thus, the number of bytes that the OLT needs to schedule is bounded by $\hat{B}_{min} = B_{min} - N(84 + g_b)$ and $\hat{B}_{max} = B_{max} - N(84 + g_b)$ bytes (recall, a REPORT requires 84 bytes), where N is the number of registered ONUs and g_b the guard time expressed in bytes, i.e., g times the line rate (see Section 2.1). An execution of the scheduling algorithm produces a set of ONU assignments a_i, where a_i indicates the number of bytes that ONU i is allowed to transmit in its TW during the next cycle. The length of the TW for ONU i is set to $w_i = a_i + 84 + g_b$ (bytes).

The REPORT messages used by the OLT to schedule cycle $n+1$ are exactly those that were received during cycle n. Now, due to the distance between the OLT and the ONUs, it should be clear that the REPORT message of some ONUs might not reach the OLT before it executes its algorithm. The execution starts some time before the start of the $n+1$ cycle such that there is enough time left for the GATE messages that result from executing the algorithm, to reach the most distant ONU before the start of cycle $n+1$ (which coincides with the end of cycle n). After all,the first TW of cycle $n+1$ could be assigned to this most distant ONU. This in its turn implies that the ONUs scheduled at the end of a cycle are somewhat disadvantaged. Therefore, the ONU order within a cycle is random. Numerical experiments have shown that this causes somewhat higher delays, but does guarantees the same treatment of all ONUs.

The starting time s_i of the TW of the i-th ONU in cycle $n+1$, for $i = 1, \ldots, N$, is found as $s_{i-1} + w_{i-1}/(\text{line rate})$, where s_0 represents the end of cycle n. Before setting the starting time in the GATE messages, the s_i values are recalculated based on the knowledge of the round-trip times of each ONU to represent their local time.

2.4.1 Processing of the REPORT Messages at the OLT

The OLT maintains a table with information about the state of the queues at each ONU. This table contains the fields $r^i_{j,l}$, for ONU $i = 1, \ldots, N$, queue $j = 0, \ldots, P-1$ and $l = 1, \ldots, 13$ (recall, 13 thresholds are used, as at most 13 QRs for the same queue j can be supported by a REPORT message): These fields are updated whenever the OLT receives a REPORT message (from ONU i) as follows:

- STEP 1: The OLT starts by clearing the $13P$ fields $r^i_{j,l}$. Next, it scans the REPORT message and whenever it encounters a QR that corresponds to queue j, it enters twice (due to the granularity of the QR values) the value v found in this QR in $r^i_{j,x}$, where x is the smallest index such that $v \leq \tau^i_{j,x}$.

- STEP 2a: If the field $r^i_{j,13}$, corresponding to the infinity threshold, is non-empty and if $y < 13$ is the largest index y for which $r^i_{j,y}$ is non-empty, then $r^i_{j,l} = \tau^i_{j,l}$ for all $y < l < 13$.

- STEP 2b: If the field $r^i_{j,13}$ is empty and if y is the largest index y for which $r^i_{j,y}$ is non-empty, then $r^i_{j,l} = r^i_{j,y}$ for all $y < l \leq 13$.

- STEP 3: Finally, the OLT will increase all $13(P-1)$ entries $r^i_{j,l}$, for $j = 1, \ldots, P-1$ and $l = 1, \ldots, 13$, by $\sum_{k<j} r^i_{k,13}$.

This procedure guarantees that $r^i_{j,l}$ contains the size of all the packets that were reported in the last REPORT message of ONU i for the queues $k < j$ as well as the first n packets of queue j, where n is the maximum number for which $\beta^i_j(n) \leq \tau^i_{j,l}$. Recall $\beta^i_j(n)$ was defined as the size of the first n packets in queue j of ONU i. Thus, $r^i_{P-1,13}$ represents the total number of bytes reported by ONU i. For instance, in the example of Figure 2.4, $r^i_{2,3} = 2 * (2250 + 1260 + 1778) = 10576$ bytes.

2.4.2 Full Threshold Level (FTL) Scheduling at the OLT

The proposed algorithm, as the name reveals, allocates bandwidth up to the same threshold for all requesting ONUs in the cycle. The scheduling at the OLT is based on the table with the $r^i_{j,l}$ fields (see Section 2.4.1). The OLT constructs the GATE messages for cycle $n+1$ as follows. First, if the REPORT message of ONU i (transmitted in cycle n) did not reach the OLT before the execution time of the algorithm (because its TW was located near the end of cycle n, see Section 2.4), then $r^i_{j,l} = 0$ for all j and l. Next, the OLT computes the following sums:

$$R_{j,l} = \sum_i r^i_{j,l}, \tag{2.1}$$

for all j and l. Notice, $R_{j,l}$ represents the amount of bandwidth requested by all ONUs for priority $p < j$ traffic and priority j traffic up to threshold l. Thus, $R_{j,l} \leq R_{k,m}$ if $j < k$ or if $j = k$ and $l \leq m$. For simplicity of the notation let $R_{tot} = R_{P-1,13}$. The amount of bandwidth a_i allocated to ONU i depends in the following manner on R_{tot}:

1. $R_{tot} < \hat{B}_{min}$: In this case the assignment lengths a_i of the ONUs are the amount they have requested (i.e., $r^i_{P-1,13}$) plus a fair share of the remaining amount of bandwidth up to $\hat{B}_{min}$ (i.e., $(\hat{B}_{min} - R_{tot})/N$).

2. $\hat{B}_{min} \leq R_{tot} \leq \hat{B}_{max}$: In this case the ONUs are assigned exactly the amount of bytes they have requested, $a_i = r^i_{P-1,13}$.

3. $R_{tot} > \hat{B}_{max}$: The scheduler now has to find the largest index l and queue j for which $R_{j,l} \leq \hat{B}_{max}$ starting from the queue with the highest priority. (i) If $l+1 \neq 13$, the assignments for the ONUs are equal to what they have reported up to this queue and threshold, i.e., $a_i = r^i_{j,l}$. An appropriate choice of the threshold values $\tau^i_{j,l}$ can in this particular case guarantee that $R_{j,l} \geq \hat{B}_{min}$.

   (ii) If, on the other hand, $l + 1 = 13$, the algorithm starts by setting $a_i = r^i_{j,l}$ and $A = \sum_i a_i$. Next, a_i is incremented in an iterative manner as long as $A \leq \hat{B}_{max}$ as

follows. Let $x_i = r^i_{j,13} - a_i$, then increment a_i by $\min(x_i, FS)$, where the fair share FS equals $(\hat{B}_{max} - A)/N_r$ and N_r equals the number of ONUs for which $x_i > 0$. This simple iteration distributes the remaining bandwidth $\hat{B}_{max} - R_{j,l}$ in a fair manner between the ONUs that requested more than $r^i_{j,l}$ bytes in such a way that $a_i \leq r^i_{j,13}$.

The distinction between case 3(i) and 3(ii) is made because $\tau^i_{j,12}$ will, in general, be much smaller than the buffer size of queue j.

2.4.3 Scheduling at the ONU

Two typical types of scheduling at the ONU are considered:

Full Priority (FP) Scheduling

By this scheduling algorithm the strict priority scheduling scheme is meant, i.e., the packets are sent according to their priority in a TW. Thus if there is a packet at the ONU with priority $p : p < q$ it will be transmitted in the TW before the packet with priority q regardless of whether it was present in the queue when the request for this TW was transmitted. The disadvantage of this scheme is that a substantial amount of time elapses between the transmission of the REPORT message and the start of the corresponding TW, meaning that the content of the queues is likely to change during this time interval. For instance, if some priority p packets arrived during this interval, these arrivals will destroy the usefulness of the threshold reporting for all the priority $q > p$ traffic (that is, the lower priority traffic). Moreover, higher priority packets may consume some of the bandwidth requested by low priority traffic, thereby postponing its transmission. This is especially true for lightly loaded ONUs. This effect, where the delay of higher priority traffic increases when the load is decreased has been identified as the "light-load penalty (see [34]).

Interval Priority (IP) Scheduling

In this scheduling scheme the ONU remembers the total number of bytes (per queue) that it reported in last REPORT message, i.e., the REPORT transmitted during the last cycle, and it transmits the reported data first. Thus, if higher priority traffic arrived meanwhile, it has to wait until the reported lower priority traffic is transmitted. If the TW is larger than the reported queues' content it continues sending packets according to the FP scheduling scheme. This means that up to certain granularity the ONU respects

also the arrival time of the packets. It can be considered as corresponding to the coloured grants in an ATM PON (APON) [99] where all the scheduling is done at the OLT site and where the ONU indicates the priority for which a given cell is destined. However the main interest here is in the interaction of the IP scheduling with the threshold reporting mechanism. Results presented in Section 2.7.3 demonstrate that combining IP scheduling with the thresholds can result in substantial efficiency gains.

2.5 Rate-Based Scheduling

To achieve a better performance for time critical applications which have a constant bit rate (CBR), e.g., voice, it would be preferable to assign the CBR bandwidth to the ONUs according to the rate of these applications, avoiding the need for the ONUs to report the status of the highest priority queue (cfr. [100]). Such a mechanism requires the possibility in EPON to establish "a connection", however, Ethernet is a connectionless protocol. Nevertheless, given the importance of such a mechanism it is reasonable to expect that a way to report or estimate the rate of a CBR application should and will be standardized.

A rate allocation scheme, named the CBR credit scheme, that also assumes that the OLT is aware of the rates and packet sizes of the existing CBR applications, was proposed and studied in [34]. The proposed rate-based scheduling explores what the performance of a CBR application would be if the rate r_{CBR}, expressed in packets/s, and packet size s_{CBR} of each CBR application is known by the OLT. In this section the rate-allocation scheme is described. It also describes a way to incorporate the information on the application rate and packet size into the scheduling algorithm from Section 2.4.

The idea behind rate-based scheduling is to predict the number of packets n_{CBR} that arrive at the ONU (from each CBR application) during the time interval spanned by two consecutive TWs W_1 and W_2. For doing so, the formula provided in [34, Equation 5] is used:

$$n_{CBR} = \left\lceil \frac{t_s + h/C - t_r}{p_{CBR} - s_{CBR}/C} \right\rceil, \tag{2.2}$$

where p_{CBR} is the period of the CBR application and can be obtained as the reciprocal of the rate expressed in packets per second, h is the amount of bytes in W_2 allocated to the non CBR traffic, C is the link capacity or the line rate, t_s the starting time of W_2 and t_{r1} the time stamp in the REPORT message transmitted during W_1 (see Figure 2.5).

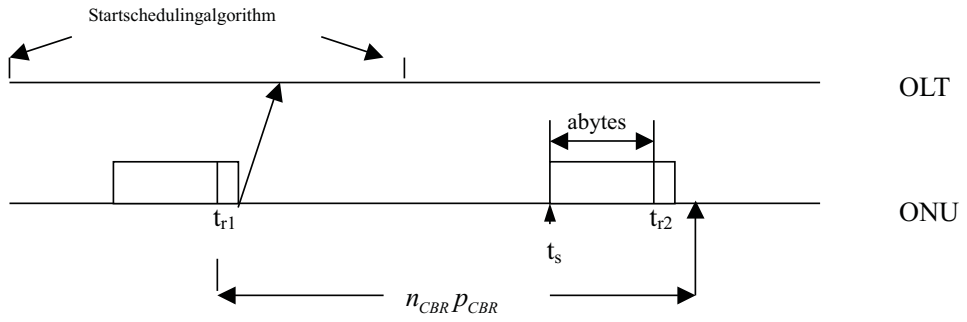


Figure 2.5: Calculation of the number of packets n_{CBR} that arrive during an interval spanned by two TWs

Knowing n_{CBR} for each CBR application of ONU i, the OLT computes the number of bytes a_i allocated to ONU i as h plus the total amount of bytes b^i_{CBR} required for the CBR traffic of ONU i. The value b^i_{CBR} clearly depends on the TW's starting time t_s, which is not known before executing the scheduling algorithm (except for the first ONU in the cycle). Moreover, $\sum_i a_i$ should be smaller than or equal to $\hat{B}_{max}$. To account for this, first the maximum number of bytes $\hat{B}_{max}$ to be scheduled is reduced to $\bar{B}_{max} = \hat{B}_{max} - B_{CBR}$, where $B_{CBR} = \sum_i b^{i,max}_{CBR}$ and $b^{i,max}_{CBR}$ equals the amount of bytes required for the CBR traffic at ONU i assuming that the distance between the two consecutive TWs is maximal, i.e., $2T_{max}$.

In conclusion, first the algorithm with $\bar{B}_{max}$ is evoked instead of $\hat{B}_{max}$ to find, for each ONU, the amount of bandwidth allocated to the non CBR traffic, afterwards we add b^i_{CBR} to find a_i starting with the first ONU in the cycle. Indeed, the end of the TW of the $(i-1)$-th ONU in a cycle determines the starting time t_s of the i-th ONU.

It is important to keep in mind that the CBR traffic is typically transmitted first in a TW when the rate-based scheme is used, independent of the scheduling algorithm implemented at the ONU. Thus, if an ONU uses IP scheduling, it will first transmit the CBR traffic, followed by (a part of) the reported data and finally (if there is still some room left) by some unreported data.

2.6 Threshold Assignment

There is no standard way to convey the thresholds to the ONUs. There are several possibilities: at the registration of an ONU, this would result in static thresholds, or in some of the Operation and Management (OAM) messages, providing the possibility for

dynamic thresholds, e.g., threshold values that dependent on the number of currently registered ONUs in the network. The number of thresholds per queue may influence the way in which the thresholds are conveyed to the ONUs, e.g., if an ONU has 8 priority queues each having 13 thresholds that require regular updates, several Mbit/s are needed to keep the thresholds up to date.

Our proposal is that for each queue j of ONU i only one threshold is assigned (being $\tau^i_{j,1}$) and all others are derived from it. As such, the assignment can be static or dynamic without requiring a lot of bandwidth. A simple and logical way is to use linear dependence

$$\tau^i_{j,l} = l\ \tau^i_{j,1}, \tag{2.3}$$

for $l < 13$. As explained in Section 2.4, the maximum number of QR values in a single REPORT message is 13 so there is no point in having more than 13 thresholds for a queue. In order to provide the OLT with a complete view of the bandwidth requirements of an ONU, $\tau^i_{j,13}$ equals infinity (which is equivalent to setting $\tau^i_{j,13}$ equal to the ONU buffer size).

2.7 Performance Evaluation of FTL Scheduling Algorithms

A variety of simulation results, which evaluate the performance of the different upstream scheduling algorithms, is presented within this section. The algorithms differ in the scheduling algorithm used at the ONU (FP/IP) and the OLT policy regarding CBR traffic. Hereafter, the four scheduling algorithms are described:

- Full priority - full threshold level scheduling (FP-FTL):
  The FP-FTL scheduling algorithm combines full priority scheduling at the ONU with full threshold level scheduling at the OLT. The biggest advantage of this discipline is that no intelligence at the ONU is required. This is a typical requirement from equipment manufacturers, which try to lower the cost of the equipment at the customer premises. However as was pointed out in Section 2.4.3 with this type of scheduling lower priorities might starve.

- Interval priority - full threshold level scheduling (IP-FTL):
  This algorithm improves on FP-FTL by using Interval Priority (IP) scheduling at the ONU. This increases the complexity of the algorithm at the ONU site but lower priorities cannot starve due to discrepancies between the requested and the

transmitted traffic by the ONU. The disadvantage is that lower priority traffic can be transmitted before high priority traffic, provided that the latter was not reported. This might result in higher delay for the high priority traffic which for some applications, like voice, which are usually mapped to this priority, is not desirable.

- Full priority - full threshold level scheduling with rate estimation (FP-FTLr): This algorithm improves on FP-FTL by using scheduling with rate estimation at the OLT. Estimating the rate of the highest priority traffic would reduce the starvation of traffic with lower priority. However, it will not avoid it. Except traffic with priority on an active level closest to the highest priority, all the other traffic will be prone to starvation.

- Interval priority - full threshold level scheduling with rate estimation (IP-FTLr): This algorithm adds rate estimation scheduling at the OLT to the IP-FTL algorithm. One would expect that this would improve the delay of the highest priority traffic and thus actually resolving both big lacks of the above described algorithms, namely the need to request bandwidth for the highest priority traffic and all consequences from this and the starvation of the lower priority traffic.

2.7.1 General Setup

The aim is to compare the performance of the four scheduling algorithms for upstream bandwidth allocation in EPON described above. To do so, an EPON system with $N = 32$ ONUs, each at a randomly chosen distance between 0.5 and 20 km, is simulated. Also, each ONU supports 3 priority queues, the size of which is 8 Mbits. The line rate or link capacity C between the OLT and the ONUs is 1000Mb/s and the rate at which Ethernet packets (IPG included) are generated at the ONUs is 100Mb/s. The guard time g between two consecutive TWs is 1μs. The time required to execute the algorithm (as well as to generate the GATE messages from the results of the execution) is assumed to be 0.1 msec.

The total data load ρ_d is varied from 0.16 up to 0.96 of the link capacity C which is calculated only on the base of the Ethernet frames (without preamble and IPG). Notice, due to the preamble, IPG and guard time g, the actual load ρ on the uplink channel will be considerably higher. The load ρ_d is equally distributed between all ONUs, which results in an ONU data rate between 5 and 30 Mb/s. All ONUs have equal traffic parameters. The traffic for priority 0 is simulated as a CBR arrival process as described

in section 2.5. The amount of CBR traffic is identical for all simulations. Thus, the load ρ_d is increased by adding more priority 1 and 2 traffic, while keeping the amount of CBR traffic fixed. For the traffic source of the two other priority classes, being priority 1 and 2, the VBR arrival process as described in the following section 2.7.2 is used. The data load for the priority 1 and 2 traffic is chosen to be equal.

The cycle length varies between $T_{min} = 0.5$ ms and $T_{max} = 1.5$ ms, meaning that $B_{min} = 62500$ bytes, $\hat{B}_{min} = 55812$ bytes, $B_{max} = 187500$ bytes and $\hat{B}_{max} = 180812$ bytes, unless otherwise stated (see Sections 2.4 and 2.5 for definitions). The thresholds are chosen as follows: $\tau^i_{0,1} = 2160$ and $\tau^i_{1,1} = \tau^i_{2,1} = 1538$ bytes for all i, unless otherwise stated. Setting $\hat{B}_{max} - \hat{B}_{min} > 2\tau^i_{j,1}$ guarantees that case 3(i) of the scheduling algorithm presented in Section 2.4.2 generates a cycle length between B_{min} and B_{max}. The other thresholds $\tau^i_{j,l}$ are obtained from these as explained in Section 2.6.

2.7.2 Arrival Processes

Two types of arrival processes are used as traffic sources in the simulations. This section describes them and their properties.

CBR source arrival process

The CBR traffic source is simulated as a stream with a constant rate of 8000 packets/s and packet size s_{CBR} of 70 bytes. It is chosen so as to emulate a T1 connection with a UDP/IP/Ethernet protocol stack.

VBR source arrival process

As a VBR arrival process a two state Conditioned Markov-Modulated Bernoulli Process (C-MMBP) as described in [101] is used. It is a discrete-time Markov process and belongs to the class of D-MAPs [102]. In each of the states of the C-MMBP there is a given probability for generating a packet. The state can be switched only on condition that a packet was generated. The process is characterized by a rate vector (α, β), that contains the mean arrival rates associated with each of the two states and a $2x2$ transition matrix that governs the transition probabilities between the states. The transition matrix is written as the sum of two matrices P and A. The matrix elements of P, i.e. $P_{i,j}$, equal the probability that an arrival occurs and a transition from state i to j takes place. The diagonal matrix elements of A, $A_{i,i}$, equal the probability that no arrival occurs in state i. They are given by

$$\text{P}= \begin{pmatrix} \alpha(1-p) & \alpha p \\ \beta q & \beta(1-q) \end{pmatrix} \quad \text{and} \quad A = \begin{pmatrix} (1-\alpha) & 0 \\ 0 & (1-\beta) \end{pmatrix},$$

where α and β are the probabilities of packet arrival for the two states and p, q are the probabilities for changing the state. The mean packet interarrival time is given by

$$\mu = \frac{\alpha p + \beta q}{\alpha\beta(p+q)}$$

and its variation is given by

$$VAR = 2 * \frac{\alpha^2 p + \beta^2 q}{\alpha^2\beta^2(p+q)} - \mu - \mu^2$$

The arrival rate in state 1 depends on the load ρ_d and is 5 times as high as in state 2. Also, the mean sojourn time in state 1 and 2 depends on the load ρ_d and lies within $[9.4, 417.3]$ msec and $[188.5, 8346]$ msec, respectively. Thus, the background sources can be considered as bursty and strongly correlated. The packet size distribution is based on real data traces from the Passive Network and Analysis (PNA) project conducted by the National Laboratory for Applied Network Research (NLANR) [103] and the probability distribution function is shown in Figure 2.6. There are four distinct peaks at 64, 552, 576 and 1518 bytes. The mean Ethernet frame size of this distribution is 455.7 bytes.

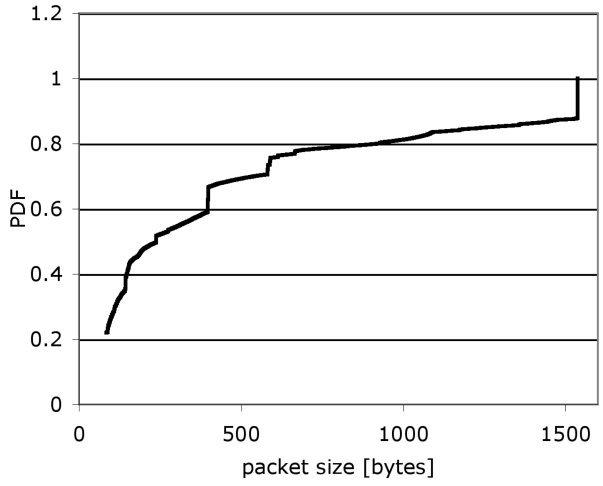


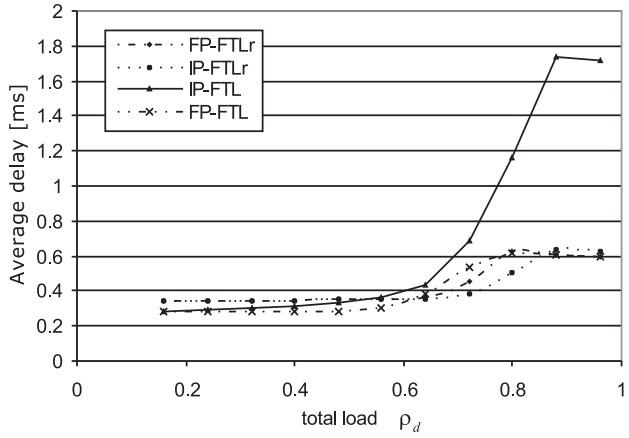
Figure 2.6: Probability distribution function (PDF) of the packet sizes

2.7.3 Numerical Results

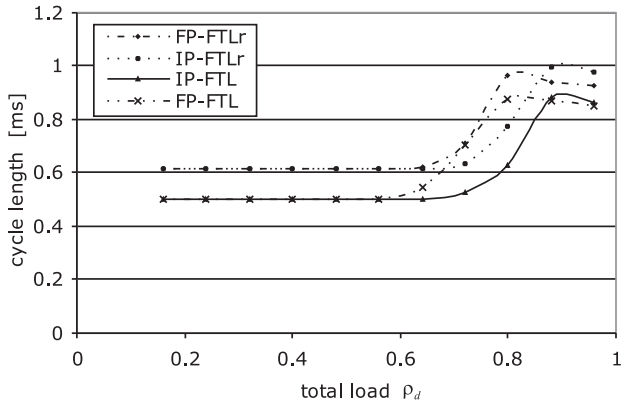
The average queuing delay of the priority 0 i.e., CBR traffic, as a function of the data load ρ_d is presented in Figure 2.7(a) for each of the four algorithms. Next, these results are discussed in detail, starting with FP-FTLr and IP-FTLr, i.e., the algorithms that use the rate-based scheduling algorithm. First, the average delay remains nearly constant as $\rho_d < 0.6$. This is easily explained by Figure 2.7(b) that indicates that the cycle length is constant and equal to the minimum length. Note that the minimum is more than 0.5 because $\hat{B}_{min}$ bytes are allocated to the priority $p > 0$ traffic on top of the bandwidth computed by the rate-based scheme for the CBR traffic. Therefore, the average delay is approximately half the cycle length. The delay variation of the priority 0 traffic also follows this behavior as seen in Figure 2.8(a). Both the mean delay and the variation start to increase around $\rho_d = 0.65$ and 0.7 for FP-FTLr and IP-FTLr, respectively. This is exactly the point where the mean cycle length starts to increase. The fact that this happens at a higher load for IP-FTLr is explained by the data throughput results shown in Figure 2.8(b). This figure indicates that the fragmentation losses, that is, the amount of bandwidth that is lost due to not fragmenting the frames, are much smaller for IP-FTLr (due to the combination of the threshold reporting and IP scheduling), meaning that more data fits into a minimal length cycle. A discussion on the losses considered is given later on in this section.

As the load ρ_d further increases (beyond 0.8 and 0.9, resp.), a slight decrement in the delay and delay variation is observed, which is in correspondence with the cycle length behavior (see Figure 2.7b). In this load region, $R_{tot} > \hat{B}_{max}$ (see Section 2.4.2), with ρ_d increasing, the mean number of ONUs that request bandwidth (of priority $p > 0$) also increases, causing a shorter mean cycle length. Again, due to the better efficiency of IP-FTLr compared to FP-FTLr, a higher data load ρ_d is needed to reach the maximum cycle length. The higher variation for ρ_d large is caused by the burstiness of the low priority traffic (that determines the boundaries of the TWs).

The FP-FTL algorithm behaves for priority 0 much like the rate-based algorithms. It has the smallest average packet delay and delay variation in the region $\rho_d < 0.6$ which can be confirmed with the corresponding average cycle lengths. In the region $0.64 < \rho_d < 0.8$ the cycle length and the delay increase to reach their maximum values at $\rho_d = 0.8$ For IP-FTL a slight increment in the average queueing delay for ρ_d in the $[0, 0.6]$ area is observed, although the cycle length is also minimal in this region. The increasing delay can be explained as an effect caused by the IP scheduling algorithm as follows. In low load situations, an ONU generally gets a TW that is larger than



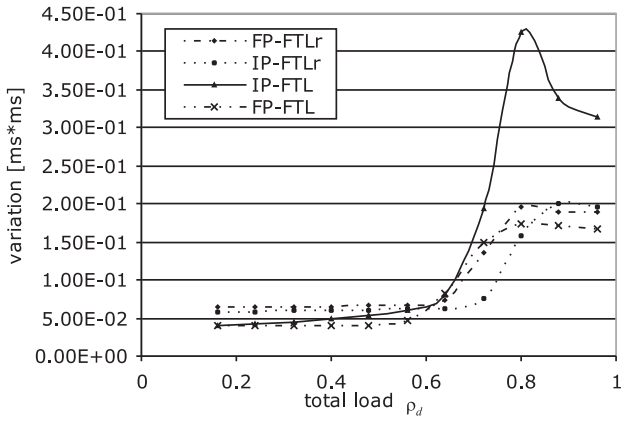
(a)



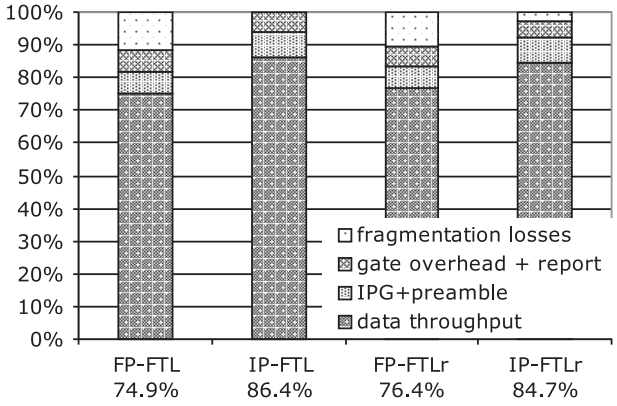
(b)

Figure 2.7: (a) The average queueing delay of the priority 0 traffic and (b) the average cycle length $E(CL)$ as a function of ρ_d.

the amount of bandwidth requested. As a result, some (or all) of the data that arrived since the transmission of the last REPORT message, can also be transmitted in the TW. For very low loads this actually causes ONUs to report 0 bytes for all the queues (as reporting happens at the end of the TW), therefore the IP scheduling reduces to FP (see Section 2.4.3) and all the CBR traffic is transmitted first. However, as the load ρ_d increases some of the newly arrived low priority traffic will not be able to use the TW anymore and therefore the ONU will report some low priority traffic. The IP scheduling algorithm will transmit this reported low priority data before the CBR traffic; hence, the CBR traffic shifts to the back of the TW as the load increases. This explains the slight increment in the mean delay. Also, due to the burstiness of the low priority



(a)



(b)

Figure 2.8: (a) The delay variation of the priority 0 traffic as a function of ρ_d. (b) The bandwidth utilization at high loads ρ_d. The numbers bellow the graphic indicate the percentage of the total bandwidth used for data.

traffic, a slight increment in the delay variation is observed. If the load ρ_d is further increased (beyond 0.6), some of the CBR traffic will no longer be able to use the TW, causing an additional delay of one cycle. Thus, more and more CBR traffic will suffer this additional delay until eventually all CBR traffic suffers a delay of approximately 1.5 cycles (explaining the strong increase in the mean delay and the delay variation). The peak in the delay variation at 0.8 corresponds to the situation where about half of the CBR traffic uses the first TW after being generated and the other half uses the second TW (this can be seen from the mean delay curve, because the data point for $\rho_d = 0.8$ is located halfway between the two high load plateaus). Finally, the efficiency obtained

by IP-FTL is even higher compared to IP-FTLr, because with IP-FTLr you only have a good estimation of the CBR traffic present in the ONU and not the exact value. It is clear that IP-FTL pays a high price in terms of the delay to achieve this minor efficiency improvement. Notice, in Figure 2.8(b) the maximum cycle length $T_{max} = 1$ ms for IP-FTL (compared to 1.5 ms for IP-FTLr), which cancels the slight efficiency gain that IP-FTL has on IP-FTLr.

Finally before proceeding further lets outline the different components of the bandwidth utilization results shown on Figure 2.8(b). The average amount of the $fragmentation\ losses$ per second, F, represents the bandwidth lost due to not fragmenting the Ethernet frames, i.e. when not an exact number of packets fit in a TW, a certain number of bytes remains unused. The average $gate\ overhead+report$ losses per second, G, are due to the overhead, which is taking into account laser turn on and off time and guard time, and also the bandwidth used for report messages. The average $IPG\ and\ preamble\ losses$ per second, I, are the losses due to the IPG and the packet preamble. The average $data\ throughput$ per second T is the bandwidth used to transmit Ethernet frames (without the ones which are encapsulating REPORT messages). Thus the data throughput can be expressed as

$$T = C - (F + G + I), \tag{2.4}$$

where C is the link capacity or line rate. The $IPG\ and\ preamble\ losses$ are constant per packet or 20 bytes per packet -12 for the IPG and 8 bytes for the preamble. The average packet overhead can be calculated from the packet size distribution and is given by

$$av.packet\_overhead = \frac{\sum B(s) * f(s) - \sum s * f(s)}{\sum B(s) * f(s)} \tag{2.5}$$

Here s is the payload, $f(s)$ is the payload distribution function and $B(s)$ is the encapsulated payload. The expression in the denominator gives simply the average packet size with encapsulation $E(B)$. Taking into account that $B(s) = 20 + s$ Equation (2.5) is reduced to

$$av.packet\_overhead = \frac{20}{E(B)} * 100\% = \frac{20}{20 + E(s)} * 100\% \tag{2.6}$$

and it gives the average percent which is lost for IPG and preamble per packet. Assuming that all the available bandwidth is used, the total average IPG and preamble overhead

per second is obtained from

$$I = \frac{20}{E(B)} * (C - F - G). \tag{2.7}$$

The fragmentation losses depend on the bytes that remained unused in a TW or the "remainder" R. To obtain the average fragmentation losses per second the average remainder for all ONUs n with granted TW in a cycle is divided to the average cycle length $E(CL)$, which gives

$$F = \frac{n * (E(R))}{E(CL)}. \tag{2.8}$$

In the simulated algorithm each ONU may transmit exactly one TW and one REPORT message in a cycle. This gives for G

$$G = \frac{N * (84 + g_b)}{E(CL)}[bytes/s] = \frac{N * (84 + g_b) * 8}{E(CL)}[bits/s]. \tag{2.9}$$

Recall that the REPORT message size is 84 bytes and that $g = 1\mu s$, thus $g_b = 125bytes$.

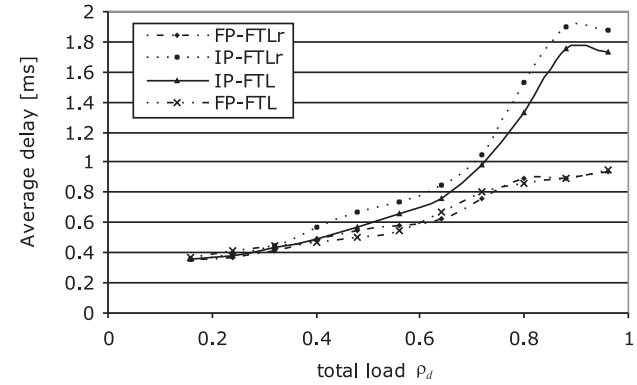
Substituting the expressions (2.7), (2.8) and (2.9) into (2.4), one obtains for the data throughput

$$T = \frac{E(s)}{E(B)}(C - F - G) = \frac{E(s)}{E(B)}(C - \frac{n * E(R) + N(84 + g)}{E(CL)}) \tag{2.10}$$

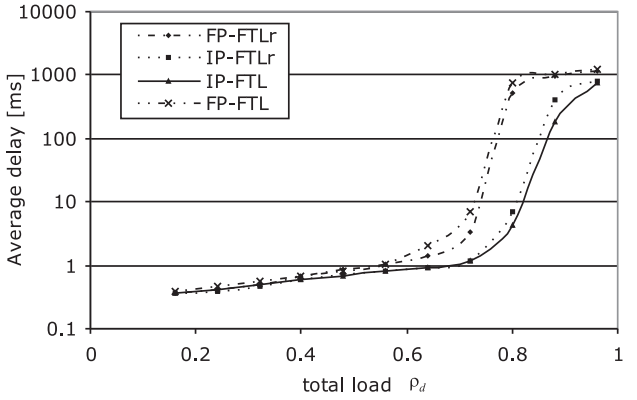
Note that the utilization shown in the Figure 2.8(b)in percentage is calculated from the data throughput

$$U = (1 - \frac{(F + G + I)}{C}) * 100\%. \tag{2.11}$$

If rate based scheduling is added to FP-FTL the efficiency increases slightly (from 74.9% to 76.4%)due to the increased cycle length (cf. Figure 2.8(a)). From the expression for the data throughput Equation (2.10) one can see that higher $E(CL)$ results in higher throughput. Adding rate based scheduling to IP-FTL decreases slightly the efficiency (from 86.4% to 84.7%), which is due to fragmentation losses. With the rate-based scheduling only an estimation of the needed bandwidth for priority 0 is made, thus there will always be some losses. Even though the cycle length is less for IP-FTL the resulting loss of efficiency is a lot less than the contribution of the fragmentation losses to the data throughput of IP-FTLr. Both IP based schedulers achieve higher data throughput



(a)



(b)

Figure 2.9: The average queueing delay for (a) priority 1 and (b) priority 2 traffic.

than the FP based ones due to the lower fragmentation losses (which for IP are ≈ 0).

The average queueing delay for priority 1 and 2 traffic is represented in Figure 2.9. The mean delay of the priority 1 traffic is substantially higher for IP-FTLr and IP-FTL compared to FP-FTLr. This is an obvious consequence of the IP scheduling, which transmits reported priority 2 traffic before unreported priority 1 traffic. Moreover, in high load situations, IP scheduling forces some priority 1 traffic to wait an additional cycle due to earlier reported priority 2 traffic. The priority 1 delays of IP-FTL are slightly better than those of IP-FTLr because IP-FTL may transmit priority 1 traffic before priority 0 traffic, which is never the case with IP-FTLr. For the same priority FP-FTL and FP-FTLr have almost equal delays due to the fact that they are treating packets from priority 1 in the same way - they have advantage over the packets from priority 2 and are transmitted after the priority 0 packets. With respect to the priority 2 traffic, FP-FTL achieves the worst results. Compared to the IP types of algorithms, where if reported the priority 2 packets have advantage above some higher priority

packets, in FP-FTL, even if reported, the priority 2 packets have the lowest priority. In FP-FTLr the bandwidth for the priority 0 is calculated such that only priority 1 packets are occupying the bandwidth allocated for priority 2. The strong delay increment for ρ_d in the $[0.6, 0.8]$ region is caused by the fact that the uplink channel becomes saturated. Indeed the actual load ρ on the channel is higher than the data load ρ_d due to the overhead caused by the preamble, the IPG, guard time g and possible the idle period at the end of a TW. A less efficient algorithm causes channel saturation at a lower data load ρ_d. This is confirmed by comparing Figure 2.8(b) and Figure 2.9(b). Finally, the nearly stable delay for ρ_d beyond 0.8 is caused by the finite buffers.

In this section four modifications of an upstream bandwidth allocation algorithm were compared via simulations. They differ in their inter and intra- ONU scheduling and the CBR traffic allocation. It was shown that rate based scheduling achieves better result in terms of delay. Furthermore it was shown that interval priority scheduling at the ONU achieves better results in terms of utilization. With the aim to further improve the data throughput a modification to the scheduling algorithm at the OLT is discussed in the next section.

2.8 Upstream Bandwidth Allocation Algorithms with Single Report Threshold Level (SRTL) Scheduling

As was seen in the previous section from Equation (2.10), the data throughput of the algorithm depends on the cycle length. The longer the average cycle length is the higher the data throughput will be. However as seen from Figure 2.7(b) even at higher loads the average cycle length is far bellow the maximum allowed of $1.5ms$. Aiming to increase the average cycle length at high loads an algorithm using a single report threshold level (SRTL) scheduling is proposed in this section. The amount of bandwidth a_i allocated to ONU i according to the SRTL scheduling depends on the total number of requested bytes in a cycle R_{tot} (see Section 2.4.2) in the following manner:

1. $R_{tot} < \hat{B}_{min}$: In this case the assignment lengths a_i of the ONUs are the amount they have requested (i.e., $r^i_{P-1,13}$) plus a fair share of the remaining amount of bandwidth up to $\hat{B}_{min}$ (i.e., $(\hat{B}_{min} - R_{tot})/N$).

2. $\hat{B}_{min} \leq R_{tot} \leq \hat{B}_{max}$: In this case the ONUs are assigned exactly the amount of bytes they have requested, $a_i = r^i_{P-1,13}$.

3. $R_{tot} > \hat{B}_{max}$: The scheduler now has to find the largest index l and queue j for

which $R_{j,l} \leq \hat{B}_{max}$ starting from the queue with the highest priority.

(i) If $l+1 \neq 13$, the algorithm starts by setting $A = \sum_i a_i$, where $a_i = r^i_{j,l}$. Next, the OLT considers each of the values $r^i_{j,l+1}$ for all ONUs i in a random order and sets $a_i = r^i_{j,l+1}$ if $A' = A + (r^i_{j,l+1} - r^i_{j,l}) \leq \hat{B}_{max}$ in which case A is replace by A'.

(ii) If, on the other hand, $l+1 = 13$, the algorithm starts by setting $a_i = r^i_{j,l}$ and $A = \sum_i a_i$. Next, a_i is incremented in an iterative manner as long as $A \leq \hat{B}_{max}$ as follows. Let $x_i = r^i_{j,13} - a_i$, then increment a_i by $\min(x_i, FS)$, where the fair share FS equals $(\hat{B}_{max} - A)/N_r$ and N_r equals the number of ONUs for which $x_i > 0$. This simple iteration distributes the remaining bandwidth $\hat{B}_{max} - R_{j,l}$ in a fair manner between the ONUs that requested more than $r^i_{j,l}$ bytes in such a way that none get more than they requested, i.e., $a_i \leq r^i_{j,13}$.

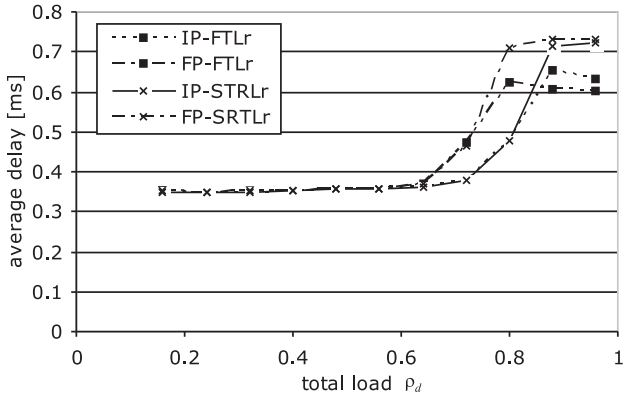
The distinction between the FTL scheduling algorithm presented in Section 2.4.2 and the above presented is in 3(i). The scheduler ends up in this case when the majority of the ONUs request a substantial amount of bandiwdth. When only few of them have a big backlog in their queues this would result in case 3(ii). In the presented algorithm the maximum difference between the maximum cycle bytes and the allocated bytes for a cycle for this case is bounded by $max(\bar{B}_{max} - \sum_i a_i) < max(\tau^i_{j,l} - \tau^i_{j,l-1})$, while for FTL scheduling $max(\bar{B}_{max} - \sum_i a_i) < (\sum_{i=1}^{N} \tau^i_{j,l} - \sum_{i=1}^{N} \tau^i_{j,l-1})$.

2.9 Performance Comparison of FTL and SRTL Scheduling Algorithms

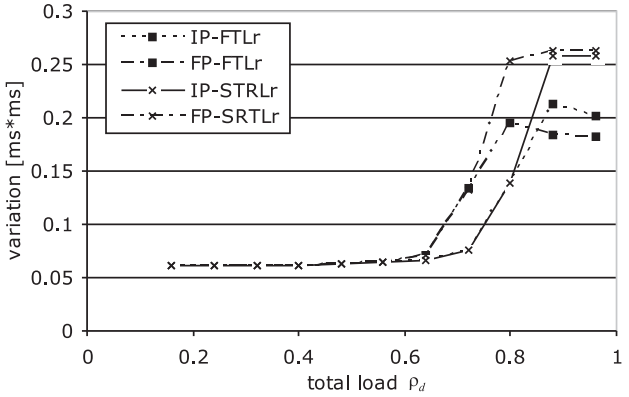
In this section the influence of two types of scheduling algorithms at the OLT on the delay, delay variation and throughput performance is compared. Next, the two considered SRTL scheduling algorithms are described:

- Full priority - single report threshold level scheduling with rate estimation (FP-SRTLr) This algorithm combines full priority scheduling at the ONU with SRTL scheduling with rate estimation at the OLT.
- Interval priority - single report threshold level scheduling with rate estimation (IP-SRTLr) This algorithm combines interval priority scheduling at the ONU with SRTL scheduling with rate estimation at the OLT.

Hereafter these two algorithms are compared with the FP-FTLr and the IP-FTLr algorithms described in Section 2.7. For the simulations the same setup and traffic profile as the one described in Section 2.7.1 is used.



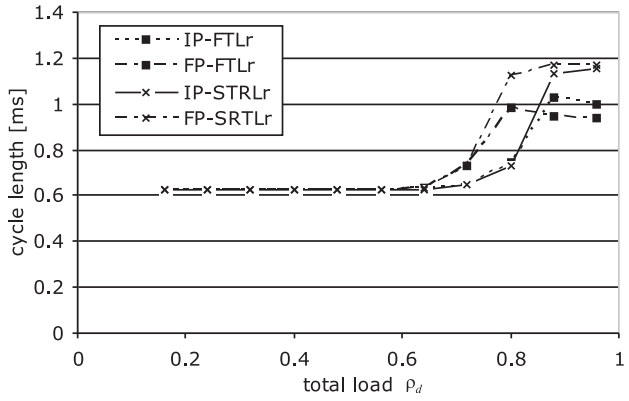
(a)



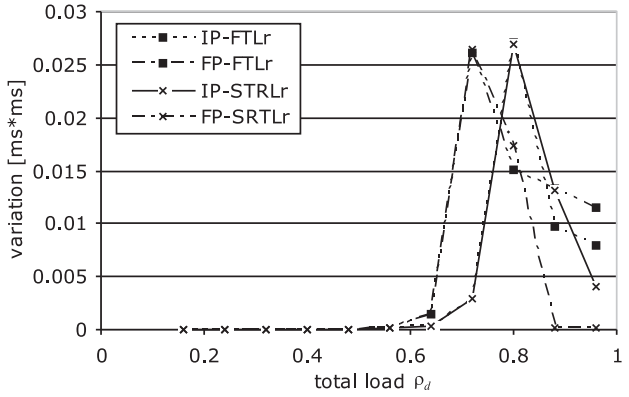
(b)

Figure 2.10: (a) the average queuing delay and (b) the delay variation of the priority 0 traffic as a function of ρ_d.

Figures 2.10 presents both the mean delay and the delay variation of the priority 0 traffic while Figure 2.11 shows the mean and variation of the cycle length. With the SLTR scheduling algorithm as described in Section 2.8, the maximum difference between B_{max} and the cycle length C is, under high load conditions, reduced from $1538N$ to $\max_{i,j}(\tau^i_{j,1})$ or 1538 bytes. The first threshold $\tau^i_{0,1}$ is not considered because on one hand in the rate-based scheduling algorithms the traffic for priority 0 is already accounted for and on the other hand, the reported traffic for this priority 0 never exceeds B_{max}. As a result, the mean cycle length of the SRTLr algorithms is larger compared to the FTLr schemes, while its variation is less (under high load conditions). This causes a higher mean delay and delay variation for the priority 0 traffic.



(a)



(b)

Figure 2.11: (a) average cycle length and (b) cycle length variation as a function of ρ_d.

Further, with the FTLr algorithms the mean delay slowly decreased as a function of the load ρ_d in the high load area (beyond $\rho_d > 0.8$, resp. 0.9) . This is caused by the fact that higher loads imply that more ONUs are actively competing for bandwidth within a cycle. Thus, the difference between B_{max} and the cycle length C under high load conditions grew as a function of ρ_d, creating shorter cycles and therefore a small reduction in the mean delay of the priority 0 traffic. With the SRTLr schemes the number of active ONUs in a cycle has no real impact on the difference between B_{max} and C (under high load conditions). Therefore, the minor drop in the average delay is not present.

Before proceeding with the priority 1 and 2 results, the impact of the threshold reporting mechanism on the efficiency results in Figure 2.12 is discussed. The figure shows the efficiency of the SRTL algorithms when the threshold reporting is considered

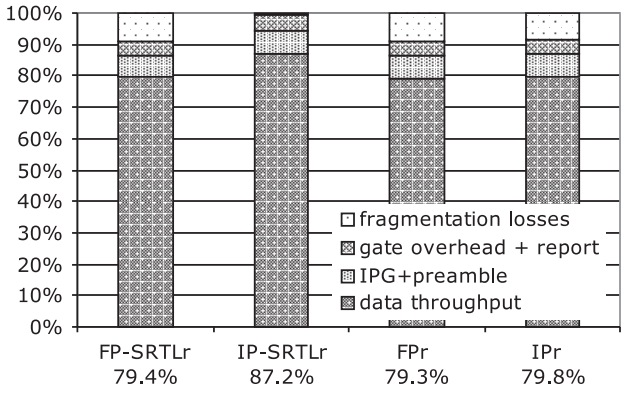
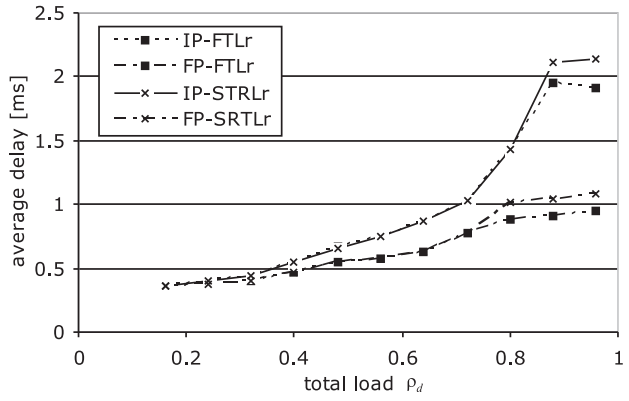


Figure 2.12: The bandwidth utilization at high loads ρ_d for FP and IP intra-ONU scheduling with and without threshold reporting.

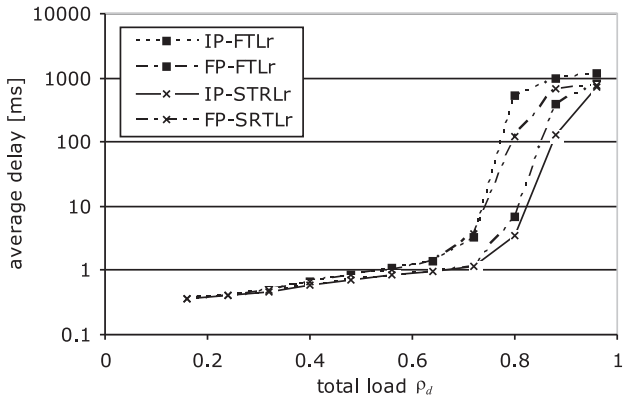
and when the thresholds are not accounted for (FPr and IPr). If the threshold reporting mechanism is turned off, nearly the same throughput results are obtained for FP-SRTLr (in overload situations: 79.4% vs 79.3%). This confirms the idea that threshold reporting is useless when the ONUs use FP, because as soon as some new priority p traffic arrives at the ONU between two TWs, all priority $q > p$ packet boundaries that were reported with the threshold mechanism in the first TW become worthless. For IP-SRTL(r) however, implementing a threshold mechanism can be worthwhile, especially in overload situations. For instance, if the threshold mechanism is turned off (and thus only the complete queue lengths are reported) the data throughput decreases from 87.2% to 79.8% with $\rho_d = 0.96$. This can be seen from Equations (2.4) and (2.10) as the threshold reporting causes the remainder $E(R)$ and correspondingly the fragmentation losses F to become ≈ 0. Comparing these results with the ones for FTL from Figure 2.8(b), it is obvious that the increased cycle length results in a better efficiency. For FP it rises from 76.7% to 79.4% and for IP from 86.3% to 87.2%.

In Figure 2.13 the average delays for priority 1 (a) and priority 2 (b) traffic are shown. The longer average cycle length of the SRTLr algorithms in the region $\rho_d > 0.7$ leads to higher average packet delay for priority 1 traffic. The saturation point for priority 2 traffic is still reached at the same load. However because more traffic from priority 2 can be transmitted in a cycle the average delay at high loads is lower for the SRTLr algorithms.

Thus algorithms with SRTL scheduling at the OLT achieve higher throughput than with FTL regardless of the scheduling at the ONUs. Even though the algorithms with



(a)



(b)

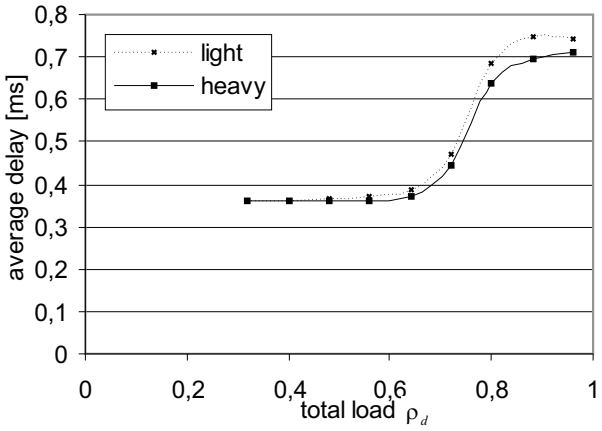
Figure 2.13: the average queuing delay of the (a) priority 1 and (b) priority 2 traffic as a function of ρ_d.

SRTL scheduling show a slightly higher average delay at high loads, based on the simulation results presented in this section they would be the better choice to be implemented in a real system. Furthermore the rate-based scheduling achieves far superior results regarding the delay of priority 0 traffic. In the following section simulation results are presented in order to gain better understanding of the performance of upstream bandwidth allocation with SRTLr scheduling at the OLT.

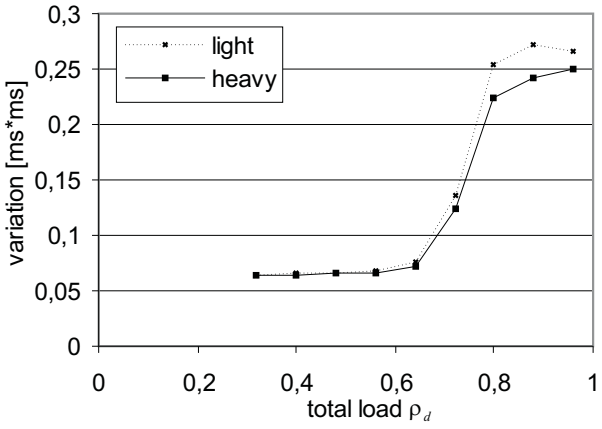
2.10 Performance Evaluation of SRTLr Scheduling Algorithms in Asymmetric Traffic Conditions

The aim of this section is to compare the performance of SRTLr scheduling algorithms in the presence of two types of ONUs on the network: ONUs with a lot of best effort traffic, called *heavily loaded* ONUs, and ONUs with considerably less best effort traffic, called *lightly loaded* ONUs. The total load ρ_d is varied between 0.32 and 0.96, but as opposed to the previous section only the amount of priority 2 traffic is changed when doing so. For priority 0 exactly the same traffic source as in Section 2.7.2 is used, which emulates a T1 connection with a UDP/IP/Ethernet protocol stack. For priority 1 and 2 a 2-state Conditioned Markov-Modulated Bernoulli Process (C-MMBP) is used with an arrival rate in state 1 five times as high as in state 2, but with different sojourn times. For priority 1 the sojourn time in state 1 is 21.7 ms and in state 2 it is 434 ms for both the heavy- and lightly loaded ONUs and this for all loads considered. For priority 2 the mean sojourn time in state 1 and 2 depends on the load ρ_d and lies within $[7.9, 313]$ msec and $[158.6, 6259.6]$ msec for the lightly loaded ONUs and 3 times less for the heavily loaded ONUs, respectively. As such the priority 2 traffic, i.e., best effort traffic, for the heavily loaded ONUs is 3 times as high as for the lightly loaded ones. The packet size distribution is the same as in the previous section.

Figure 2.14(a) presents the delay of priority 0 traffic for the FP-SRTLr algorithm. At low loads up to $\rho_d = 0.4$ the two types of ONUs experience the same delay. This is the region where $R_{tot} \ll B_{min}$ and the allocated bandwidth, i.e., TW, for the ONUs is always larger than the requested bandwidth. As the total load gradually increases the asymmetric nature of the priority 2 traffic becomes visible. In the region $0.4 < \rho_d \leq 0.9$ the lightly loaded ONUs suffer a slightly higher (priority 0) delay in comparison with the heavily loaded ones. This stems from the fact that while all TWs are more or less of the same length when $\rho_d < 0.4$, this is no longer the case in the $[0.4, 0.9]$ area, meaning that heavily loaded ONUs tend to get larger TWs. Priority 0 traffic that arrives while the TW is ongoing will interject the transmission of lower priority traffic (as soon as the transmission in progress ends). Therefore, the priority 0 traffic of heavily loaded ONUs is favored in comparison with lightly loaded ONUs. Also, the mean distance between two consecutive TWs (measured from the end of the first until the start of the second) is less for heavily loaded ONUs, which implies that the mean time until the next TW is less for a packet that arrives at the ONU outside a TW. As the load increases in the $[0.4, 0.9]$ area, so does the difference between the average length of a TW assigned to a heavily loaded ONU and a lightly loaded one. Thus, the difference in delay between



(a)

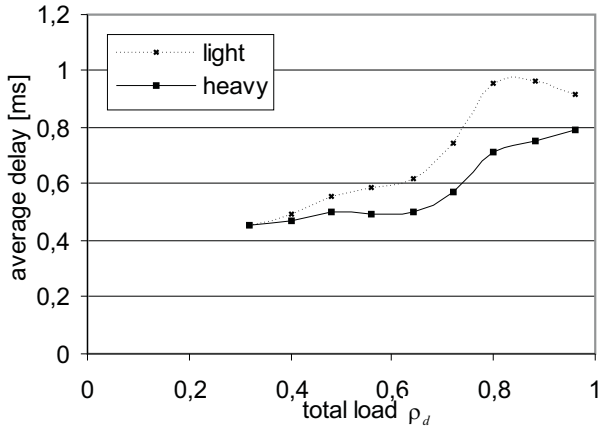


(b)

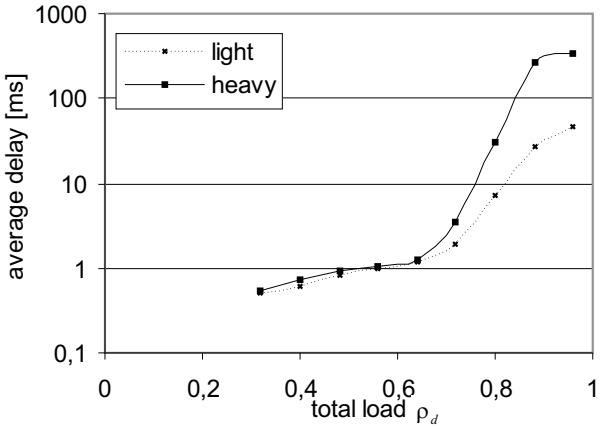
Figure 2.14: Asymmetric Traffic: FP-SRTLr (a) the average queueing delay and (b) the delay variation of the priority 0 traffic as a function of ρ_d.

both types of ONUs grows.

In the $\rho_d > 0.9$ region the system becomes severely saturated. Meaning that the heavily loaded ONUs will drop large amounts of best effort traffic. Thus, the throughput ratio for priority 2 traffic between heavy- and lightly loaded ONUs, which equaled 3 at low loads, begins to decrease. For instance, at $\rho_d = 0.96$ the ratio equals 1.7. Hence, the ratio between the length of the TWs allocated to heavy- and lightly loaded ONUs decreases causing the delays for priority 0 to converge to the same value. As is to be expected, the packet delay variation for priority 0 (see Figure 2.14(b)) behaves similarly to the average delay.



(a)

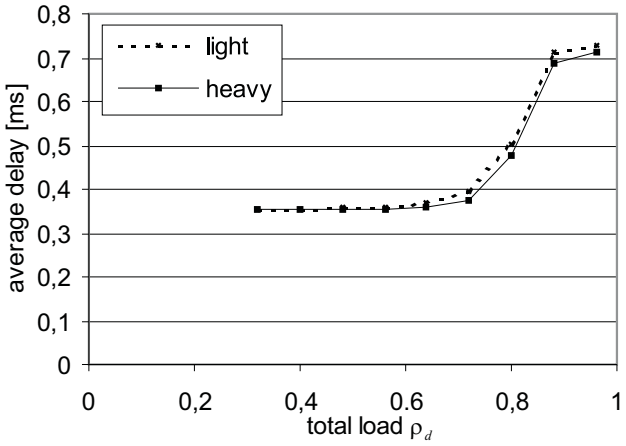


(b)

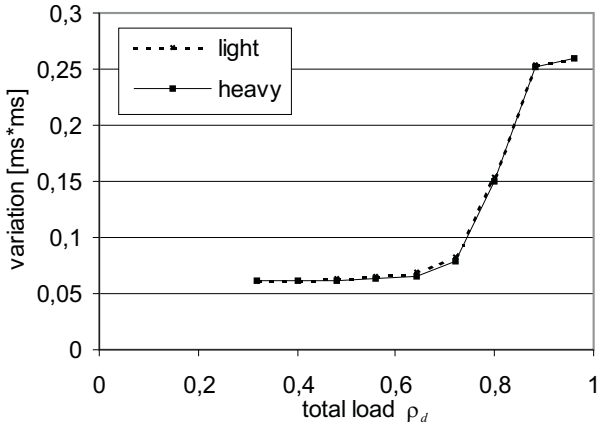
Figure 2.15: Asymmetric Traffic: FP-SRTLr (a) the average queueing delay for priority 1 (b) the average queueing delay for priority 2 as a function of ρ_d.

The delay for priority 1 traffic, when using FP-SRTLr, behaves similarly to the priority 0 traffic, but here the difference between the delays experienced by both types of ONUs is substantially larger. This difference is again caused by the asymmetry in the TW sizes as explained for the priority 0 traffic (because, priority 1 traffic is allowed to interject priority 2 traffic when using FP-SRTL(r)). This is a serious drawback of any FP scheduling scheme. The amount of best effort traffic should not affect the performance of higher priority traffic, therefore, an FP scheduling scheme is not advised. As with the priority 0 traffic, the delay of both types of ONUs at severe overload conditions converges to the same value. As can be seen from Figure 2.15, the delay of priority 2 traffic for

the lightly loaded ONUs is obviously less than the one for the heavily loaded ONUs for all loads. The difference between both types of ONUs starts to increase significantly as soon as the system reaches its saturation point.



(a)

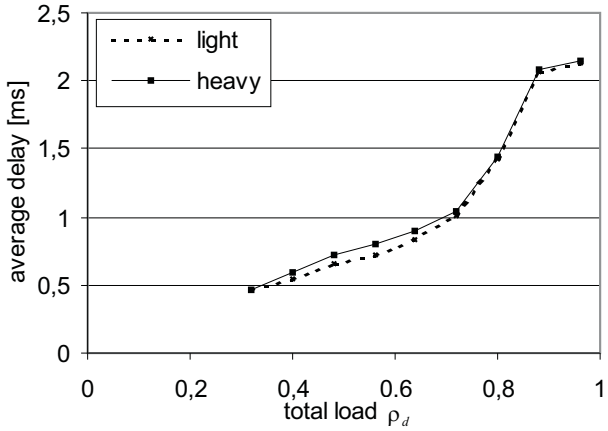


(b)

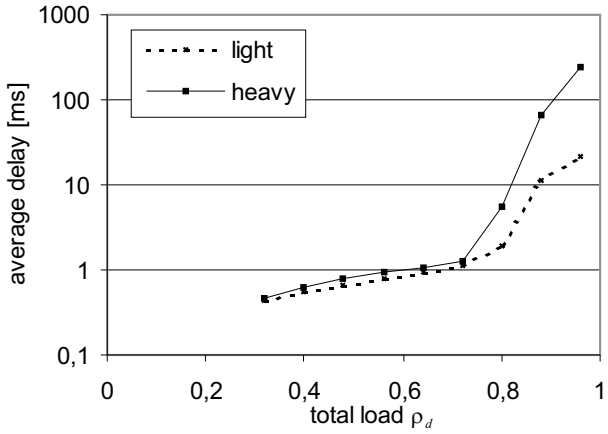
Figure 2.16: IP-SRTLr (a) the average queueing delay and (b) the delay variation of the priority 0 traffic as a function of ρ_d.

The average packet delay for priority 0 when using IP-SRTLr is presented in Figure 2.16(a). The delay for the two types of ONUs are close to each other. Recall that with IP-SRTLr the packets from priority 0 are still transmitted before the reported lower priority traffic. Thus, as far as the policy to transmit priority 0 traffic is concerned, there is no difference between FP-SRTLr and IP-SRTLr. The reason for the higher delays for lightly loaded ONUs is thus the same as with FP-SRTLr. The packet delay variation for

priority 0 of IP-SRTLr is presented in Figure 2.16(b). It is fair to state, based on these figures, that the asymmetry does not affect priority 0 traffic in a significant way.



(a)



(b)

Figure 2.17: IP-SRTLr (a) the average queueing delay for priority 1 (b) the average queueing delay for priority 2 as a function of ρ_d.

The average packet delay for priority 1 is presented in Figure 2.17(a). Due to the IP scheduling scheme, priority 1 traffic is no longer able to interject reported priority 2 data. Thus, one would expect the same results for heavy- and lightly loaded ONUs for all data loads ρ_d. Somewhat surprisingly, IP-SRTLr slightly favors lightly loaded ONUs in the $[0.3, 0.7]$ region. In this region the TW of all ONUs is composed of two parts: a first that corresponds to the reported (mostly priority 2) data and a second being some fair share which is identical for all ONUs (in order to reach a cycle of length B_{min}). With

IP-SRTL(r), newly arrived priority 1 traffic can only make use of the fair share of the TW. Now, any priority 1 traffic that arrives during the TW is transmitted immediately, provided that no other priority 0 or 1 traffic is pending and unless a transmission of a (priority 2) packet is ongoing (in which case it has to wait until this transmission is completed). Such ongoing transmissions occur more frequently for ONUs with more priority 2 traffic.

Thus, lightly loaded priority 1 traffic has a slight advantage on the heavily loaded ONUs. Notice, when an ongoing transmission of a (priority 2) packet ends there might not be enough bandwidth left in the TW to transmit the priority 1 packet, causing an additional delay of one cycle. At a load $\rho_d = 0.32$ both curves coincide as there is hardly any priority 2 traffic generated in either type of ONU. As the load increases the priority 1 packets of lightly loaded ONUs are more likely to make use of the fair share. The mean size of the fair share however decreases as a function of the load (at $\rho_d = 0.7$ its size is zero). Thus, the load at which the advantage of the lightly loaded ONUs is maximal is about halfway between 0.32 and 0.7. Finally, Figure 2.17(b) represents the priority 2 delays, these are obviously higher for heavily loaded ONUs.

2.11 Conclusions

In this chapter a variety of upstream bandwidth allocation algorithms for EPON using threshold reporting were studied. For the purpose first methods are defined to generate reports with thresholds and to process them at the OLT in a way suitable to base the scheduling decision on these reports. An upstream scheduling algorithm for EPON is defined by the scheduling mechanisms in both the ONU and in the OLT. Two types of scheduling at the ONU were considered, namely full priority (FP) scheduling, which is the common strict priority scheduling and interval priority (IP) scheduling, where higher priority traffic is transmitted before lower one only if it was already reported to the OLT. The types of algorithms proposed for the OLT differ in the threshold level they base their scheduling algorithm - full and single report threshold level (FTL and SRTL) - and in their scheduling policy for constant bit-rate traffic . Namely, the upstream bandwidth allocation algorithms designed and compared in this chapter are:

- Full priority - full threshold level scheduling (FP-FTL):
  The FP-FTL scheduling algorithm combines full priority scheduling at the ONU with full threshold level scheduling at the OLT. The biggest advantage of this discipline is that no intelligence at the ONU is required. This is a typical requirement

from equipment manufacturers which try to lower the cost of the equipment at the customer premises.

- Interval priority - full threshold level scheduling (IP-FTL):
  This algorithm improves on FP-FTL by using Interval Priority (IP) scheduling at the ONU. This increases the complexity of the algorithm at the ONU site but lower priorities can not starve due to discrepancies between the requested and the transmitted traffic by the ONU. The disadvantage is that lower priority traffic can be transmitted before high priority traffic, provided that the later was not reported. This results in slightly higher delay for the high priority traffic which for some applications, like voice, that are usually mapped to this priority, is not desirable.

- Full priority - full threshold level scheduling with rate estimation (FP-FTLr):
  This algorithm improves on FP-FTL by using scheduling with rate estimation at the OLT. Estimating the rate of the highest priority traffic avoids the starvation of traffic with lower priority. However, with the exception of the traffic with priority on the highest active level closest to the highest priority, all the other traffic will be prone to starvation.

- Interval priority - full threshold level scheduling with rate estimation (IP-FTLr):
  This algorithm adds rate estimation scheduling at the OLT to the IP-FTL algorithm. This improves the delay of the highest priority traffic and thus actually resolves both big lacks of the above described algorithms, namely the need to request bandwidth for the highest priority traffic and all consequences from this and the starvation of the lower priority traffic.

- Full priority - single report threshold level scheduling with rate estimation (FP-SRTLr):
  This algorithm combines full priority scheduling at the ONU with SRTL scheduling with rate estimation at the OLT. Due to the longer average cycle length the achievable throughput is increased. However, the algorithm significantly favors ONUs with substantial amounts of best effort traffic, causing unfairness between the end users.

- Interval priority - single report threshold level scheduling with rate estimation (IP-SRTLr):
  This algorithm combines interval priority scheduling at the ONU with SRTL

scheduling with rate estimation at the OLT. This algorithm achieves the highest data throughput and due to the rate estimation, low delays for the high priority traffic.

In conclusion the results on the designed algorithms presented in this chapter demonstrate that the most simple and straightforward algorithm, FP-FTL has acceptable delay characteristics for CBR traffic but achieves the lowest utilization. This drawback is resolved by IP intra-ONU scheduling, which leads to the IP-FTL variety. It was shown that with IP-FTL it is possible to reduce the bandwidth losses, caused by not fragmenting Ethernet frames, to almost zero. The drawback of IP-FTL is a strong increase of the average queueing delay and the delay variation for the highest priority traffic (CBR traffic) at high data loads. By combining IP-FTL with rate based scheduling for CBR traffic (IP-FTLr), one can significantly reduce this high load delay increment. With respect to the bandwidth efficiency and the delay of the lower priority traffic, IP-FTL was shown to perform slightly better than IP-FTLr. Combining FP-FTL with the rate based scheduling for CBR traffic (FP-FTLr) results in a priority 0 performance similar to IP-FTLr, but improves the average queueing delay characteristics of the priority 1 traffic. The drawbacks of FP-FTLr compared to IP-FTLr are a significant reduction in the bandwidth efficiency and that it significantly favors ONUs with substantial amounts of best effort traffic, causing unfairness between the end users

The SRTL scheduling at the OLT can further improve the efficiency if combined with IP scheduling at the ONU (IP-SRTLr). Combined with FP scheduling (FP-SRTLr) no efficiency gains were observed. It was demonstrated that the QoS support of high priority traffic is not influenced by the presence of best effort traffic when using IP-SRTLr, as opposed to FP-SRTLr that favors ONUs with lots of best effort traffic. While slightly higher delays for some priorities at high loads are observed with IP-SRTLr, it realizes the best fairness and efficiency when compared to the other algorithms.

Chapter 3

Simulation Framework for DOCSIS 2.0 based HFC networks and Performance Evaluation of Upstream Scheduling Services

In this chapter the implemented simulator for the DOCSIS 2.0 based HFC networks is described. It is implemented in C++ using the open source OMNET++ events-based simulation package. Unlike other open source DOCSIS simulators it models in detail the physical medium dependent layer and many features of the MAC layer. It is designed to be used to evaluate the performance of DOCSIS networks in different scenarios and to evaluate different scheduling algorithms. This is very useful for cable operators in order to fine tune parameters on already deployed networks, to compare different manufacturers equipment and to design improvements.

The implemented upstream scheduling service types are: the Unsolicited Grant Service (UGS), the real time Polling Service (rtPS) and the Best Effort (BE) service. The simulator has been extensively validated and some very simple validation scenarios, which constitute good examples of the operation of the DOCSIS protocol and of the implemented model, are presented in this chapter. The performance of the rtPS and BE scheduling service is evaluated via simulations and improvements to the scheduling and optimal values for some contention channel and MAC protocol parameters are discussed in the chapter.

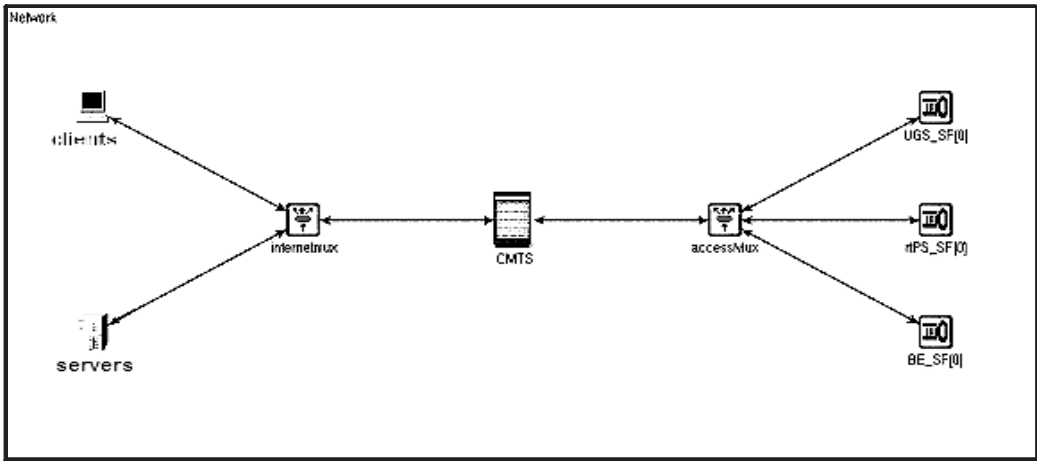


Figure 3.1: DOCSIS OMNET++ Model

3.1 The DOCSIS 2.0 Model

The model implemented uses the OMNET++ network simulation package and is developed in C++. It is based on the DOCSIS 2.0 specifications [3]. It models in detail the physical, the MAC and the TCP network layers.

In Figure 3.1 a top view from the graphical interface of OMNET++ of the implemented model is presented. It consists of a CMTS node connected via a shared medium emulator to a number of cable modem nodes in the access part and to a number of clients and servers on the Internet side.

The server nodes in the network model represent nodes on the Internet side of the HFC network, which can be content servers in the head-end itself or servers/clients situated in a distant location over the Internet. A server consists of a packet source and a transport layer model. The clients, on the other hand, have the role of the corresponding packet sink used to collect statistics. The server can generate packets with constant and variable bit rate. The variable bit rate source is modelled as an ON-OFF source, which can have different distributions for the ON and OFF times and for the packet inter-arrival times. The constant bit rate source can be used to model voice traffic, while the ON-OFF source is meant for use as video and data traffic generator. Each client corresponds to a server on the other side of the network (”Access” or ”Internet”) and gathers statistics for the packets generated at this server. It also generates acknowledgments in case TCP is used as the transport protocol.

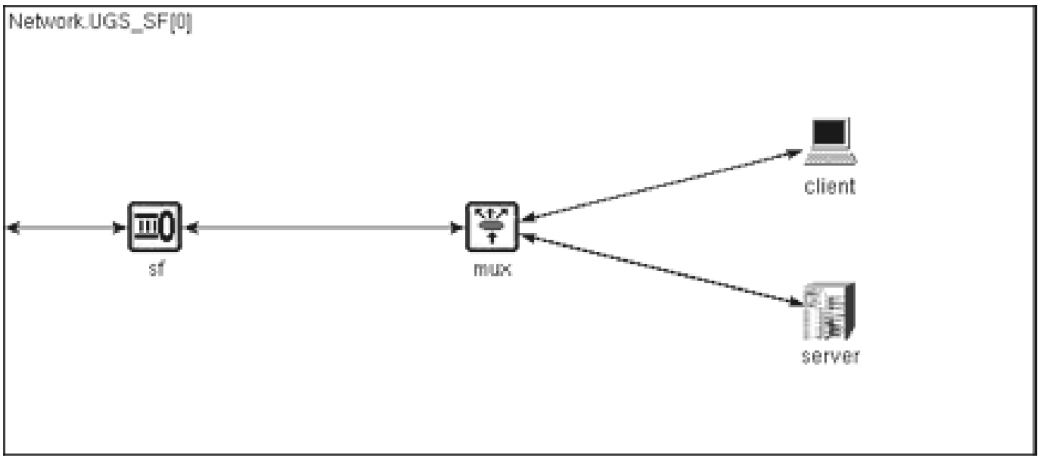


Figure 3.2: Compound Service Flow Model

The CMs are grouped in three groups depending on their upstream (US) scheduling service type. A schematic view of the CM module is given in Figure 3.2. It consists of a Service Flow (SF) module, which is connected to 1 client and 1 server via a local "mux". The "mux", as the "internetmux" and the "accessmux" in Figure 3.1, sends the traffic coming from the multiple inputs directly to the node connected to the single output without queuing. In the other direction packets coming from the single input are send to the output corresponding to the address in the packet. Thus these elements do not have complicated logic and in one direction serve as simple repeaters. The server and client nodes in the CM model are the same as in the network model from Figure 3.1. It is the SF node which is responsible for modelling the DOCSIS protocol on the user (CM) side. Once a SF is set at the start of the simulation it remains active throughout it, i.e., the model does not support dynamic service flow creation or destruction.

The implemented protocol stack between the servers/clients and the CMTS or the SF is TCP/IP/Ethernet or UDP/IP/Ethernet. The TCP model implements the basic behavior of the protocol according to RFC 793 [104] and RFC 1122 [105]. The IP layer is accounted for by adding extra 20 bytes to the packets corresponding to the IP header size. On the link layer the packets are formatted as Ethernet frames, hence, incurring 18 bytes overhead.

Further on, the physical medium dependent layer model and the MAC layer features and operations will be explained in detail.

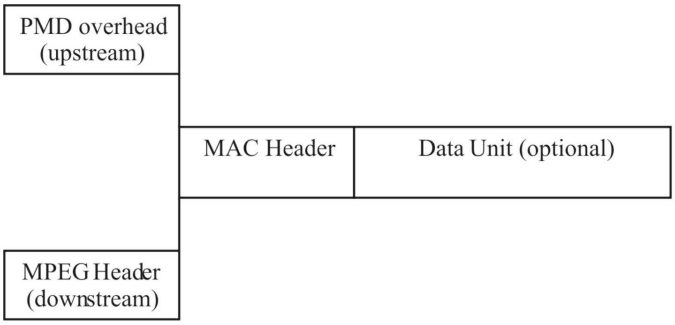


Figure 3.3: Generic MAC frame format

3.1.1 Physical Layer Model

The upstream channel between the CMs and the CMTS is modelled as a stream of mini-slots. A mini-slot represents the time needed for the transmission of a fixed number of symbols and can represent a minimum of 16 and a maximum of 48 bytes, depending on the modulation used.

The basic transfer unit between MAC sub-layers at the CMTS and the cable modem is the MAC frame. The same basic structure is used for both directions. There are three distinct regions to consider, as shown on Figure 3.3. Preceding the MAC frame is either the physical medium dependent (PMD) sub-layer overhead (upstream) or MPEG transmission convergence header (downstream). The first part of the MAC frame is the MAC header. It identifies the MAC frame. Following is the optional Data unit. The format of the data unit and whether it is present or not is described in the header. The actual size of the MAC headers is given in Table 3.1.

The transport of MAC frames by the PMD sub-layer for upstream channels is shown on Figure 3.4. Besides the PMD overhead, additional overhead comes from the forward error correction (FEC) per MAC frame containing a data unit and from the truncation to an integer number of mini-slots indicated on the figure as the PMD overhead at the

Table 3.1: MAC headers size

MAC headers	bytes
Packet MAC header	6
Extended MAC header	5

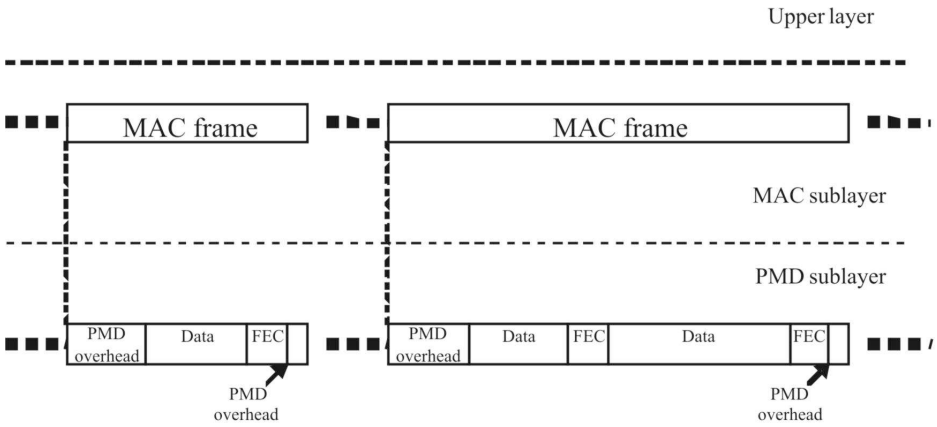


Figure 3.4: Upstream MAC/PMD Convergence

end of a MAC frame. On the figure "Data" indicates the MAC frame including the MAC header and the data unit from Figure 3.3. Hereafter, the exact calculation of the overhead, which is incorporated in the implemented DOCSIS model is described.

The parameters which contribute to the PMD overhead are given in Table 3.2. They depend on the modulation scheme used. Note that each separate transmission should start at the start of a mini-slot. This also contributes to the overhead. The 'Specific' parameters can be of three types depending on the type and length of the frame: "request", "short" data or "long" data. The 'Common' parameter is the same for all the types. It determines which data units are considered "long" and which "short". A "request" frame is a MAC frame without data unit and MAC header consisting of the regular packet MAC header and extended MAC header, with sizes as given in Table 3.1.

Using these parameters the total layer 1 (L1) or physical layer size of an 'Ethernet frame' packet of size X bytes and header of size H bytes is calculated in minislots as follows.

$$L1(X+H)[minislots] = \frac{L1(X+H)[bytes]}{bm}, \tag{3.1}$$

where

$$L1(X+H)[bytes] = bm * \lceil \frac{(Preamble + Guard * bs)/8 + (Frame + FEC)}{bm} \rceil,$$

Table 3.2: Modulation Parameters

Common
Max Short Payload [bytes]
Specific
FECerr [bytes]
FEClength[bytes]
Guard[symbols]
Preamble[bits]
LastCW[fixed/short]
Bits/symbol(bs)
Bytes/minislot(bm)

where $\lceil x \rceil$ is the minimum integer $\geq$x and

$$(Frame + FEC) = NfullCW * (FEClength + 2 * FECerr) + Y.$$

The number of full code words $NfullCW$ is obtained from

$$NfullCW = \lfloor \frac{X + H}{FEClength} \rfloor,$$

where $\lfloor x \rfloor$ is the biggest integer $\leq x$ and Y is obtained by the pseudo-code

$if(Frame\_last\_CW == 0)$
  $Y = 0;$
else
  $Y = MAX(CW\_FEC, 16) + 2 * FECerr,$

where

$$Frame\_last\_CW = X + H - NfullCW * FEClength.$$

And CW_FEC depends on the LastCW parameter and is given by

$If(LastCW == fixed)$
  $CW\_FEC = FEC\_length;$
else
  $CW\_FEC = Frame\_last\_CW.$

The downstream is defined as a continuous stream of 188-byte MPEG [106] packets. These packets consist of a 4-byte header followed by 184 bytes of payload. The physical

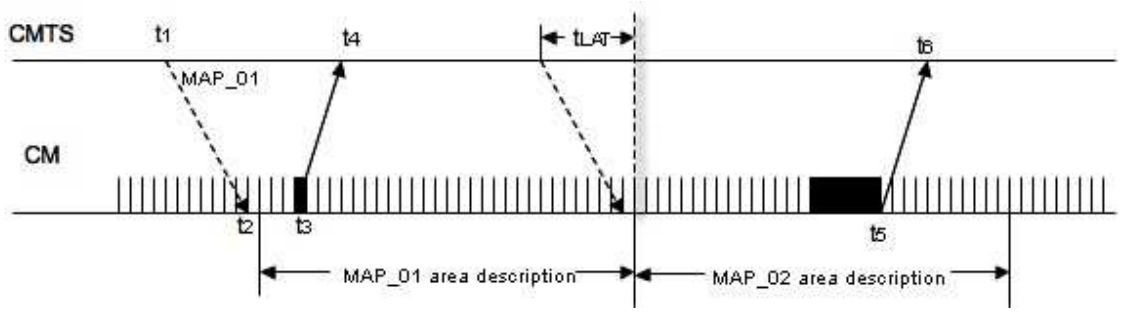


Figure 3.5: US MAP allocation based on bandwidth requests

overhead in the DS direction is constant, equals $\frac{4}{188} * 100\%$ and is accounted for in the model as a constant.

3.1.2 Medium Access Control layer model : features and operations

Recall from Section 1.1.1 that the scheduler at the CMTS arbitrates the transmissions from the SFs by periodically sending a Bandwidth Allocation Map (MAP) message over the downstream channel to indicate to the SFs the specific mini-slots allocated to them. Note that because in the model each CM has only one SF in the upstream, the terms CM and SF are used interchangeably. A MAP consists of 'Information elements' each of which describes a transmission opportunity including the type - contention or unicast, the SF it is addressed to, if unicast, the start time, the length, the modulation and other information. In the contention transmit opportunities many CMs are allowed to transmit, while in unicast opportunities only one can transmit.

A MAP describes the bandwidth allocations for a period, which can have variable length and minimum and maximum values for the MAP length can be set as parameters in the simulation model. In Figure 3.5 the concept of the allocation MAP is visualized. The CMTS ensures that by the time the MAP description area starts all SFs would have received it and processed the information elements in it. To account for the transmission and processing delays at both sides the MAP is generated and sent t_{LAT} seconds before the actual start time of the MAP description area. The t_{LAT} (called 'Look Ahead Time') is a parameter of the simulator and can be related to the maximum distance of the currently active SFs. For example, the time instant t_1, when the MAP is transmitted, is such that by the time the MAP_01 description area starts, all CMs have received and processed it. If t_2 is the time when a CM receives the MAP then $t_2 - t_1 \leq t_{LAT}$. When determining t_{LAT} it should be taken into account that the CMs can be at distances of

up to 160 km. A SF from the Best Effort upstream scheduling service type may use the contention mini-slots for transmitting its requests at t_3. The contention opportunities are at the beginning of the MAP such that if t_{LAT} is less than one MAP length, a contention request can reach the CMTS at time t_4 before the scheduling for the next MAP starts. If capacity is available the CMTS will allocate transmission opportunity for the SFs in the next MAP. At the time instant t_5 the data is transmitted from the SF and at time t_6 it will reach the CMTS.

In each MAP there is a minimum number of contention transmit opportunities. When there are not enough bandwidth requests to utilize the full minimum MAP length, the CMTS schedules the unused bandwidth for contention request opportunities.

There are a number of ways by which a SF can explicitly request bandwidth from the CMTS for data transmission. Namely, it can do so through contention, piggybacking or unicast opportunities.

- Contention Requests
  Portions of the upstream bandwidth are open for requesting upstream bandwidth. The requests transmitted through contention are subject to collision, and these collisions are resolved by a Contention Resolution Algorithm.

- Piggybacking
  A request for additional bandwidth can be sent together with data transmission. Piggybacking obviates contention, since the requests are transmitted with the data packets.

- Unicast Request Polls
  Periodic unicast request opportunities are sent as a means of real-time polls regardless of network congestion. These opportunities are used by the rtPS SFs to transmit their request packets, avoiding contention.

As a result of bandwidth reservation, the SFs are guaranteed collision-free data transmission. But collisions may occur during the contention (request) period, and this is resolved using a Contention Resolution Algorithm (CRA). The mandatory method of contention resolution that must be supported is based on a Truncated Binary Exponential Back-off, with the initial back-off window (Wi) and the maximum back-off window (Wmax) controlled by the CMTS. These values are simulation parameters and represent a power-of-two value. Every time a CM wants to transmit in a contention region, it enters the contention resolution process by setting its internal back-off window equal to the Data Backoff Start. The CM selects randomly a number within its back-off window.

This random value indicates the number of contention transmit opportunities which the CM must defer before transmitting. If after a contention transmission the CM does not find a MAP indicating that the transmission is received or that a grant has been scheduled, it increases its back-off window by a factor of two, as long as it is less than the maximum back-off window. The CM selects randomly a number within its new back-off window and repeats the deferring process described above. The retry process continues until the maximum number of retries (16) has been reached.

In the developed simulator the cable modem has only one service flow (SF) for each direction. It can be one of the following service flow types: UGS, rtPS or BE.

- Unsolicited Grant Service (UGS) SF
  The UGS SFs do not use any polling opportunities and the CMTS does not provide such. When a UGS SF receives a MAP which has an element address to it and it has data (Ethernet frame(s))in its queue it reads the grant start time indicated in the MAP element and sets a timer for it. When the timer expires it transmits frames until the indicated in the MAP 'Information Element' grant length. The key service parameters are : 'Unsolicited Grant Size', 'Nominal Grant Interval', 'Grants per Interval' and 'Tolerated Grant Jitter'.

  - 'Unsolicited Grant Size' parameter indicates the size of the grant the CMTS has to schedule
  - 'Nominal Grant Interval' parameter indicates the nominal interval between which a UGS SF has to receive a grant
  - 'Grants per Interval' parameter indicates the number of grants in one grant interval that the service flow must be granted
  - 'Tolerated Grant Jitter' parameter indicates the maximum interval that a grant can be late or early.

- Real-time Polling Service (rtPS) SF
  The CMTS provides periodic unicast request opportunities. The rtPS SF does not use any other request opportunities than the unicast ones. It sends request containing the size of the frames in its buffer. When a grant for data arrives it transmits the frame itself. The key service parameters are 'Nominal Polling Interval' and 'Tolerated Poll Jitter'.

  - 'Nominal Polling Interval' parameter indicates the nominal interval between which a rtPs SF has to receive a unicast polling grant

  - 'Tolerated Poll Jitter' parameter indicates the maximum interval that a polling grant can be late or early.

- Best Effort (BE) SF
  The intent of the BE service is to provide efficient service to the best effort traffic. BE SF uses contention and piggyback request opportunities. When a MAP element, indicating contention opportunities, is received by a BE SF, it stores this element. When an Ethernet frame from its server arrives at the SF while its queue was empty it enters the contention process. A number is generated between 0 and its data start back-off value. The SF then waits the same number of contention request opportunities before transmitting its request. If in the next MAP it doesn't receive indication that its request has been received by the CMTS it follows further the contention resolution algorithm. The key service parameters are 'Minimum Reserved Traffic Rate' and 'Maximum Sustained Traffic Rate'.

  - 'Minimum Reserved Traffic Rate' parameter indicates the reserved rate for the flow accounting for all overheads. However as there is typically no admission control in subscriber access networks this parameter is used to determine weight w_i for the flow.
  - 'Maximum Sustained Traffic Rate' parameter indicates the maximum sustained rate for the flow accounting for all overheads. In the model a leaky bucket shaper is used to enforce this parameter.

Within one request a SF sends the sum of the sizes of the frames in its buffer, which are Ethernet frames. When the CMTS receives a bandwidth request from a SF it has to recalculate it so that it accounts for all upstream overheads as calculated in section 3.1.1. A SF can have only one outstanding request.

The DOCSIS model supports fragmentation and concatenation of data packets in the upstream. Concatenation allows the SFs to send multiple MAC frames in the same transmission. Fragmentation happens when the CMTS provides a data grant smaller than the amount the SF requested. In such a case, the SF fills the partial grant it receives with the maximum amount of data possible, and sends the rest of its data payload in the subsequent grants.

In the downstream (DS) direction the data packets from the Internet side servers and clients are multiplexed in one stream, which is terminated at the CMTS. They are buffered in per destination service flow queues and the downstream scheduler selects the next queue from which a packet will be transmitted on the access network. Note

that in the DS direction the CMTS is just a router. Thus, any scheduling algorithm implemented for routers can be applied without changes in the DOCSIS DS scheduling. Vice versa, any algorithm designed for routers can be simulated in the proposed network model.

3.2 Operation of the Simulation Model

This section describes several simple scenarios which illustrate and clarify the operation of the DOCSIS protocol in the upstream and the working of the implemented simulator. It includes examples of the rtPS SF operation and the unicast request polls generation at the CMTS. It also includes examples of the BE SF operation and of the contention resolution algorithm.

3.2.1 rtPS SF Request - Grant Mechanism

As described in the previous section unicast request polls are used by rtPS SFs to request US bandwidth from the CMTS. The queuing delay is considered as the interval of time Δt between the packet arrival at the SF's queue and the time the packet leaves the queue. Further on, the elements of this queuing delay are discussed in detail.

When a packet arrives at a SF from rtPS type at time $t_{arriveCM}$ (see Figure 3.6) it is queued in the service queue. The SF waits for a unicast request poll to report this packet. *Time_tillRequest* designates the time the packet spends in the queue before the SF generates a request indicating the size of the packet in the queue expressed in minislots. This is the time between $t_{arriveCM}$ and the start of a unicast request for the SF, indicated on Figure 3.6 as the interval *Time_forTransmitRequest*. It can vary between 0 and Nominal_Poll_Interval.

The interval of time between a request being generated at a SF and the time this SF receives a data grant in response to this request is designated with *Time_forRequest*. On the figure this will be the time between the start of *Time_forTransmitRequest* and T_{k+1}. The time that the grant response is received is taken to be the start of a MAP. This *Time_forRequest* interval includes the time to transmit the request (*Time_forTransmitRequest*), which is determined from the actual size in minislots of the request divided by the upstream rate, the transmission delay (*TransmissionDelayCM-toCMTS*), the time this request spends in the CMTS before the start of the scheduling algorithm (*Time_tillScheduling*) and the look-ahead time of the scheduler t_{LAT}. It can

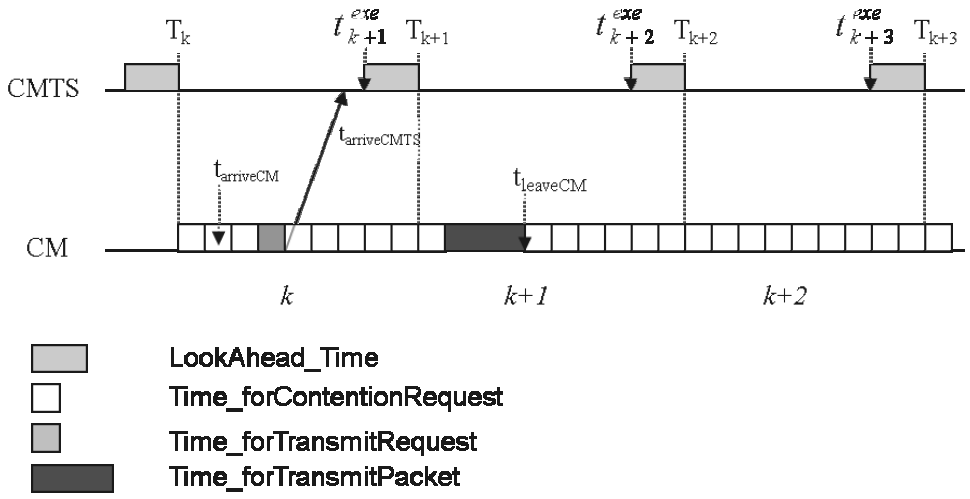


Figure 3.6: Queueing delay for a packet that arrived at an empty queue

be expressed as

$$\begin{aligned} Time\_forRequest = \quad & Time\_forTransmitRequest + \\ & TransmissionDelayCMtoCMTS + \\ & Time\_tillScheduling + t_{LAT}. \end{aligned} \tag{3.2}$$

The request arrives at time $t_{arriveCMTS}$ at the CMTS. It is queued there and is taken into account by the scheduler at the next execution of the scheduling algorithm. As explained in Section 3.1.2 the scheduling algorithm is executed some time t_{LAT} before the actual starting time of the MAP. It is called "Look Ahead Time" and accounts for the time it takes to perform the scheduling and for the time it takes the MAP to reach all CMs. So the CMTS executes the scheduling algorithm at time

$$t^{exe}_{k+1} = T_{k+1} - t_{LAT}, \tag{3.3}$$

where T_{k+1} is the instant the $k+1$-th MAP description area starts. The MAP length is expressed as $\Delta T = T_{k+i+1} - T_{k+i}$.

The minimum compulsory contention request opportunities per MAP are scheduled always at the beginning of the MAP. If after all grants and unicast polls are granted, there are still unassigned periods in a MAP, it is filled with contention opportunities. The starting time of a data grant has some offset from the starting time of the MAP (*DataGrantOffset*). The $DataGrantOffset$ in this implementation of the scheduling

algorithm is at minimum *Time_forContentionRequest*, which is the time for the minimum number of contention requests per MAP.

The packet for which bandwidth was requested in the unicast poll is considered to have left the queue when the last bit has left it, i.e., after time $Time\_forTransmitPacket$. The packet arrives at the CMTS after a certain transmission delay.

The queuing delay of the packet is exactly the time interval between $t_{arriveCM}$ and $t_{leaveCM}$. This time spent in the queue can be expressed by

$$\begin{aligned} \Delta t &= t_{leaveCM} - t_{arriveCM} \\ &= Time\_tillRequest + Time\_forRequest + \\ &\quad DataGrantOffset + Time\_forTransmitPacket. \end{aligned} \tag{3.4}$$

Consider the following concrete example. The upstream rate is 40000 minislots/s. The minimum MAP length is 2ms and $t_{LAT} = 0.2ms$. One contention request opportunity is of 4 minislots, which is the same as for unicast request opportunity, or $Time\_forContentionRequest = Time\_forTransmitRequest = 0.1ms$. There is only one active rtPS CM at a distance such that the delay between the CM and CMTS is $0.2ms$. The $Nominal\_Poll\_Interval$ is set to 10 ms, the size of each packet is 12 minislots or $Time\_forTransmitPacket = 0.3ms$. The packet inter-arrival time is big enough to ensure that there is never more than one packet in the queue.

We are now interested in the minimum and maximum possible queuing delays for a packet from the rtPS SF. The scenario when the minimum packet queuing delay is achieved is presented on Figure 3.7.

When an rtPS flow becomes active the first unicast request is scheduled in the beginning of a MAP and each next one, after the Nominal_Polling_Interval, which in this scenario is $10ms$. As there are no other active flows in the system and the load from the only active flow is less than the system's capacity the MAPs have always the same size and exactly an integer number of MAPs fits in this interval. Thus, the unicast polls for the rtPS SF are always scheduled in the beginning of each fifth MAP right after the contention request opportunity, $Time\_forContentionRequest$. This fact influences other components of the delay.

The minimum $Time\_forRequest$ is calculated from Equation (3.2) when the minimum values of all the components are added

$$\begin{aligned} Time\_forRequest(min) = \quad & Time\_forTransmitRequest(example = 0.1ms) + \\ & TransmissionDelayCMtoCMTS(example = 0.2ms) + \end{aligned}$$

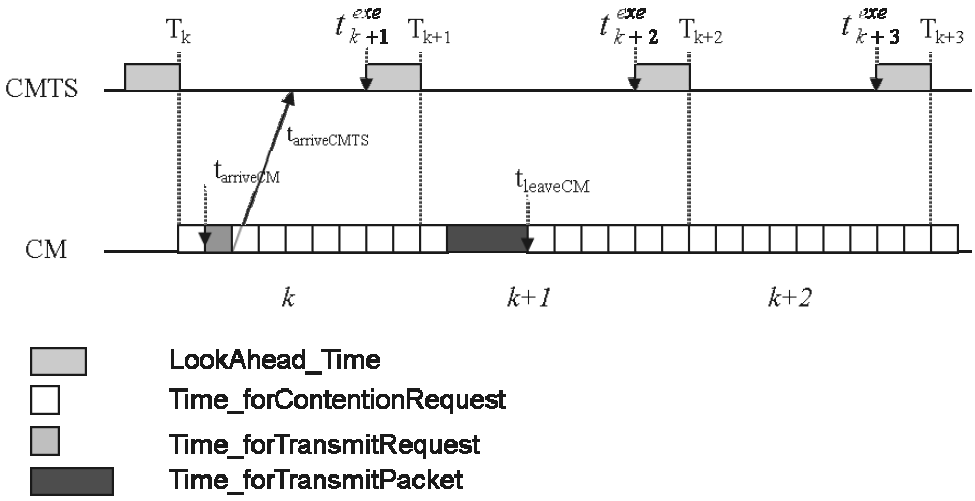


Figure 3.7: Minimum queuing delay for packet from rtPS SF

$$Time\_tillScheduling + t_{LAT}(example = 0.2ms). \quad (3.5)$$

The parameter $Time\_tillScheduling$ is the interval of time between the moment the request arrives at the CMTS ($t_{arriveCMTS}$) and the moment of the execution of the scheduling algorithm t^{exe}_{k+1} for the next MAP. This can be expressed as

$$\begin{aligned} Time\_tillScheduling &= t^{exe}_{k+1} - t_{arriveCMTS} \\ &= (T_{k+1} - t_{LAT}) - t_{arriveCMTS}. \end{aligned}$$

$t_{arriveCMTS}$ can be expressed relative to the k-th MAP starting time as

$$\begin{aligned} t_{arriveCMTS} = {} & T_k + \\ & Time\_forContentionRequest + \\ & Time\_forTransmitRequest + \\ & TransmissionDelayCMtoCMTS. \end{aligned}$$

If this expression is replaced in the one for $Time\_tillScheduling$ one obtains

$$\begin{aligned} Time\_tillScheduling = {} & (T_{k+1} - t_{LAT}) \\ & -(T_k + Time\_forContationRequest \end{aligned}$$

$$
\begin{aligned}
& + Time\_forTransmitRequest \\
& + TransmissionDelayCMtoCMTS) \\
= \; & \Delta T - Time\_forContationRequest \\
& - Time\_forTransmitRequest \\
& - TransmissionDelayCMtoCMTS \\
& - t_{LAT} \\
= \; & 2ms - 0.1ms - 0.1ms - 0.2ms - 0.2ms \\
= \; & 1.4ms.
\end{aligned}
$$

Finally the minimum value of the time for request is obtained by replacing the calculated minimum of *Time_tillScheduling* in Equation (3.5) and is

$$Time\_forRequest(min) = 1.9ms.$$

The minimum data grant offset is

$$DataGrantOffset(min) = Time\_forContationRequest = 0.1ms.$$

Recall also that the time for one request regardless if it is in a contention opportunity or unicast message is the same.

In this exemplary scenario the CM's server generates packets with a constant inter-arrival time of 10.001ms. Thus, the packet arrival in the SF queue ($t_{arriveCM}$) increments in steps of 1μs relative to the 2ms frame. Consequently, because the simulator schedules first the grants and then the packet arrival, the minimum $Time\_tillRequest$ is not 0 but

$$Time\_tillRequest(min) = \varepsilon = 1\mu s.$$

Thus the minimum queuing delay is obtained by replacing the minimum values of the terms in Equation (3.4) and is

$$
\begin{aligned}
\Delta t_{min} = \; & Time\_tillRequest(0.001ms) + Time\_forRequest(1.9ms) + \\
& DataGrantOffset(0.1ms) + Time\_forTransmitPacket(0.3ms) \\
= \; & 2.301ms.
\end{aligned}
$$

For the maximum queuing delay all the parameters stay the same except $Time\_tillRequest$,

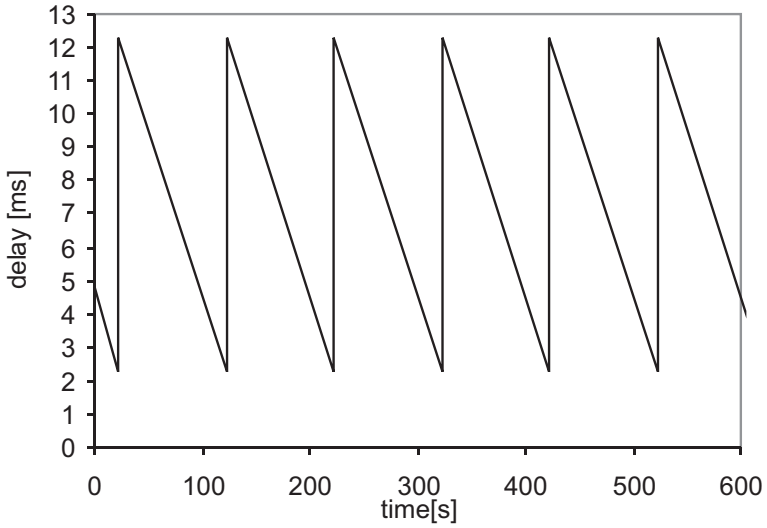


Figure 3.8: Queuing delay of the upstream packets reported via unicast request

which becomes the $Nominal\_Poll\_Interval$. Thus

$$\begin{aligned} \Delta t_{max} &= Time\_tillRequest(10ms) + Time\_forRequest(1.9ms) + \\ &\quad DataGrantOffset(0.1ms) + Time\_forTransmitPacket(0.3ms) \\ &= 12.3ms. \end{aligned}$$

On Figure 3.8 the results of the simulation of this scenario are given. On the X-axis is the progress in time in seconds and on the Y-axes is the queuing delay for each packet expressed in seconds. The packets experience different delays bounded by the minimum and maximum values. When a packet arrives at the queue just before the unicast requests scheduled for the flow, it will experience the minimum possible delay. The following packet will arrive just after the unicast request and will have to wait until the next unicast poll, hence experiencing the maximum possible delay. As can be seen from figure the analytically calculated values are met.

3.2.2 Best Effort Request-Grant Mechanism

In case the request is sent in a contention slot, like for the BE SF type, the queuing delay Δt is again given by Equation (3.4) but the different components have different maximum and minimum values. The interval of time between a request being generated at a SF and the time this SF receives data grant in response to this request is designated with

$Time\_forRequest$. For a successful request $Time\_forRequest$ has the same components as for unicast request explained in section 3.2.1, but some of them have different value intervals. $DataGrantOffset$ and $Time\_toTransmitPacket$ have the same meaning as in Section 3.2.1.

Consider the following concrete example. The upstream rate is 40000minislots/s. The minimum MAP length is 2ms and $t_{LAT} = 0.2ms$. One contention request opportunity is 4 minislots or $Time\_forTransmitRequest = Time\_forContentionRequest = 0.1ms$. The scheduling algorithm fills unclaimed bandwidth with contention opportunities. There is only one active SF and it is from the BE scheduling type, so at the moment of arrival of a packet in the empty queue the whole channel bandwidth is assigned for contention opportunities. The CM is at a distance such that the delay between the CM and CMTS is $0.2ms$. The size of the packets generated is 20 minislots or 0.5ms, which gives $Time\_forTransmitPacket = 0.5ms$. The server at the BE SF generates packets with a constant inter-arrival time of 10.001ms. Again as in Section 3.2.1 the packet arrival in the SF queue ($t_{arriveCM}$) increments in steps of 1μs relative to the 2ms frame.

Figure 3.9 shows an example where the minimum value is reached. When a packet arrives at the BE SF at time $t_{arriveCM}$ it is queued in the flow's queue. The CM enters the contention resolution process and according to the truncated binary exponential back-off algorithm chooses after how many contention slots it will send a request for this packet. The algorithm was described in Section 3.1.2. Because there is only one active BE SF then its contention requests will succeed on first attempt. The Data Back-off Start value of exponential binary back-off algorithm is set to 0 so the SF will send its request in the first contention opportunity and it will be successful. The successful request arrives at time $t_{arriveCMTS}$ at the CMTS, where it is taken into account by the scheduler at the next execution of the scheduling algorithm. As explained in Section 3.2.1 the execution of the scheduling algorithm can be expressed with respect to the next MAP starting time with Equation (3.3).

Again, as for the rtPS service flow, $Time\_tillRequest$ is the time until the SF generates a successful request indicating the size of the packet in the queue expressed in mini-slots. The minimum $Time\_tillRequest$ is again determined from the granularity of the arrival process or $Time\_tillRequest(min) = \varepsilon = 0.001ms$.

$Time\_forRequest$ can be calculated from Equation (3.2). The minimum value of the parameter $Time\_tillScheduling$ can be theoretically 0. However, one should also take into account that the request should have arrived some time before the execution of the scheduling algorithm at the CMTS. The granularity of these events is the size of a contention request because an integer number of requests fit in one MAP. Consequently

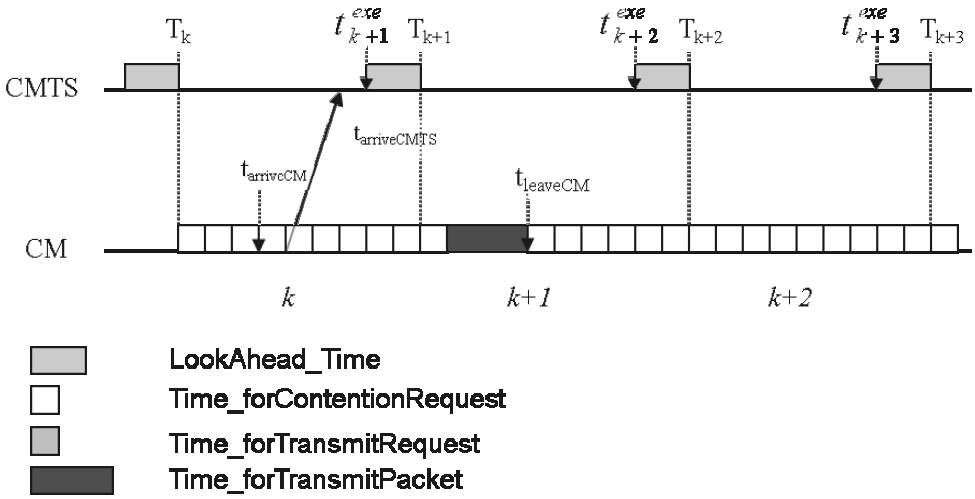


Figure 3.9: Minimum queuing delay for packet from BE SF.

for the minimum $Time\_forRequest$ is obtained

$$\begin{aligned} Time\_forRequest(min) &= Time\_forTransmitRequest(example = 0.1ms) + \\ &\quad TransmissionDelayCMtoCMTS(example = 0.2ms) + \\ &\quad Time\_tillScheduling(example = 0.1ms) + \\ &\quad t_{LAT}(example = 0.2ms) \\ &= 0.6ms. \end{aligned}$$

The minimum data grant offset is again determined from the number of contention transmission opportunities at the start of the MAP, which is one or $DataGrantOffset(min) = 0.1ms$. Thus, the minimum queuing delay for this scenario is

$$\begin{aligned} \Delta t_{min} &= Time\_tillRequest(0.001ms) + Time\_forRequest(0.6ms) + \\ &\quad DataGrantOffset(0.1ms) + Time\_forTransmitPacket(0.5ms) \\ &= 1.201ms. \end{aligned}$$

An example when the maximum queuing delay is reached is presented on Figure 3.10. If $t_{arriveCMTS}$ is more than t^{exe}_{k+1} then the request will be considered in the following MAP k+1 as depicted on Figure 3.10 and scheduled in MAP $k+2$. Thus the maximum queuing delay is reached when the request arrives at the CMTS just after it has started the scheduling algorithm.

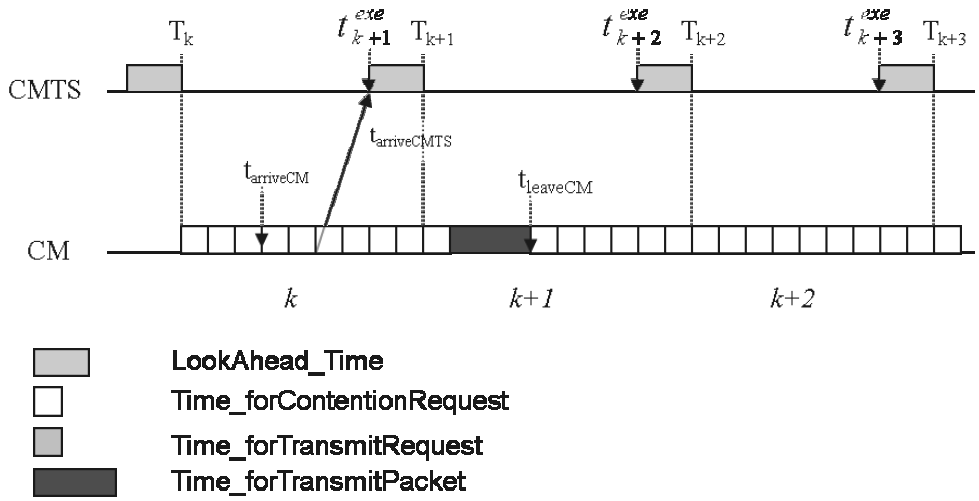


Figure 3.10: Maximum queuing delay for packet from BE SF.

MAP k is filled only with contention transmission opportunities. Thus the maximum time before a request is generated is the size of a contention request opportunity or $Time\_tillRequest(max) = 0.1ms$.

In the example, as can be seen from Figure 3.10, the maximum $Time\_tillScheduling$ is given by

$$\begin{aligned} Time\_tillScheduling &= t^{exe}_{k+2} - t_{arriveCMTS} \\ &= (T_{k+2} - t_{LAT}) - t_{arriveCMTS}. \end{aligned}$$

$t_{arriveCMTS}$ can be expressed relative to the $k+1$-th MAP starting time as

$$t_{arriveCMTS} = T_{k+1} - t_{LAT}.$$

Or,

$$\begin{aligned} Time\_tillScheduling &= (T_{k+2} - t_{LAT}) - (T_{k+1} - t_{LAT}) \\ &= \Delta T. \end{aligned}$$

Recall that ΔT is one MAP length. Thus the maximum $Time\_forRequest$ is calculated from

$$\begin{aligned} Time\_forRequest(max) &= Time\_forTransmitRequest(example = 0.1ms) + \\ &\quad TransmissionDelayCMtoCMTS(example = 0.2ms) + \end{aligned}$$

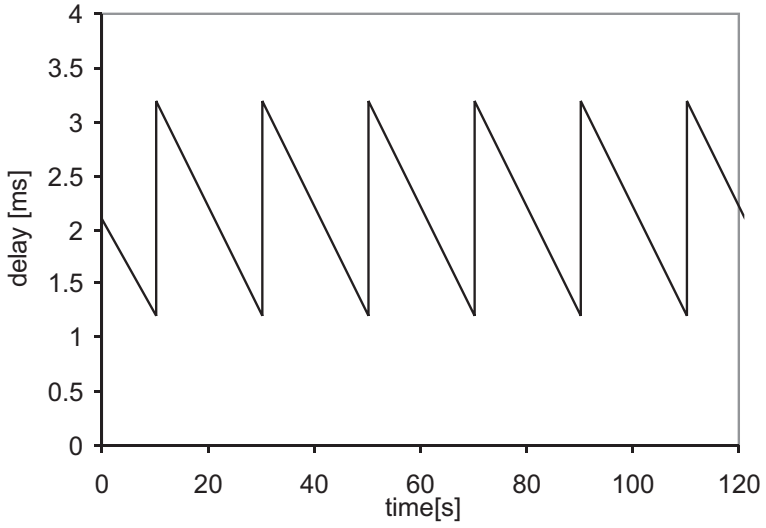


Figure 3.11: Queuing delay of the upstream packets reported via contention request

$$\begin{aligned} & Time\_tillScheduling(example = 2ms) + \\ & t_{LAT}(example = 0.2ms) \\ = \; & 2.5ms. \end{aligned}$$

Now we have the values of all the components contributing for the maximum queuing delay

$$\begin{aligned} \Delta t_{max} &= Time\_tillRequest(0.1ms) + Time\_forRequest(2.5ms) + \\ & DataGrantOffset(0.1ms) + Time\_toTransmitPacket(0.5ms) \\ &= 3.2ms. \end{aligned}$$

Figure 3.11 shows the delay of the packets. They experience different delays bounded by the minimum and maximum values. As is seen from Figure 3.11 these values are met. The delay experienced by the packets from the BE SF in this example is lower than the delay of the packets of the rtPS SF in the example from Section 3.2.1. There the biggest contribution to the maximum delay was from "Nominal_Polling_Interval", which was 10 ms. The BE SF profits from the many contention opportunities in a MAP, which allows it to transmit a successful request with a much smaller delay than the polling interval. Of course, when there are more active SFs, the maximum delay of the packets from

the BE SF is unbounded and no bandwidth is guaranteed, while an rtPS SF receives a minimum bandwidth.

3.2.3 Contention Resolution Algorithm

To verify the contention resolution algorithm a scenario has been established where there are four active BE SFs. Collisions are provoked by issuing packets at the four SFs synchronously. The traffic source at the SFs generates packets with size 8 mini-slots and inter-arrival time of 100ms, that ensures that no piggy back mechanism starts. The back-off start value is set to 2 which implies that 1 out of $2^2 = 4$ transmit opportunities will be randomly chosen by a SF. Assuming a perfect random generator, each opportunity should be equiprobable. In each MAP there are at least four contention transmit opportunities, thus, the successful requests can be sent and arrive at the CMTS in one frame. If collision has occurred, however, the SF will become aware of it only in the following MAP when it does not receive a confirmation for successful request or a grant. So the success requests can be identified from the SFs as the ones with delay less than a MAP length. If the packets are small there will be a gap in the packet delays of almost a MAP length between the packets for which the requests succeeded after the first attempt and the ones which succeeded after the second and so on.

To verify the results the probability for SF to issue a request, which is not subject to a collision can be analytically calculated according to

$$p_{NoColl} = p_{TO}(1 - p_{TO})^{N-1} n_{TO}$$

with p_{TO} being the access probability for a dedicated transmit opportunity (TO), N - the number of SFs and n_{TO} - the number of transmit opportunities. In the given scenario the number of SFs is N=4, the number of transmit opportunities for the first attempt is $n_{TO} = 4$. To obtain the access probability p_{TO} is applied that each transmit opportunity is equiprobable. So, the access probability is $p_{TO} = 1/n_{TO} = 0.25$. With this considerations, the probability for SF to issue a request, which is not subject to a collision is as follows:

$$p_{NoColl} = 0.25 * (1 - 0.25)^{4-1} * 4 = 42.2\%$$

The statistical results of the simulation are depicted in Figure 3.12. The chart shows the frequency of packets applied over its queuing delay, i.e. the number of packets with queuing delay in an interval with granularity 0.2ms. The first four bars of the histogram

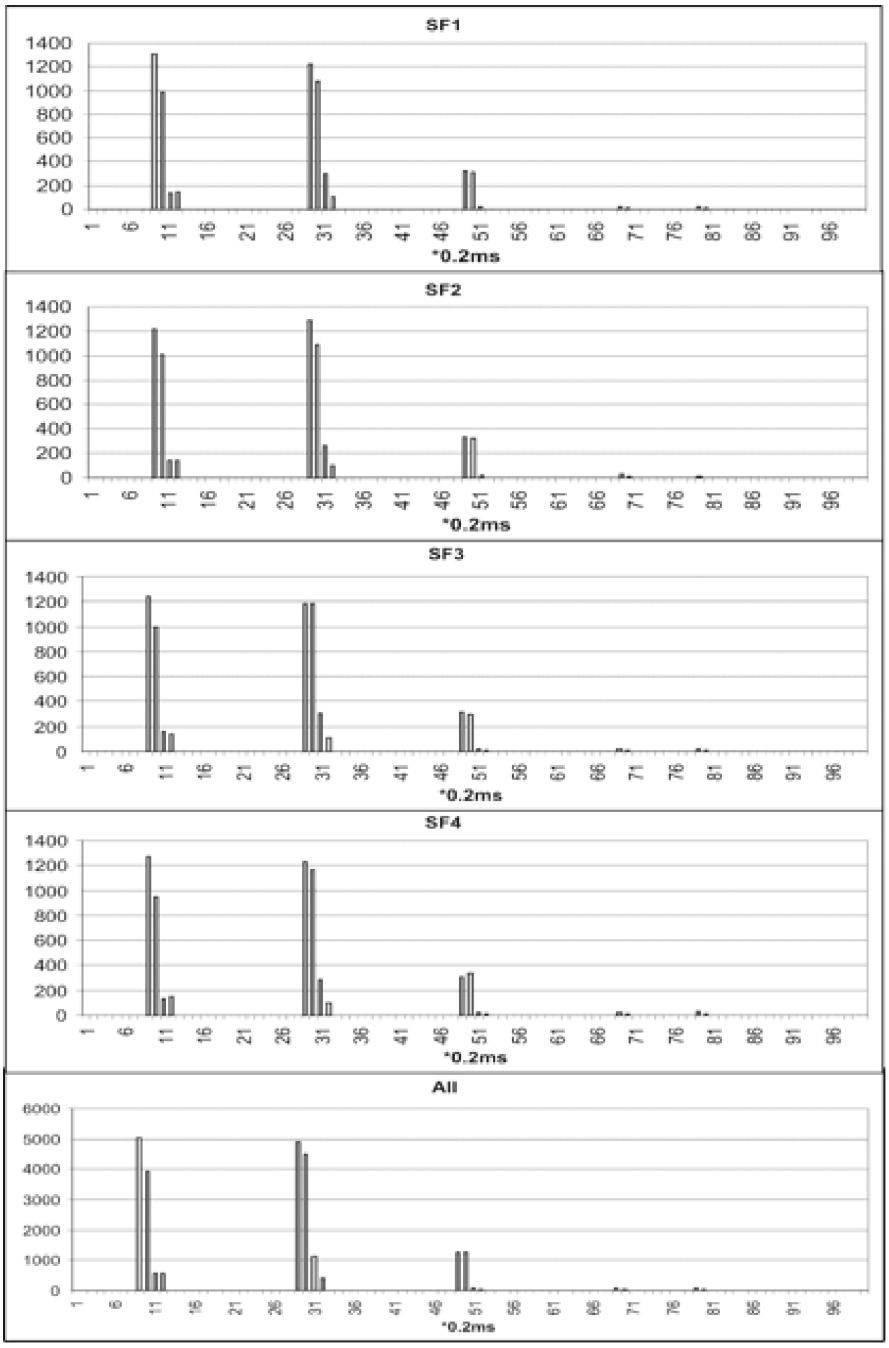


Figure 3.12: Histograms of the packets queuing delay

Table 3.3: Evaluated Simulation Results

SF #	success	trials	Pr
SF1	2576	5990	43.01%
SF2	2511	5990	41.92%
SF3	2528	5990	42.20%
SF4	2494	5990	41.64%
All	10109	23960	42.19%

indicate packets with an queuing delay between 1.69ms and 2.09ms. This is due to the fact that these packets are not subject to any collisions and, as mentioned above, are sent in one frame. The values of these four bars are extracted in Table 3.3 and it can be stated that simulation results are in accordance with the expected results.

3.3 Performance Evaluation of rtPS US Scheduling Service Type

The rtPS scheduling service type is designed to be used for video and gaming traffic. As was already discussed in Section 3.1.2 this service is characterized with regular polling opportunities in which the SFs can report their queue state. The key service parameters are the 'Nominal Polling Interval', which is the interval of the polls and the 'Tolerated Poll Jitter'.

3.3.1 Description of the Scheduling Algorithms

Even though the importance of video and gaming traffic has increased significantly, there are few standard conforming schedulers discussed in the literature. An example is [24] which addresses rtPS scheduling. It follows explicitly the standard description. More specifically, the rtPS scheduler described in [24], to which further on in the chapter will be referred to as "common", has exactly two queues which are served with strict priority. One for the uni-cast polls and one for the data grants. The scheduler keeps a timer which expires each $NominalPollingInterval$ seconds and forces a grant for an unicast request to be added to the polls' queue. This queue is scheduled with high priority. The maximum throughput of a system served with this rtPS scheduler, in the

absence of active UGS flows can be calculated as

$$T_{common} = C - s_r \sum_{i=1}^{N_{rtPS}} \frac{1}{NominalPollingInterval_i} - C_c, \tag{3.6}$$

where C is the total upstream capacity, s_r is the size of a request, N_{rtPS} is the number of rtPS SFs and C_c is the capacity of the contention channel. C_c can be easily calculated from the minimum number of contention request opportunities per MAP Nr_{min_c}

$$C_c = \frac{Nr_{min\_c} * s_r}{MAPsize}. \tag{3.7}$$

Note that in DOCSIS a flow is allowed to have only one outstanding grant. Thus until the request from a flow is not granted there is no actual need to poll this flow. Utilizing this fact an "improved" rtPS scheduler is constructed, which can improve the throughput of the "common" one. The "improved" rtPS scheduler also keeps two queues - polls' and grants' - and a timer. The polls' queue is always served with strict priority. When a rtPS flow becomes active a poll is added to the poll's queue and a timer dedicated to this flow is initialized with the value of the "NominalPollingInterval". Each time the timer expires a poll is added to this queue. However, when a data request is received the timer is deactivated. The scheduler keeps a variable $NextPollTime$ indicating the moment when the timer would expire if it is not deactivated. The scheduler then adds a data grant into the grants' queue. When this grant is scheduled at, say, time t_{Grant} the timer is activated again. If $t_{Grant} \geq NextPollTime$ a poll is scheduled immediately otherwise the timer is set to expire at $NextPollTime$.

With the described "improved" scheduler the generation of unnecessary polls is avoided. The simulation results in the following section will show that with this improved version the rtPS throughput can be increased and also that the packet delay is decreased.

3.3.2 Simulation Results

The simulated system has total US capacity of 40000 minislots/s or 5120Kb/s for minislots of 16 bytes. There are 17 active US rtPS flows. The parameters of the flow are $NominalPollingInterval = 10ms$ and $Jitter = 3ms$. The MAP size is dynamic and can vary between a minimum of 2ms and maximum of 4.625ms. The minimum number of contention request opportunities in a MAP is 1, unless stated otherwise. One such opportunity has size 4 minislots. From Equation (3.6) and Equation (3.7) one can

calculate the expected maximum throughput under the common algorithm to be in the range [3993.6; 4139]Kb/s. The throughput is defined as a range because the capacity of contention channel can vary.

Two scenarios are simulated, depending on the type of traffic generated from the sources, namely ON-OFF and constant bit rate (CBR). In the *CBR scenario* each CM has a server which transmits CBR traffic with a packet size of 1518 bytes or 106 minislots after all overhead, as described in Section 3.1.1, is accounted for. The packet inter-arrival time is determined from the desired load with all SFs contributing equally to it. In the *ON-OFF scenario* On Off traffic sources with variable frame size are considered. The distribution of the frame sizes is bounded pareto. Such distribution has three parameters a, L and U. a defines the shape, L the lower bound and U the upper bound. The probability density function of the bounded pareto distribution is

$$\frac{aL^a x^{-a-1}}{1-\left(\frac{L}{U}\right)^a}$$

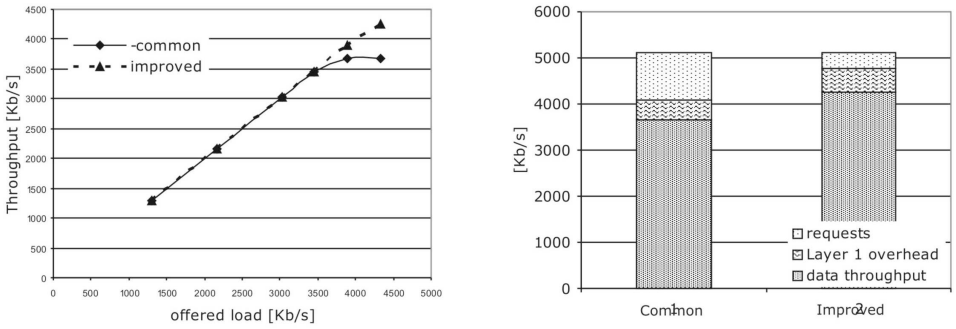
and the mean μ is given by

$$\mu = \frac{L^a}{1-\left(\frac{L}{U}\right)^a} * \left(\frac{a}{a-2}\right) * \left(\frac{1}{L^{a-2}} - \frac{1}{U^{a-2}}\right). \tag{3.8}$$

In the simulated scenario the parameters are $a = 1.8$, the maximum frame is $U =$ 4000 bytes and the minimum frame is 200 bytes. These parameters give average frame size of 3287bits. The ON times are exponentially distributed with mean 0.4s while the OFF times have mean 0.6s and are also exponentially distributed. The inter-arrival time of the frames in the ON periods depends on the desired load and varies between 6ms and 20ms. All simulation results are obtained after gathering statistics of at least 100000 packets.

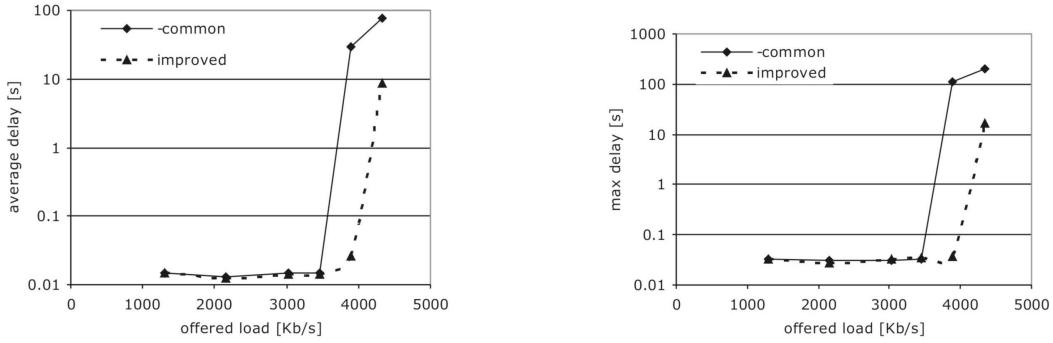
Results for CBR traffic

In Figure 3.13(a) the throughput versus the offered load for the two algorithms is shown. The offered load is calculated at the link layer i.e. the data load including only the UDP/IP/Ethernet overhead. This is the case for all scenarios in this chapter unless explicitly stated otherwise. The improved version of the algorithm achieves approximately 16.3 % higher maximum Ethernet throughput than the common one. The gain in the throughput is achieved by not scheduling the unnecessary unicast requests. This is demonstrated in Figure 3.13(b) where the utilization of the bandwidth in an over-



(a) (b)

Figure 3.13: a) The total Ethernet throughput b) Bandwidth utilization during overload simulations for CBR traffic

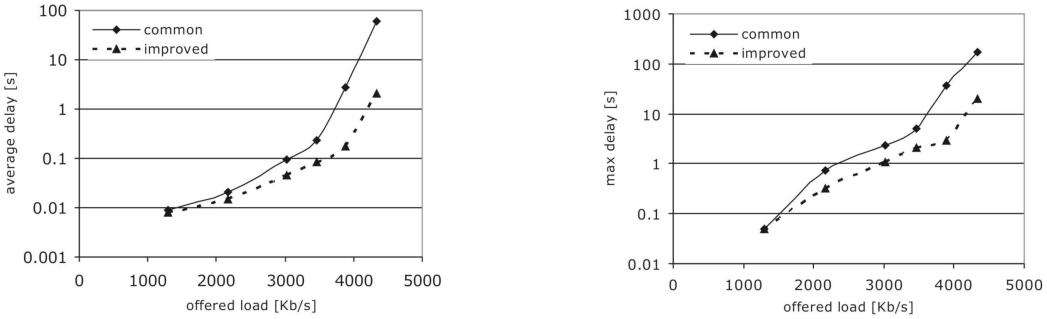


(a) (b)

Figure 3.14: The packet queuing delay a) average b) maximum values for CBR traffic

load scenario is shown. The request bar on the figure accounts for both unicast and contention request opportunities. The layer 1 overhead is 11.7% for both algorithms. The maximum layer one throughput for the 'common' algorithm is 4095 Kb/s and falls within the theoretical range.

At low loads both algorithms have similar packet delays which are shown in Figure 3.14. The delay experiences a sharp jump above 3665Kb/s for the 'common' algorithm. As seen from the throughput figure this the maximum Ethernet throughput for this setup. The maximum Ethernet throughput for the 'improved' algorithm in this scenario is 4261Kb/s.



(a) (b)

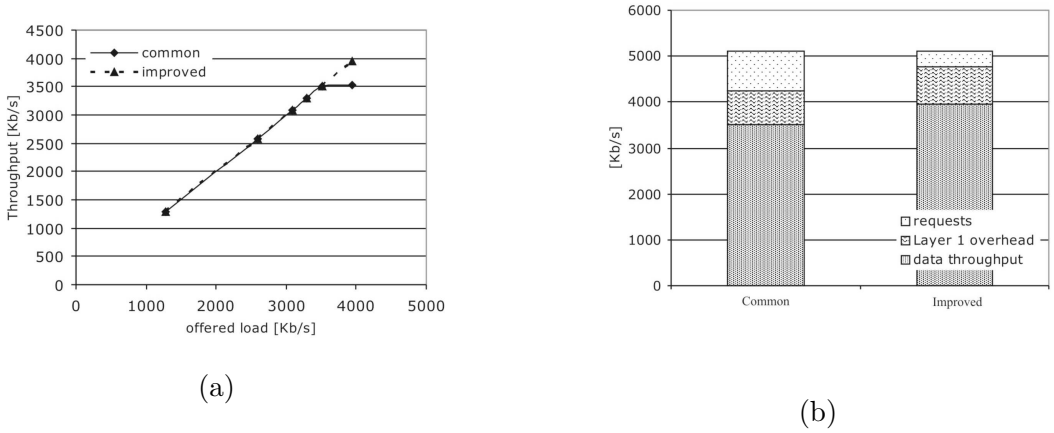
Figure 3.15: The packet queuing delay a) average b) maximum values for On/Off traffic

Results for ON-OFF traffic

In Figure 3.15 the average and maximum packet queuing delay vs the offered load is shown. In the load region below 3500Kb/s due to the variable arrival rate the schedulers have variable number of requests in their rtPS FIFO queue, in difference with the CBR traffic scenario. The amount requested varies and can take more than 10ms to schedule all requested bandwidth. Hence, some of the grants will be delayed so the uni-cast requests opportunities can be scheduled. Thus, with the increase of the load the number and the size of the requests to be scheduled increases also in this load region when the system is not saturated. As the load increases some requests will be delayed with more than one uni-cast request opportunity per flow. Again, as in the CBR traffic scenario, the system served with the "common" algorithm becomes faster saturated. This is in conformance with the throughput results shown in Figure 3.16 (a) which show that the improved algorithm achieves 13% more throughput than the common rtPS scheduling algorithm.

In Figure 3.16(b) the throughput in overload is shown. The main difference with the throughput in the scenario with CBR traffic is that here the Layer 1 overhead is significantly higher. This is due to the variable frame sizes. The lower the packet size the higher the overhead. The layer 1 overhead for the simulated scenario for both algorithms is 17.3%. As a result the maximum data throughput for the improved algorithm in this setup is 3945Kb/s while for the common algorithm it is 3500Kb/s.

In this subsection the performance of the rtPS US scheduling service type was evaluated in two traffic scenarios. It was shown that it is possible to design an rtPS scheduling mechanism which avoids scheduling unnecessary polls and is still in conformance with the standard requirements for this service. It does not increase the scheduling complexity



(a) (b)

Figure 3.16: The throughput a) vs. load b) in overload for On/Off traffic

and it was demonstrated that it achieves higher throughput and lower packet delay.

3.4 Performance Evaluation of the BE US Scheduling Service Type

To evaluate the performance of the BE scheduling service type and to establish advisory values for different parameters a simple FIFO scheduling algorithm is used, where each requesting SF gets fixed amount of bandwidth. The contention channel in DOCSIS can be constant or variable. We talk about variable when the scheduling algorithm assigns the unused minislots in a MAP for contention requests, otherwise it is constant. From the example in Section 3.2 is clear that variable contention channel has better performance. The contention channel parameters, which can be varied are the minimum number of contention opportunities *min_con* or the minimum size of the contention channel, the Data Back-off start and the Data Back-off end values.

The simulation setup consists of 80 active SFs only from the BE scheduling service with the same US priority and having the same traffic source parameters. The packet size of the data is constant 492 bytes. Adding all the headers of the different protocols (UDP/IP/Ethernet) results in 538 bytes. After the DOCSIS overhead is added the packet size is 42 minislots. There is no packet header suppression and piggybacking is allowed. The SF use ON-OFF source to generate packets with exponentially distributed ON and OFF times. The mean time in ON period is 4s and in OFF period - 1s. The mean packet inter-arrival time while in ON period is 134.5ms to obtain 40% load, 106.8ms for 50%, 90ms for 60%, and 82.8ms for 65%. The offered load is calculated by dividing the

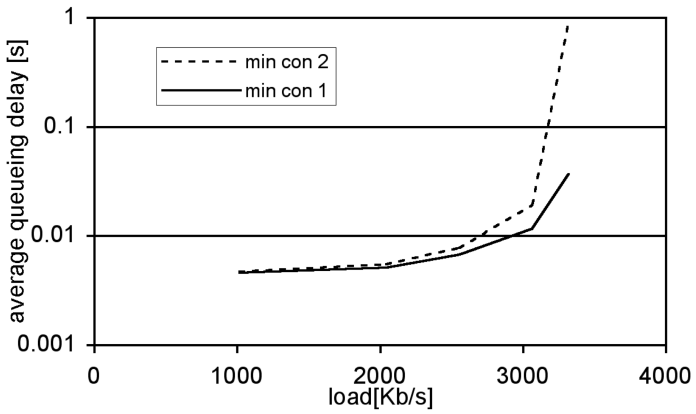


Figure 3.17: The average queuing delay in the upstream of an BE SF vs. the load

average traffic generated from a SF, expressed in minislots and multiplied by the number of active SFs, by the total available bandwidth which is 40000 minislots/s.

We first look at the influence of the minimum contention channel parameter *min_con*. In Figure 3.17 the average queuing delay i.e. the delay from the moment a packet enters the service flow queue until the last bit of the packet leaves it, versus the system load is given. The 80 active SF have equal traffic parameters. We have simulated the same scenario when the minimum number of contention opportunities per MAP is 1 and 2. One contention request opportunity is 4 minislots. As expected at average and low loads there is no noticeable difference in the delay as the MAP is filled with many contention opportunities. The difference is at higher loads where when there is only 1 contention opportunity per MAP the average delay is lower. Notice again the very high overhead in DOCSIS networks where around 65-70% of the bandwidth is utilized for Ethernet data traffic.

The results for the dependence of the average delay from the back-off start value for several loads is shown in Figure 3.18. The minimum number of contention requests per MAP is 1 and the Back-off end value is 10. From previous simulation, not shown here, was seen that this value does not have big influence on the results for the studied loads. With the increase of the Back-off start value the average delay also increases. This follows from the fact that a SF should wait on average longer to send a request. However an interesting finding from these simulations is that there is an optimal value for the back-off start which is the same for all studied loads - 2.The existence of such a value is due to the influence of two factors with opposite action: on one hand increased collision probability if the back-off start value is less than 2 and on the other hand the

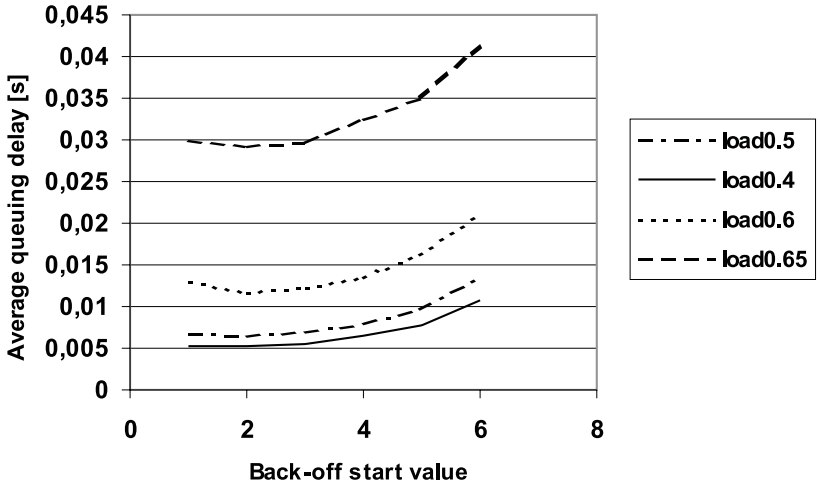


Figure 3.18: The average packet delay vs. the Back-off start value for 80 active BE SFs

delays get longer if the back-off start value gets bigger.

To further confirm this conclusion a scenario where the total load is the same but the number of active SFs changes is studied. The total offered load is 50%. In Figure 3.19 the results for three different numbers of active SFs are presented . For all the cases the minimum delay is achieved for back-off start value 2.

In Figure 3.20 the results for the same scenario (50%load, 80 BE SFs) but for MAP lengths of 2 and 10 ms are shown. As can be expected the average delay for the case when the MAP is 10 ms is higher than the one when it is 2 ms and the difference is approximately 8 ms. Remarkably the optimal back-off start value is 2 also when the MAP length is 10 ms.

3.5 Summary

In this chapter the implemented simulator for DOCSIS 2.0 based HFC network was described. In difference with other freely available DOCSIS simulators, it models in detail the physical layer overhead, the physical medium dependent layer and many detailed features of the MAC layer. This includes the unsolicited grant service, the real time polling service and the best effort service. The different polling mechanisms implemented are uni-cast request, bandwidth request in contention slot and via a contention channel. The polling mechanisms and the scheduling services are validated, including the implementation of the contention resolution algorithm, by analytically calculating

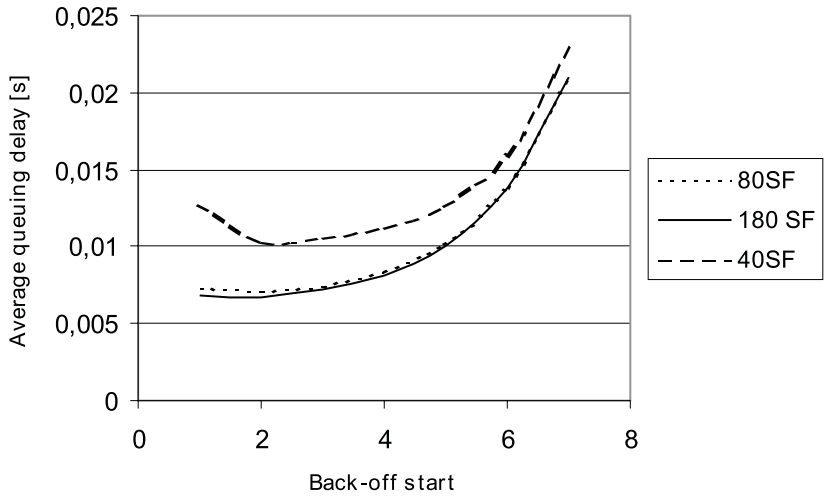


Figure 3.19: The average packet delay vs. the Back-off start value for 50% offered load.

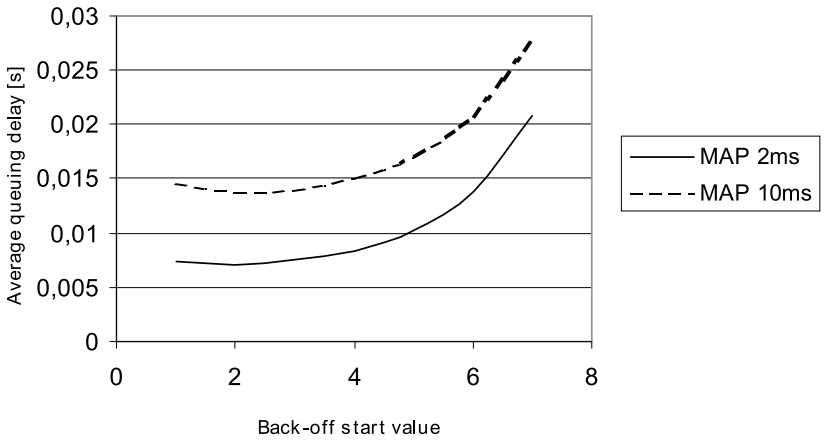


Figure 3.20: The average packet delay vs. the Back-off start value for 50% offered load and 80 active SFs.

the behavior for simple scenarios and comparing them with the simulation results for these scenarios.

The simulator implements three upstream scheduling service types. The UGS type, which is designed for applications which require constant delay jitter like voice traffic, provides regular unsolicited grants with bounded jitter. The rtPS scheduling type, which is designed to be used for video traffic, offers real-time, periodic, unicast request opportunities, which meet the flow's real-time needs and allow the SF to specify the size of

the desired grant. In the chapter, an improvement to the common rtPS scheduler has been proposed. It was shown that it is possible to design an rtPS scheduling mechanism which avoids scheduling unnecessary polls and is still in conformance with the standard requirements for this service. It does not increase the scheduling complexity and the simulator was used to demonstrate that it achieves higher throughput and lower packet delay.

The simulator was also used to study the influence of some DOCSIS parameters on the BE scheduling service type. The performance of this service depends on the contention channel in terms of its size and the contention parameters which are initial and maximum back-off window. The data back-off start value is the parameter communicated to the CM from the CMTS which is used by the former to set their initial back-off window. It was shown that when the back-off start value is 2 optimal results are achieved for all investigated scenarios, like different system load, number of competing flows, minimum contention channel size and MAP length. It was also confirmed that when the contention channel is variable the best choice for the minimum number of contention transmit opportunities per MAP is 1.

The simulator differs from other simulation models also in that it is implemented using the OMNET++ software tool. This tool has the advantages that it is free and offers a graphical user interface. This makes the simulator easy to use and suitable for use by operator companies to fine tune their systems. We have in fact used it, in a confidential study [107], to simulate and evaluate the performance of the actual scheduler implemented in Motorola BSR 64000 CMTS/Edge routers [60], which are commercially deployed by a cable operator [108].

Chapter 4

Packet Scheduling Algorithms for Networks with a Single Output Channel

The previous two chapters covered issues from upstream scheduling in subscriber access networks. In the downstream, typically high-speed routers using packet scheduling algorithms are utilized at the central office to perform the scheduling. This chapter discusses two such scheduling algorithms, which perform in constant time. The Last Backlogged First Served- Deficit Round Robin (LBFS-DRR) is a novel algorithm achieving significantly lower average packet delay than DRR and providing the same end-to-end delay bounds and fairness. The ideas behind the second algorithm, the Surplus Round Robin, were already proposed in [67]. In this chapter, an implementation for the algorithm is proposed and it is proven that it provides an end-to-end delay bound lower than DRR and has the same fairness. Both algorithms are also evaluated via simulations.

4.1 Background and Overview

In high-speed packet networks packets from millions of flows, originated from different users and pertained to different types of applications contend for the network resources. It is the task of the scheduling algorithm at the network nodes to arbitrate the transmissions of the packets on the output link. Recall from the introduction, Section 1, that a scheduling scheme applicable to high-speed networks should have low complexity - computational as well as in implementation. The complexity is evaluated by the number of operations the algorithms has to perform to reach the scheduling decision. An algorithm

is said to have an O(1) complexity if the worst-case number of operations needed to select the next packet is constant with respect to the number of flows.

Another desired feature for scheduling algorithms is that the flows should be treated fairly. A suitable measure for the fairness of an algorithm is the modified Golestiani fairness, which was discussed in Section 1.3.3 and is defined as the maximum bound on the normalized service provided to any two flows over any backlogged period.

It is required from a scheduler that it isolates the flows, that is flows shouldn't be able to degrade the service of other flows, to the extent that the performance guarantees are violated.

A packet scheduler is expected to provide low and in some cases bounded, average and maximum queuing delay. The end-to-end delay is the most assessed measure for the user experience. Moreover in a work-conserving system low packet delays imply low buffer requirements. Thus, the choice of the scheduler may directly affect the cost of the implementation in terms of the required memory. A metric to evaluate the worst-case performance of a scheduler is given from the Latency -Rate theory as discussed in Section 1.8. It introduces the notion of scheduler's *latency* and relates it to the flow's reserved rate. Besides that it is useful as a metric to compare and classify schedulers, from the latency, one can easily obtain a delay bound provided the arrival process is bounded like leaky-bucket shaped traffic.

There are very few schedulers implementing per-flow queuing with effectively O(1) complexity, which fulfil the requirements for fairness, bounded delay and isolation of the flows. Such is the Deficit Round Robin (DRR) scheduler, discussed in Section 1.2.2, which is also implemented in several routers. The Priority DRR algorithm, proposed in [69] and [70] combines First-Come First Served (FCFS) queuing with DRR. It provides very low average latency for flows transmitting at rates lower than their estimated fair share of the link rate. Other per-flow queuing schedulers proposed in the literature either failed to achieve fairness [63], [64] (see the discussion in Section 1.2.2 or in [66]) or have higher complexity [49], [66], [71], [75], [77], [76], [78], [79], which for some depends on the number of flows being scheduled [25], [44], [26], [23], [45], [48] , [59], [74], [69] as was discussed in Section 1.2. From them the Worst-Case Fair Weighted Fair Queuing + (WF^2Q+) achieves the closest service to the Generalized Processor Scheduling (GPS) discipline (see Section 1.2).

An inherent drawback of the DRR is the high average and maximum packet delay and correspondingly file transfer delays. A possible scheduling discipline to address this drawback and which does not require sorting of flows is Last-Come First-Served (LCFS). The stability and asymptotic of LCFS scheduling in its pre-emptive (PR) and non pre-

emptive version has been extensively theoretically studied [109]. While the FCFS policy can be directly applied to packet scheduling by replacing jobs by packets, this is however not true for the LCFS. The packets from a flow can not be reordered, thus LCFS is not directly applicable for scheduling in packet telecommunication networks. It can however, be suitably modified, to serve first the last flow which becomes backlogged resulting in Last Backlogged - First Served discipline as will be described in this chapter.

The chapter is organized as follows: In the following section a novel packet scheduler with O(1) complexity, the Last Backlogged - First Served Deficit Round Robin is described and the pseudo code is discussed. Section 4.3 describes the Surplus Round Robin algorithm and proposes a novel implementation method for this scheduler. Both algorithms are evaluated theoretically by calculating their latency and fairness in Section 4.4. In the following section simulation results from several scenarios are presented. The results are discussed and summarized in the last section.

4.2 Last Backlogged First Served- Deficit Round Robin (LBFS-DRR) scheduling algorithm

As the DRR algorithm, LBFS-DRR also assigns for each flow i, a quantum Q_i, which indicates the portion of the resources the flow should get in a round robin cycle. Also, to each flow i, is associated a counter, called deficit counter DC_i, which indicates the amount of service the flow can still receive in a round. The LBFS-DRR selects for service the last flow to become backlogged in a round and serves it until either another flow becomes backlogged or it has used up its quantum. In a round all flows receive service proportional to their weight thus assuring that each backlogged flow receives its fair share of the bandwidth.

Before describing in detail the pseudo-code, this section clarifies the algorithm on the basis of the example given on Figure 4.1. In the upper part of the figure, the arrival sequence of the packets of three flows is shown. In the lower part of the figure, the output that will be generated by the proposed LBFS-DRR scheduling algorithm and by the DRR algorithm is depicted.

A packet is considered arrived when its last bit has arrived at the scheduler. If two packets arrive at the same time the scheduler considers that the packet from the flow with lower identification number has arrived first. Further on for clarity, if a packet arrives at certain moment in time t_k it is considered arrived just before this moment of time t_k^-. Each flow has a quantum $Q_i = 1000$ bytes. The packets from flows 1 and 2 have

length of 1000 bytes. The packets from flow 3 have the following lengths $p1 = p2 = 500$ bytes and $p3 = 1000$ bytes. The link capacity is such that for 10 time units thousand bytes are transmitted, i.e., if the transmission on the link of a packet of 1000 bytes starts at t_0 it will finish at time t_{10}.

On the example at time t_0 there are two active flows in the system and each has one packet to send. The scheduler visits them according to the round robin order and as a result in round 1 the two flows send one packet each, which is also of the size of their quantum. In the first round, where there are only backlogged flows in the system, the two schedulers do not differ from each other. At time t_{10}, second packets arrive for the first and second flow, thus they both are backlogged at the start of round 2, which is at time t_{20}. At this moment of time a packet of flow 3 arrives having length 500 bytes. Thus when the LBFS-DRR scheduler decides which flow can send the next packet at t_{20} it chooses the newly backlogged flow 3. This is different from the DRR scheduling algorithm which will insert the newly backlogged flow at the end of its $BackloggedFlows$ list and at the start of the second round it will schedule for service the next flow in the round robin order, which is flow 1. After flow 3 sends the packet its queue is empty and the LBFS-DRR scheduler choses the next flow from the round robin order, which in this case is flow 1.

At time t_{35} a packet from flow 3 arrives. The flow still has not used its quantum for round 2 and thus is still eligible for service. Consequently at time t_{35} flow 3 can transmit the packet and its deficit counter is reduced with its length to become $DC_3 = 0$. This is another advantageous difference of the LBFS-DRR, that provided there is enough deficit left, it visits a flow multiple times in a round. At time t_{40} the next one in the round robin order, which is flow 2 is selected for service. In the DRR scheduling discipline at this time the first packet for the newly backlogged flow will be sent. This gives difference in the delays for this packet in the two disciplines of almost one round.

When the third packet arrives for flow 3 with length $p3 = 10$, the scheduler is still in round 2 and the remaining deficit for flow 3 is zero. This is not sufficient for $p3$ to be sent in this round, thus flow 3 is added to the list for the next round after flow 1. Note that flow 1 has already used up all its deficit before packet $p3$ arrived for flow 3 in round 2 and thus has already been added to the list for the next round. At time t_{50} all the flows have used up their quantums thus there are no more eligible flows to be selected for service in this round and the LBFS-DRR algorithm begins with scheduling the next round.

In round 3 all flows are backlogged under both algorithms. They are scheduled in the same order as in round 2, which is different for both algorithms

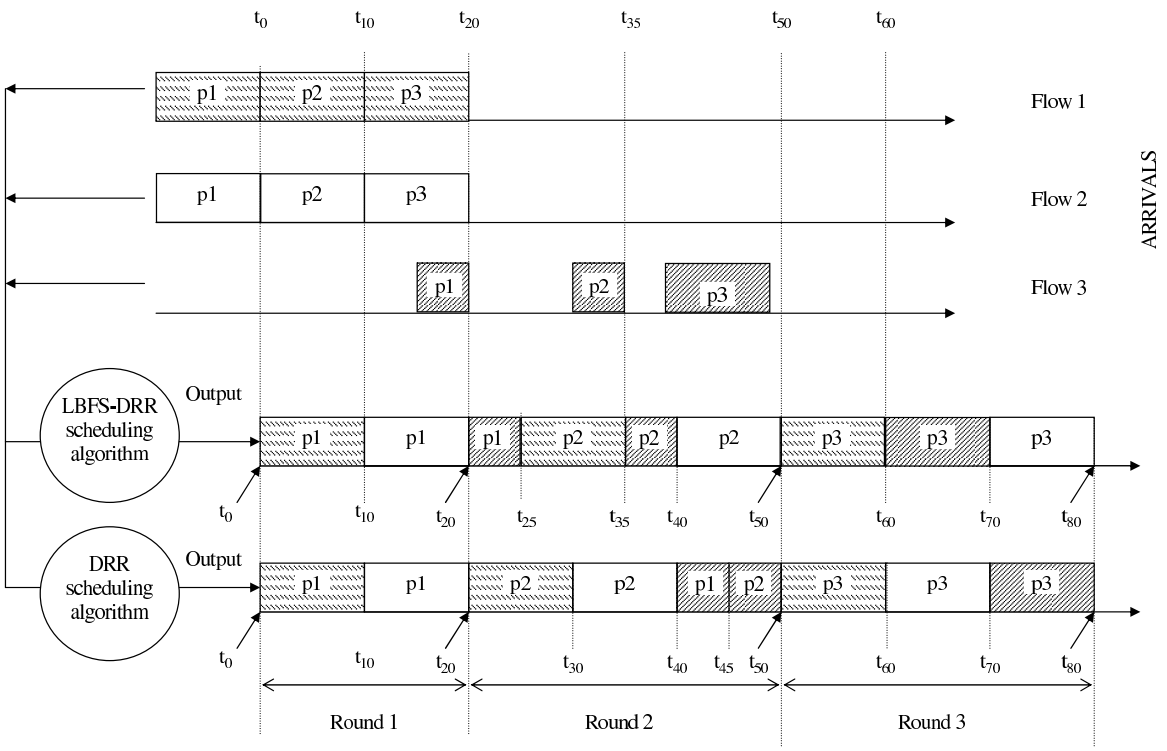


Figure 4.1: Example of LBFS-DRR scheduling

The example demonstrated that the new algorithm can have different order of service of the contending flows. Through this it succeeds to naturally spread the quantum within a round. This brings advantages like orders of magnitude improvement of the expected delay under the LBFS-DRR in comparison with DRR and also reduces the buffer size required. These results are more pronounced for high guaranteed rates and will be presented in Section 4.6.

Below, a detailed explanation of the implementation of LBFS-DRR is given, starting with a summary of the variables needed.

4.2.1 Variables

In order to track the service received from a flow in a round, LBFS-DRR needs to distinguish consecutive rounds. Thus besides the DRR flow variables Q_i and DC_i, it keeps two *BackloggedFlows* lists. One for the flows which should receive service in the current round as in DRR, the other one for the flows which already have used up their deficit counter for the current round and are still backlogged. The variables for the algorithm are summarized in Table 4.1. When the current round *BackloggedFlows*

Table 4.1: LBFS-DRR VARIABLES

2 $BackloggedFlows$ lists - lists with backlogged flows.
RC- rounds counter
Per flow:
$Queue_i$, Q_i, DC_i - DRR variables ;
LRS_i - the last round the flow received service

Table 4.2: LBFS-DRR ENQUEUE PROCESS

```
1. i=p.Flow();
2. Queue_i.Insert(p);
3. IF(i not in any list)
4.    if(LRS_i == RC)
   /*The flow has already received service in this round.*/
5.       IF(Length(p) ≤ DC_i)
6.          CurrentActiveList.push-front(i);
7.       else
8.          DC_i+ = Q_i;
9.          NextActiveList.push-back(i);
10.   else
   /*The flow hasn't received any service in this round.*/
11.      DC_i = Q_i;
12.      CurrentActiveList.push-front(i)
```

list becomes empty the two list are switched. Note that this is an O(1) operation. The algorithm keeps a global Round Counter RC which counts the rounds passed thus giving a sort of identification to a round. Each flow stores this identification in a variable Last Round Served LRS_i.

The algorithm can be divided in two processes dedicated to the enqueuing and to the dequeuing of a packet.

4.2.2 Enqueue Process

The pseudo code of the enqueue process is given in Table 4.2. When a flow becomes backlogged it is examined whether it has already received service in the current round by comparing its LRS_i variable with the RC. In case it has, the enqueue process continues further to check whether the flows deficit counter is sufficient for the packet. If not the flow is inserted in the next active list to receive service in the next round. This is an

Table 4.3: LBFS-DRR DEQUEUE PROCESS

```
13. WHILE (Any Active list NOT empty)
14.   IF(CurrentActiveList NOT empty)
15.         flow-iterator = Begin(CurrentActiveList);
16.         i= flow-iterator.flow;
17.    else
18.         swap(ActiveList1 and ActiveList2);
19.         RC++;
20.         break;
21.    p=Head(Queue_i);
22.    send(p);
23.    DC_i = DC_i − p.Length();
24.    IF(Queue_i.empty()
25.       LRS_i = RC
26.       CurrentActiveList.erase(flow-iterator);
27.    else /*the queue is not empty*/
28.       IF(p.Length() > DC_i) /* the DC is over. */
29.          DC_i+ = Q_i;
30.          CurrentActiveList.erase(flow-iterator);
31.          NextActiveList.push-back(i)
```

important check as it insures that no flows with insufficient counter are found in the list which is necessary in order to achieve O(1) complexity. If the flow is found to have sufficient deficit for the current round it is inserted at the front of the active list. This ensures the LBFS part of the algorithm.

4.2.3 Dequeue Process

The pseudo code of the dequeue process is given in Table 4.3. Before discussing it let us mention for the reader without programming knowledge that a list is a data structure of elements, in which each element, besides the data itself keeps a pointer to the next one, i.e., the address in the memory where the next element is kept. Thus an element is determined by its address, also called *iterator*, and it can be removed or inserted in a constant time from any position in the list if the *iterator* is known.

The dequeue process selects the head of the current active list for service. After a packet is transmitted and the DC of the flow updated, the scheduler has to check whether the flow is still eligible for transmission in the current round. This includes that the flow is backlogged and the DC is sufficient for the next packet in the flow's

queue to be transmitted. Otherwise the flow is erased from the current active list. The LBFS-DRR algorithm works with the *iterators* and not with the actual elements. This provides the flexibility to manipulate elements not only at the start or end of the list. Flow i was selected from the head of the list. If while the packet from the flow was transmitted (line 22), another flow became backlogged, flow i would no longer be at the head of the list. However because the algorithm knows the flows' *iterator* it can still access flow i list element (line 26 or line 30) in constant time.

4.2.4 Discussion

In the proposed implementation the different rounds are distinguished by keeping two lists. There are other possibilities like for example the one realized in [64]. There, a counter of the elements in the list is kept. However, this solution is less suitable for LBFS-DRR, as the number of flows is highly variable during one round.

To demonstrate the advantages of the LBFS scheme the DRR algorithm is used as the round robin scheduling but it can be replaced with some other RR scheduling algorithm like for example SRR [67] or ERR [64].

The algorithm can be easily modified in a simpler non-quantum-preemptive version such that, when a flow is chosen for service it is popped from the list and served until it has used its quantum or is no longer backlogged. In other words no flow can pre-empt the service of a flow in a round as long as the later has not exhausted its DC.

The complexity of the proposed algorithm is O(1). As it is seen from the pseudo code all the operations used are performed in constant time. Further on after a packet is sent the scheduler checks the eligibility for another packet to be sent from the flow. When the flow is not eligible it is removed from the current active list. When a flow is to be added to the active list its eligibility is also checked, so no ineligible flows are inserted in the current active list.

In a round all flows served by LBFS-DRR receive service proportional to their weight thus assuring that each backlogged flow receives its fair share of the bandwidth. It will be proven that the algorithm is fair and provides rate guarantees in Section 4.4 when the theoretical evaluation of single channel schedulers is discussed.

4.3 Surplus Round Robin (SRR) scheduling algorithm

In [67] an algorithm called Surplus Round Robin (SRR) was proposed. It is very close to the DRR, but has the advantage that it does not need the size of the next packet in order

to make the scheduling decision. Later on in [68] the same authors further developed their idea and proposed a stripping algorithm which can perform SRR. In this section an implementation of SRR as a scheduling algorithm is proposed.

We found by theoretical analysis, which will be described in the next section, that SRR has lower latency and consequently a lower maximum packet delay bound than DRR. As one of the few fair O(1) complexity scheduling algorithms we have considered it important to further study the algorithm by proposing an implementation and performing simulations to demonstrate its performance.

To each flow i is associated a quantum Q_i proportional to the flow's weight w_i and a counter called surplus counter SC_i, which indicates the amount of service the flow should still receive in a round. When a flow becomes backlogged its surplus counter is set to its quantum if the flow has not received service in this or the previous round. The backlogged flows are serviced in a round robin order. Upon a visit to a flow a packet is transmitted provided that the surplus counter is not negative. After a packet is sent SC_i is decreased with the size of the packet. If as result of a packet transmission a flow obtains a surplus amount, i.e., SC_i becomes non-positive, it is penalized by this amount in the next round, regardless whether the queue became empty or not after the packet was transmitted. The last condition is important because of the unfairness that might arise if a flow is not penalized. Consider the scenario, where the packets of flow i arrive in bursts of $Q_i + SC_i$ bytes each round. When these bytes are transmitted the queue of the flow becomes empty. If the scheduler does not penalize the flow during the next round, it will be allowed to send each round above its quantum. Thus the SRR algorithm cannot be implemented as a straightforward extension of DRR. Because the value of SC_i is not reset to 0 after packet transmission leaves the queue empty, a flow can be visited multiple times in around.

If a flow becomes backlogged in a round in which it has already received surplus or it has received surplus service in the previous round the surplus counter is increased with the quantum. In this way the surplus amount is not lost. Otherwise the surplus counter is set to the quantum.

The SRR algorithm keeps the same variables as the LBFS-DRR with the difference that the deficit counter DC_i is referred to as a surplus counter SC_i. Again if the minimum quantum, say Q_{min} is chosen not less than the maximum packet length in the network then the algorithm has $O(1)$ complexity. Hereafter, a possible implementation is proposed, which would ensure that the algorithm is fair and with bounded latency.

Table 4.4: SRR ENQUEUE PROCESS

```
1. i=p.Flow();
2. Queue_i.Insert(p);
3. IF(i not in list)
4.    IF(LRS_i == RC)
   /*The flow has already received service in this round.*/
5.       if(SC_i > 0)
6.          CurrentActiveList.insert(i);
7.       else
8.          SC_i+ = Q_i;
9.          NextActiveList.insert(i);
10.   ELSE IF(LRS_i == RC − 1)
/*The flow has received service in the previous round.*/
11.         SC_i = min(0, SC_i) + Q_i
12.         CurrentActiveList.insert(i);
13.   ELSE
   /*The flow hasn't received any service in this or in the previous round.*/
14.      SC_i = Q_i;
15.      CurrentActiveList.insert(i);
```

4.3.1 Enqueue Process

When a packet of flow i arrives, it is inserted in the queue i. If the queue was empty before this arrival, the algorithm selects to which list to add the flow, based on when it has last received service and possibly how much. Provided that it has received service in the current round and it still has a positive surplus counter, it is added to the currently active list. Otherwise, it is added to the list for the next round and its SC is increased with its quantum. Performing this check on packet enqueue, ensures that when the scheduler selects a flow for service, it will be allowed to transmit at least one packet. Provided that the flow has received service in the previous round it is eligible for service in the current round but any penalty from the previous round should be accounted for. This is reflected in the enqueue pseudo code in Table 4.4 on line 11. And finally if the flow did not receive service in the current or previous round, its surplus counter is initialized to its quantum and added to the current active list.

4.3.2 Dequeue Process

The dequeue process of the algorithm, given in Table 4.5, selects flows for service from the currently active list. When it becomes empty, the pointer for current active list is

Table 4.5: SRR DEQUEUE PROCESS

```
16. WHILE (true)
17.   IF(CurrentActiveList NOT empty)
18.         i=CurrentActiveList.popHead);
19.    else
20.         swap(ActiveList1 and ActiveList2);
21.         RC++;
22.         continue;
23.    p=Head(Queue_i);
24.    WHILE(SC_i > 0 AND Queue_i NOT empty)
25.      send(p);
26.      SC_i = SC_i − p.Length();
27.    IF(Queue_i NOT empty)
28.      SC_i+ = Q_i;
29.      NextActiveList.insert(i)
30.    ELSE
31.      LRS_i = RC
```

swapped with the one for the next active list and the rounds counter RC is increased. When a flow is selected for service from the dequeue process, its packets are transmitted on the link as long as its surplus counter is positive. After each packet is transmitted the counter is decreased with the packet size. When the flow is no longer eligible for service and it still has packets to transmit, its surplus counter is increased with its quantum and it is added for service for the next round. If it has no more packets to transmit it records the current round in its Last Round Served (LRS) variable. The SC is not initialized in order to allow that the service for the flow resumes, if it becomes backlogged again in the current round.

It is straightforward to see that the proposed processes are performed in constant time, i.e, the algorithm has O(1) complexity.

4.4 Analysis of packet scheduling algorithms for single channel

In this section the performance of the two round robin algorithms, namely the LBSF-DRR and the SRR is evaluated by calculating their latency and fairness and comparing it with DRR. In Sections 1.3.2 and 1.3.3 the ideas behind these metrics were outlined. The latency gives the longest period of time a flow has to wait before receiving service at its

guaranteed rate. To calculate the latency of a scheduler one has to show that bandwidth scheduled for a flow in a busy period can be bounded and calculate this bound. The fairness measures the maximum difference in the normalized service scheduled for two flows after a time instance when they have infinite backlog of packets. Both metrics are useful for the comparison of scheduling algorithms and are known for a variety of schedulers as was summarized in Table 1.1 in Section 1.3.4.

In the next subsection some necessary preliminaries are outlined, which give relations between the parameters and are valid for both algorithms.

4.4.1 Preliminary

Consider n flows contending for a shared link with rate r and a DRR based scheduler, which arbitrates the packet transmissions from the flows. To each flow a weight w_i is assigned or a rate r_i is reserved such that

$$\frac{w_i}{\sum_{j=1}^{n} w_j} = \frac{r_i}{\sum_{j=1}^{n} r_j}. \tag{4.1}$$

A DRR-based scheduler can be for example DRR, SRR or LBFS-DRR. Such a scheduler is characterized by a quantum Q_i for each flow i. The quantum indicates the share in bytes, which a flow is entitled to receive in a round if there is no deficit or surplus to be compensated from the previous round. The quantum typically is proportional to the flow's weight/rate and can be expressed as

$$Q_i = w_i Q_{min}, \tag{4.2}$$

where Q_{min} is the minimum possible quantum, hence $w_i \geq 1$, for all i. A typical choice is the maximum packet length on the network, for example if the scheduler is for an Ethernet network $Q_{min} = 1518$. This guarantees that when a flow is selected for service at least one packet will be sent. The sum of the quantums of the contending flows determines the length of the round called the frame f. It is given by

$$f = \sum_{i=1}^{n} Q_i = \sum_{i=1}^{n} w_i Q_{min}. \tag{4.3}$$

Note that the ratio of a quantum Q_i to the frame size f can be expressed by replacing Q_i from Equation (4.2), f from Equation (4.3) and accounting the weight-rate relation

from Equation (4.1)

$$\frac{Q_i}{f} = \frac{w_i Q_{min}}{\sum_{j=1}^{n} w_j Q_{min}} = \frac{r_i}{\sum_{j=1}^{n} r_j}. \tag{4.4}$$

In order to be able to guarantee the assigned rates r_i, they should of course at least be bounded

$$\sum_{i=1}^{n} r_i \leq r. \tag{4.5}$$

This will be presumed in the following analysis. Further on these rates are referred to as guaranteed or reserved rates. f can be expressed from Equation (4.4)

$$f = \frac{Q_i}{r_i} \sum_{j=1}^{n} r_j \leq \frac{r}{r_i} Q_i = F, \tag{4.6}$$

where F designates the maximum frame size for n contending flows with rates bounded by Equation (4.5). Thus the relation between quantums and the reserved rates is determined by

$$\frac{Q_i}{F} = \frac{r_i}{r}. \tag{4.7}$$

The defined minimum quantum determines a minimum rate

$$\frac{Q_{min}}{F} = \frac{r_{min}}{r}. \tag{4.8}$$

Dividing Equation (4.7) by Equation (4.8) and accounting for Equation (4.2), one obtains for the weights

$$w_i = \frac{r_i}{r_{min}} \tag{4.9}$$

Considering the above outlined relations, the chapter proceeds further with the calculation of the latency and fairness of the LBSF-DRR and the SRR algorithms.

4.4.2 Latency bound of LBFS-DRR

In order to calculate the latency of the algorithm it is sufficient to bound the service a flow receives in a backlogged period as we discussed in Chapter 1.3.2. The flow's service, referred also as the work done by the scheduler for the flow or the bytes transmitted on

the shared link from a flow, is tracked by the flow's deficit counter. The deficit counter of a backlogged flow at the end of a round is bounded, with bounds given by the lemma proven in [91]

Lemma 4.4.1. *The deficit counter of a backlogged flow i at the end of a service from a LBFS-DRR scheduler is bounded by*

$$0 \leq DC_i \leq L_{max} - 1 \tag{4.10}$$

where L_{max} indicates the maximum packet length.

Hereafter the theorem for the latency bound will be proven. The latency is the same as the DRR and the derivation is very similar to the one in [94]. It is provided here for completeness and it is simplified. Moreover the same arguments will be used later for deriving the latency of the SRR algorithm.

Theorem 4.4.2. *The LBFS-DRR scheduler belongs to the class of the LR-servers, with an upper bound on the latency θ_i, for flow i given by*

$$\theta_i^{LBFS-DRR} \leq \frac{F - Q_i + (N-2)(L_{max} - 1)}{r} + \frac{L_{max} - 1}{r_i}, \tag{4.11}$$

where N is the number of contending flows with r_i being the guaranteed rate for flow i and the sum of the guaranteed rates of all the N flows being equal to the link rate r. Q_i is the quantum for flow i, which is at least the maximum packet size on the network L_{max} and F is the frame size.

Proof. Before describing the actual proof let us first make some observations about the way service per flow in a round is tracked by the scheduler. When a flow becomes backlogged in a round, say 0, where it has not received any service yet, its deficit counter is initialized with its quantum $DC_i(0) = Q_i$ (see line 11 from the pseudo code of the enqueue process in Table 4.2). When a flow is still backlogged after it was found non-eligible for further service in the round, its deficit counter is increased with its quantum (see lines 27-29 from the pseudo code of dequeue process in Table 4.3). Thus before the first service in the next round the deficit counter of flow i is $DC_i(1) = Q_i + DC_i(0)$.

Mathematically the service W_i received by a backlogged flow i in one round, say k_b, starting at time t_{k_b-1} and finishing at time t_{k_b} is given by

$$W_i(t_{k_b-1}, t_{k_b}) = Q_i + DC_i(k_b - 1) - DC_i(k_b), \tag{4.12}$$

where $DC(k_b - 1)$ is the deficit after the service in round $k_b - 1$. If the flow's queue was empty at the start of the $k_b - th$ round, then $DC(k_b - 1) = 0$. If the flow does not remain constantly backlogged during a round but has backlogged and empty periods, it will be visited multiple times by the scheduler during the round. However the state of the deficit counter is not lost. If the flow is no longer backlogged after transmitting a packet at, say time t_p, the value of its $DC_i(t_p)$ is kept. If it becomes backlogged again at some time instance t'_p in the same round, the service continues to be tracked from the same $DC_i(t'_p) = DC_i(t_p)$. Thus regardless of how many times in a round the flow becomes empty and backlogged again, the service received can never exceed the service it would have received if it remained backlogged during the round.

Further, the theorem is proven by showing that it is true for a backlogged period of a flow and then showing that it is a tight bound. Consider a system with N flows served by a LBFS-DRR scheduler and a flow i which becomes backlogged during round k_0, which starts at t_0.

For any time $t : t_{k-1} \leq t \leq t_k$, where t_k indicates the start of round k

$$W_i(t_0, t_{k-1}) \leq W_i(t_0, t) \leq W_i(t_0, t_k). \tag{4.13}$$

Using Equation (4.12) over $k - 1$ consecutive backlogged rounds after k_0 the work received by the flow is given by

$$W_i(t_0, t_{k-1}) = (k-1)Q_i + DC_i(k_0 - 1) - DC_i(k_0 + k - 1) \tag{4.14}$$

Because round k_0 is the start of a backlogged period $DC_i(k_0 - 1) = 0$.

Considering the bounds on the deficit from Lemma 4.4.1 we obtain via the lhs. of Equation (4.13) a lower bound on the work in the backlogged period (t_0, t)

$$W_i(t_0, t) \geq (k-1)Q_i - (L_{max} - 1) \tag{4.15}$$

Extending Equation (4.14) for k rounds and considering the bounds on the DC in a backlogged period from Lemma 4.4.1 one can upper bound the service in the period (t_0, t) via the rhs. of Equation (4.13)

$$W_j(t_0, t) \leq kQ_j + L_{max} - 1 \tag{4.16}$$

for $j \neq i$. $DC_j(k_0 - 1)$ might be different from 0 because we consider the service received from the flows in the period which might have been backlogged at the start of the k_0

round.

The total work done by the server for the time interval (t_0, t) can be written as the sum of the amount of service received from all flows and can be bound considering Equation (4.16)

$$\begin{aligned} W_s(t_0,t) &= \sum_{j=1;j\neq i}^{N} W_j(t_0,t) + W_i(t_0,t) \\ &\leq \sum_{j=1;j\neq i}^{N} (kQ_j + L_{max} - 1) + W_i(t_0,t) \\ &\leq k(F - Q_i) + (N-1)(L_{max} - 1) + W_i(t_0,t), \end{aligned} \tag{4.17}$$

where $W_j(t_0, t)$ was bounded by Equation (4.16). Expressing k and considering that for a work conserving, constant rate server $W_s(t_0, t) = r(t - t_0)$

$$k \geq \frac{r(t-t_0)}{F - Q_i} - \frac{(N-1)(L_{max}-1) + W_i(t_0,t)}{F - Q_i} \tag{4.18}$$

We now replace the value of k in (4.15)

$$W_i(t_0,t) \geq \frac{r(t-t_0) - (N-1)(L_{max}-1) - W_i(t_0,t)}{F - Q_i} Q_i - Q_i - (L_{max} - 1). \tag{4.19}$$

Moving W_i to the lhs. gives

$$W_i(t_0,t) + \frac{Q_i}{F - Q_i} W_i(t_0,t) \geq \frac{(r(t-t_0) - (N-1)(L_{max}-1)}{F - Q_i} Q_i - Q_i - (L_{max} - 1) \tag{4.20}$$

and simplifying it further results in

$$W_i(t_0,t) \frac{F}{F - Q_i} \geq \frac{(r(t-t_0) - (N-1)(L_{max}-1)}{F - Q_i} Q_i - Q_i - (L_{max} - 1). \tag{4.21}$$

Multiplying both sides by $(F - Q_i)/F$ gives for the work of the i-th flow

$$W_i(t_0,t) \geq \frac{r(t-t_0) - (N-1)(L_{max}-1)}{F} Q_i - Q_i + \frac{Q_i^2}{F} - (L_{max} - 1) + \frac{Q_i(L_{max}-1)}{F}. \tag{4.22}$$

Simplifying further the rhs. gives

$$W_i(t_0,t) \geq \frac{Q_i}{F}(r(t-t_0) - (N-2)(L_{max}-1) - F + Q_i) - (L_{max} - 1). \tag{4.23}$$

Taking into account the relation (4.7) we obtain for the service received by a backlogged session i in the time interval (t_0, t)

$$W_i(t_0, t) \geq max\left(0, r_i(t - t_0 - \frac{F - Q_i + (N-2)(L_{max} - 1)}{r} - \frac{(L_{max} - 1)}{r_i})\right) \quad (4.24)$$

The inequality is valid for the interval (t_0, t) where the flow i will experience its worst latency and thus from Equation (4.24) the theorem is proved. □

Next will be shown that the latency bound given by Theorem 4.4.2 is tight by illustrating a case where it is actually achieved. It describes a similar scenario as the one described in [94]. Assume that flow i becomes busy at some time instant t_0 which coincides with the start of a round k_0. Assume that at this time instant there are $(N-1)$ backlogged flows in the scheduler. The flow will be added to the backlogged flow list in front of them. Thus it will receive service immediately. Assume further that the N flows are active for any time t, $t \geq t_0$ and the sum of their reserved rates equals the link rate r, hence $\frac{Q_i}{F} = \frac{r_i}{r}$. Since flow i became backlogged at the start of the round its deficit $DC(k_0 - 1)$ is 0. Let the deficit counters of all the other flows be equal to $L_{max} - 1$. Assume that during its service opportunity flow i transmits $Q_i - (L_{max} - 1)$ bytes. Note that from Equation (4.15) this equals the minimum service a flow can receive in a round. From Equation (4.16) , a flow j can transmit at most $Q_j + L_{max} - 1$ when its deficit counter is $L_{max} - 1$. In the worst case all the other $N - 1$ backlogged flows will receive their maximum service $\sum_{j=1, j\neq i}^{N} Q_j + L_{max} - 1 = F - Q_i + (N-1)(L_{max} - 1)$. Thus, the size of the first round $t_1 - t_0$, in which flow i is backlogged is

$$t_1 - t_0 = \frac{Q_i - (L_{max} - 1) + F - Q_i + (N-1)(L_{max} - 1)}{r} = \frac{F + (N-2)(L_{max} - 1)}{r}. \quad (4.25)$$

A plot of the service received by flow i versus time is illustrated in Figure 4.2. Thus from the moment flow i becomes backlogged it starts to receive service at rate r for a period of length $\frac{Q_i - Lmax}{r}$. Thus for the period $\frac{Q_i - Lmax}{r_i}$ the rate at which flow i is served is $\geq r_i$. Suppose that after the $N - 1$ flows received their service in round k_0 their packet backlogged was exhausted. However just before the start of round $k_0 + 1$ part of them become backlogged again. Thus they are inserted in the *BackloggedFlows* list before flow i. The scheduler starts to service these newly backlogged flows. Before flow i starts to be served in round $k_0 + 1$ at time t_1^i all $N - 1$ flows have become backlogged each flow j transmitting Q_j bytes. At time t_1^i flow i starts to receive service greater than or equal

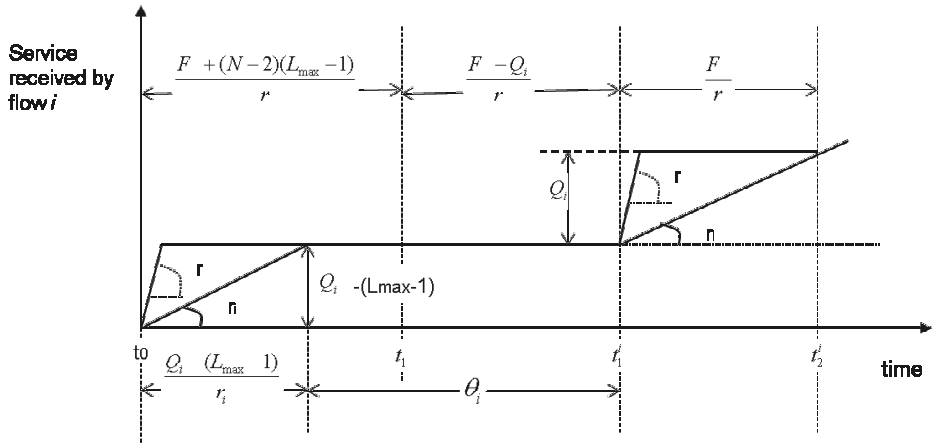


Figure 4.2: Illustration of the latency bound of LBFS-DRR

to its reserved rate r_i. The latency of the flow is the interval in which it has received service less than its reserved rate, which is indicated with θ_i on the Figure 4.1. Referring to the figure we can write

$$\begin{aligned} \theta_i &= \frac{F + (N-2)(L_{max}-1)}{r} + \frac{F - Q_i}{r} - \frac{Q_i - (L_{max}-1)}{r_i} \\ &= \frac{F - Q_i + (N-2)(L_{max}-1)}{r} + \frac{L_{max}-1}{r_i}. \end{aligned} \tag{4.26}$$

Thus in the described scenario the latency bound is exactly met.

4.4.3 Fairness of LBSF-DRR

If two connections are continuously backlogged over a certain interval then their round robin order, relative to each other under the LBSF-DRR scheduling disciplines, remains the same over the interval. To obtain the fairness Equation (1.12) from Section 1.3.3 is used. Consider time interval (τ, t) where two flow i and j are continuously backlogged.

Let τ be some time instant in round k_0, before flow i is considered for service. Let flow i receive service in round k_0 before flow j and in round k the moment t is taken after flow i receives service and before flow j is start being served. Replacing the minimum bound from Equation (4.15) and the maximum bound from Equation (4.16) on the service received from two backlogged flows in the equation giving the fairness one obtains

$$\left| \frac{W_i(\tau,t)}{r_i} - \frac{W_j(\tau,t)}{r_j} \right| \leq \frac{Q_j}{r_j} + \frac{L_{max}-1}{r_i} + \frac{L_{max}-1}{r_j}. \tag{4.27}$$

Taking into account Equation (4.7) the fairness of the LBFS-DRR scheduler is

$$\Phi^S = \frac{F}{r} + \frac{L_{max} - 1}{r_i} + \frac{L_{max} - 1}{r_j}. \tag{4.28}$$

4.4.4 Latency bound of SRR

In order to derive a latency bound for SRR, first the bounds on the surplus counter have to be obtained. It is trivial to prove that they are given by the following Lemma

Lemma 4.4.3. *The surplus counter of a backlogged flow i at the end of a round is bounded by*

$$-(L_{max} - 1) \leq SC_i \leq 0 \tag{4.29}$$

where L_{max} indicates the maximum packet length.

Now it can be shown that the SRR algorithm, with the implementation from Section 4.3, is a $Latency - Rate$ server by proving the following theorem:

Theorem 4.4.4. *The SRR scheduler belongs to the class of the LR-servers, with an upper bound on the latency θ_i, for flow i given by*

$$\theta_i^{SRR} \leq \frac{F - Q_i + (N-1)(L_{max} - 1)}{r} \tag{4.30}$$

Proof. The proof follows closely the ideas behind the proof for Theorem 4.4.2. As was discussed earlier the latency can be bound based on the offered service in any backlogged period. Consider a backlogged period starting in a round k_0 in which the flow does not have to be penalized, i.e., $SC_i(k_0 - 1) = 0$. An example of such a period is the first backlogged period of the flow. Let the start time of round k_0 be t_0. The service received by a backlogged flow i in one round, say k, starting at time t_{k-1} is given by

$$W_i(t_{k-1}, t_k) = Q_i + SC_i(k-1) - SC_i(k) \tag{4.31}$$

$SC(k-1)$ is the penalty for the surplus the flow received the previous round. Using equation (4.31) over k consecutive backlogged rounds after k_0 the work is given by

$$W_i(t_0, t_k) = kQ_i + SC_i(k_0 - 1) - SC_i(k_0 + k) \tag{4.32}$$

Both expressions are the same as the ones for LBFS-DRR but as the bounds on the SC are different from the ones on the deficit counter the work done for a backlogged

period differs for the two schedulers. More specifically, the upper bound on the service in a time interval (t_0, t), where t is a time instance in the k-th round, is the same as for LBFS-DRR given by Equation (4.16). It is obtained by replacing in Equation (4.32) the lower bound on the surplus counter $SC(k_0, k)$ from Lemma 4.4.3. The lower bound is used because it is reached when a flow receives surplus service.

The lower bound on the service for flow i is given by

$$W_i(t_0, t) \geq (k-1)Q_i, \tag{4.33}$$

where the upper bound on the surplus from Lemma 4.4.3 was used. It is reached when the flow does not receive any surplus in a backlogged period. It is greater the bound for the LBFS-DRR scheduler given by Equation (4.15) thus in a backlogged period a flow would receive more service under SRR than under LBFS-DRR. This will result in a lower latency as will be demonstrated further on.

The upper bound on the total work done by an SRR scheduler in the time interval (t_0, t) is given by the same expression as for LBFS-DRR and consequently the number of rounds k in this period is bounded by Equation (4.18). This bound on k can now be replaced in Equation (4.33)

$$W_i(t_0, t) \geq \frac{r(t-t_0) - (N-1)(L_{max}-1) - W_i(t_0, t)}{F - Q_i} Q_i - Q_i. \tag{4.34}$$

Expressing W_i and further simplifying the rhs. in a similar way as for Theorem 4.4.2

$$W_i(t_0, t) \geq \frac{Q_i}{F}(r(t-t_0) - (N-1)(L_{max}-1) - F + Q_i). \tag{4.35}$$

Taking into account the relation (4.7) the service received by a backlogged flow i in the time interval (t_0, t) is bounded by

$$W_i(t_0, t) \geq max\left(0, r_i(t - t_0 - \frac{F - Q_i + (N-1)(L_{max}-1)}{r})\right) \tag{4.36}$$

In order to show that this is a tight bound an example is given. In the case of SRR algorithm it is a rather obvious one because the upper latency bound coincides with the maximum start up delay. Consider a flow which starts service in round r when there are N-1 other backlogged flows in the system. Before being considered for service the scheduler will service the other N-1 flows, i.e., it has to wait for N-1 flows to receive

service given by Equation (4.31)

$$t - t_0 \leq \frac{\sum_{j=1}^{N-1} Q_j - SC(k-1) + SC(k)}{r} \leq \frac{F - Q_i + (N-1)(L_{max} - 1)}{r}, \quad (4.37)$$

where the last inequality used that $F = \sum_{i=1}^{N} Q_i$ and Lemma 4.4.3. This concludes the proof. □

4.4.5 Fairness of SRR

As for LBFS-DRR, to calculate the fairness of the SRR algorithm the Golestiani fairness is used from Section 1.3.3. For this the upper and the lower bound on the service received by a backlogged flow are needed. The lower bound in a period τ, t is obtain in the same way as Equation (4.31) but replacing SC with the lower bound from Lemma 4.4.3, giving

$$W_i(\tau, t) \geq (k-1)Q_i - (L_{max} - 1) \quad (4.38)$$

The upper bound is the same as for the LBFS-DRR algorithm. Replacing the minimum bound from Equation (4.38) and the maximum bound from Equation (4.16)) on the service received from two backlogged flows in Equation (1.12) one obtains

$$\left| \frac{W_i(\tau, t)}{r_i} - \frac{W_j(\tau, t)}{r_j} \right| \leq \frac{Q_j}{r_j} + \frac{L_{max} - 1}{r_i} + \frac{L_{max} - 1}{r_j}. \quad (4.39)$$

The moment τ is at some time instant in round k_0 before flow i is considered for service. Flow i receives service in round k_0 before flow j and in round k the moment t is taken after flow i receives service and before flow j starts being served. Recall that if two connections are continuously backlogged over a certain interval then their round robin order relative to each other under the SRR scheduling disciplines remains the same over the interval.

Taking into account Equation (4.7) the fairness of the SRR algorithm can be written as

$$\Phi^S = \frac{F}{r} + \frac{L_{max} - 1}{r_i} + \frac{L_{max} - 1}{r_j}. \quad (4.40)$$

4.5 Discussion

Several latency bounds have been reported in the literature for DRR [91], [94], [73]. The one reported in [94] is the lowest bound and is

Table 4.6: Comparison of Scheduling Algorithms with O(1) complexity

Scheduler	Fairness	Latency
LBFS-DRR	$\frac{F}{r} + \frac{L_{max}-1}{r_i} + \frac{L_{max}-1}{r_j}$	$\frac{F-Q_i+(N-2)(L_{max}-1)}{r} + \frac{L_{max}-1}{r_i}$
SRR	$\frac{F}{r} + \frac{L_{max}-1}{r_i} + \frac{L_{max}-1}{r_j}$	$\frac{F-Q_i+(N-1)(L_{max}-1)}{r}$
DRR	$\frac{F}{r} + \frac{L_{max}-1}{r_i} + \frac{L_{max}-1}{r_j}$	$\frac{F-Q_i+(N-2)(L_{max}-1)}{r} + \frac{L_{max}-1}{r_i}$

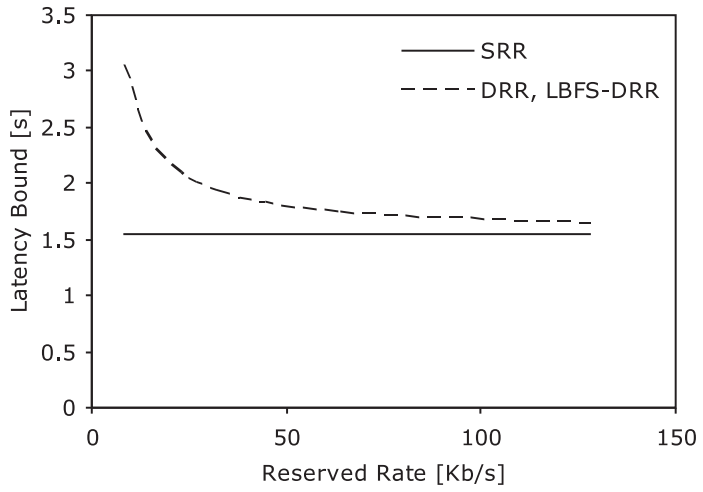


Figure 4.3: Comparison of the latency bounds of SRR, DRR and LBFS-DRR for flows with different reserved rates

$$\theta_i^{DRR} = \frac{(W - w_i)Q_{min} + (N-2)(m-1)}{r} + \frac{m-1}{r_i}. \tag{4.41}$$

In this equation $(W - w_i)Q_{min} = F - Q_i$ and the authors in [94] use m to denote the maximum packet size in a round, which in the considered model is bounded by the maximum packet size for the network - L_{max}. This result is the same is the one obtained for LBFS-DRR in Theorem 4.4.2.

In Table 4.6 the fairness and latencies of the three O(1) complexity algorithms DRR, SRR and the LBFS-DRR are summarized. The fairness value for DRR is the one reported in [73]. The LBFS-DRR latency bound obtain in Section 4.4.2 and the fairness are the same as the one for the DRR.

On Figure 4.3 the latencies depending on the flows' reserved rate are compared by considering a practical example from broadband access cable network. Consider an output link with rate $r = 40Mb/s$ shared by 100 flows. The minimum reserved rate r_{min} is 8 Kb/s and the maximum packet size is 1518 bytes. Let the minimum quantum equal the maximum packet size then the length of frame is expressed from Equation (4.7) as $F = \frac{Q_{min}}{r_{min}} = 1.518s$.

The SRR latency bound is lower than the bound of the other two algorithms algorithm with a term $\frac{L_{max}-1}{r_i} + \frac{L_{max}-1}{r}$. The difference is inversely proportional to the reserved rate thus it is more significant for flows with small reserved rates. It comes of course from the fact that the DRR based algorithm would delay a packet and accumulate a deficit, while SRR would schedule with surplus, but would send a packet even if only part of it fits in the fair share for the round.

4.6 Simulation results

The performance of the LBFS-DRR and the SRR algorithms are evaluated by means of the simulation program implemented in OMNET++ [82] and discussed in Section 3.1. The algorithms are implemented at the CMTS module (recall from Figure 3.1). The Internet side servers act as traffic (packet) generators. There is no delay between the servers and the CMTS node. Each server generates traffic destined to only one service flow. Packets coming from the Internet side servers, are classified into (service) flows and queued at the scheduler. The scheduler keeps one queue per flow.

The network modeled accounts only for the Ethernet layer. Recall that in the downstream, the DOCSIS overhead from the MPEG cells is constant. It is not taken into account in these simulations. The maximum packet size is the maximum size of the Ethernet frame - 1518 bytes.

Three sets of simulations' scenarios are reported in this section. In the first the end-to-end average delay depending on the file size is shown. In the second, the isolation of the flows is demonstrated and in the third, the queuing delay dependance on the reserved rate for the weighted version is studied. In all the scenarios the performance of LBFS-DRR and SRR is compared with the DRR and WF^2Q+ scheduling algorithms.

Application Level Delay

In the first simulation scenario the network link rate, from the CMTS to the modems, is R=10 Mbits/s and there are 40 active flows each transmitting at rate 0.95*250Kb/s,

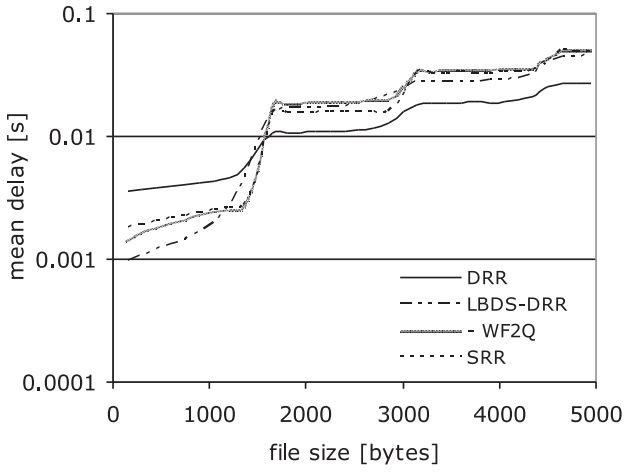


Figure 4.4: Mean transfer delay vs. file size

which results in 0.95*Link rate. The files coming from the application layer of the servers on the Internet side are fragmented in Ethernet frames, thus adding additional 46 bytes overhead, and then transmitted on the network to arrive at the CMTS. They are queued there, at the scheduler, at the corresponding queue for the corresponding service flow. The quantum sizes for all flows for the DRR, SRR and LBFS-DRR are 1518 bytes. Thus the maximum file size, which can fit in one quantum is 1472 bytes.

Typical user traffic patterns for file sizes, as shown in [110] are close to heavy tailed distributions. Specifically, they can be modelled as pareto bounded distribution with parameters (24, 15800, 1.1). Such a distribution is commonly used to model file size distribution on the Internet [111]. In Figure 4.4 the mean file transfer delay vs. the file size is shown. The delay function is stepwise because the Ethernet layer fragments the file received from the application layer in packets with maximum length of 1472.

For files with small size, below the quantum, the LBFS-DRR achieves the lowest delay. It is expected as they fit in only one packet and are more likely to pre-empt the service of the other backlogged flows. For longer files, which still fit in one packet, the experienced delay increases with the file sizes because the probability that a packet will find other packets in the flow's queue such that all the packets can not be transmitted in one round, increases with the size of the arriving packet. Thus, these longer files are more likely to be delayed with a round even if their size is less than the quantum.

The SRR has lower average delay for files fitting in one packet than DRR for two reasons. First, due to the advantage of the surplus policy, that it transmits one packet

Table 4.7: File Delay

	Mean[ms]	Max[ms]	StdDev[ms]
DRR	3.78	256.28	4.45
LBFS-DRR	1.31	308.22	4.48
WF2Q+	1.78	343.33	5.11
SRR	2.13	355.31	5.12

more than DRR in a given period. Second, because of the implementation with the rounds counter. When a flow becomes backlogged in a middle of a round, part of the already backlogged flows would have received their quantum of service for the round and would be in the list for the next round. With the SRR this newly backlogged flow will still be able to transmit packets in the current round, while with DRR it will have to wait until all other backlogged flows receive their quantum, thus effectively will wait for a round.

The small slope in the mean delay for file sizes below 1472 bytes in the results for DRR and SRR and partially for WF^2Q+ and LBFS-DRR is due to the longer packet size.

For all other file sizes bigger than the quantum it is the DRR scheduler which results in the lowest delay. Ones queued they receive service in a round before other flows which become backlogged meanwhile.

In Table 4.7 the results for the average and the maximum packet delay and standard packet delay deviation per flow are given. As it can be seen for heavy tailed distributed file sizes the lowest mean delay is achieved with LBFS-DRR. Figure 4.5 shows the packet size distribution used in the simulation. For such distribution the majority of the files are small. Namely, in the simulated scenario there are 99% files with size less than 1500 bytes. When the user experienced delay is averaged out over all files, the small ones deliver the highest contribution.

Of course, as can be deduced from Figure 4.4 when all files are bigger than 1472 bytes and thus no file can be transmitted in only one round, LBFS-DRR has the worst performance, while DRR achieves the lowest delay. The results from such a scenario, where all files are bigger than 1500 bytes is given in Table 4.8.

Packet Queuing Delay

In the second simulation scenario the packet level performance of the scheduler, i.e., the packet queuing delay is studied. The aim is to demonstrate that flows, which transmit

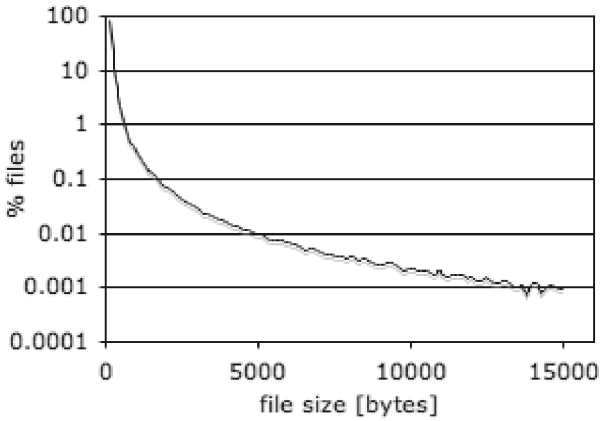


Figure 4.5: File Size Distribution

Table 4.8: File delay for file size distribution with minimum of 1500 bytes

	Mean[ms]	Max[ms]	StdDev[ms]
DRR	27.8	240.5	23.1
LBFS-DRR	30.8	311.6	31.8
WF2Q+	29.2	298.3	29.4
SRR	29.0	303.6	28.9

below their respective share, experience lower packet delay than flows transmitting more than that. A more realistic setup is used, with link rate R=40Mb/s and N=160 flows contending for the bandwidth. The flows are divided in 8 groups depending on the rate they generate traffic. The lowest is 16 Kb/s and the highest 4Mb/s. The packet sizes are drawn from an approximate distribution of the one reported in [112] with four distinct peaks at 64, 552, 576 and 1518 bytes with probability 0.45, 0.05, 0.06 and 0.1 correspondingly. The probability that a packet has a size in the range [65,551] bytes is uniformly distributed and gives a total of 0.3. The same distribution is for sizes in the range [577,1517] for a total probability of 0.04. Thus the average packet size E[p] is 426.16 bytes. The packet generation process at the servers is ON-OFF with pareto distributed ON times with average time in ON period E[ON]= 4ms and exponentially distributed OFF times with average length E[OFF]= 6ms. The packet inter-arrival times (IAT) are exponentially distributed and are calculated from

$$IAT = \frac{E[p] * E[ON]}{\rho_i * (E[ON] + E[OFF])}, \tag{4.42}$$

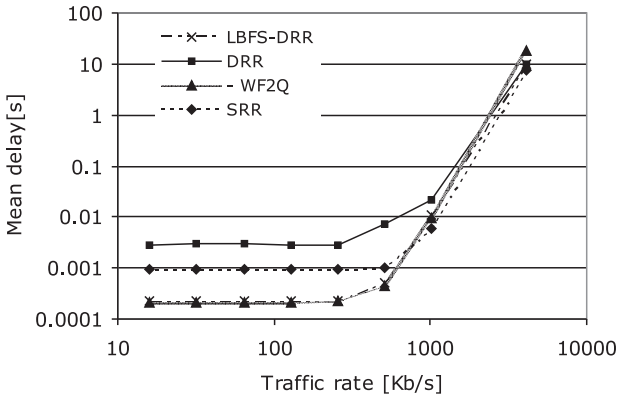


Figure 4.6: Average packet queuing delay with all $w_i = 1$ traffic load $\approx$ Link Rate

where ρ_i is the desired traffic generation rate for flow i.

The results for the average and maximum packet queuing delay for the simulated scenario are shown in Figures 4.6 and 4.7 correspondingly. Each point is obtained from the delays of all the packets from all the flows in a group. The simulations run for a sufficient amount of time to assure acceptable convergence of the delays though the confidence intervals are not shown on the figures.

The fair share of a flow if all flows are backlogged is $R/N = 250Kb/s$. As it can be seen, flows which generate at rate below or at their fair share, experience lower delay for all simulated algorithms. For LBFS-DRR packets, arriving with lower rate than the flow's fair share, are more likely to find their corresponding queue empty, on arrival at the scheduler. Thus they are likely to be inserted at the head of the current active list. Their delay will not be influenced by the backlogged flows but only by new arrivals from the other flows.

From the figures it can be seen that the average behaviour of the LBFS-DRR algorithm is closer to the one of the WF^2Q+. The delay for flows transmitting below their fair share under the DRR scheduling discipline is found to be ≈ 10 times worse than the one experienced with the other two disciplines. Indeed, flows scheduled under DRR, which transmit at a rate lower then their fair share, have to wait until all other flows received service, while under LBFS-DRR and WF^2Q+ they are competing only between each other.

SRR scheduling results in a delay approximately halfway between the delay under LBFS-DRR and DRR for flows transmitting at lower rate than their fair share. In comparison with LBFS-DRR, packets from a newly backlogged flow, like the flows trans-

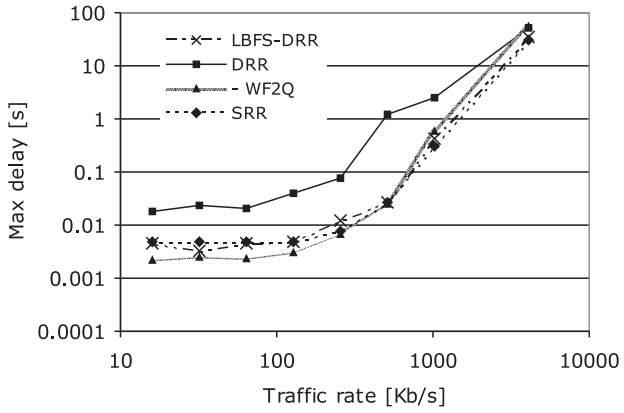


Figure 4.7: The maximum packet queuing delay with all $w_i = 1$ traffic load $\approx$ Link Rate

mitting at a rate lower than the fair share are most of the time, will have to wait for the packets of other backlogged flows in the round. Still they do not have to wait until all backlogged flows receive their quantum of service, as under DRR scheduling.

The maximum delays achieved in this simulation scenario for flows with rate lower than the fair share for LBFS-DRR and SRR are close to each other and are below one maximum frame size ($\approx 49ms$). Thus packets from these flows are never delayed for a round and have to wait before being transmitted only for the packets from other low rate flows. The maximum delays for the packets under the DRR discipline is close to one frame size as they always have to wait for all other backlogged flows to receive service.

Packet Queueing Delay for Weighted Flows

With the third simulation scenario the aim is to demonstrate the performance of the algorithms when there are flows with different weights. The 160 flows are divided in groups with different weights ranging from 1 to 256. The sum of all weights is 2500 and weight $w_i = 1$ corresponds to $r_i = 16$ Kb/s reserved rate. Each point on the figures is obtained by averaging or taking the maximum value of the packet delays from all the packets from all flows from the same group. The total guaranteed rate is the same as the link capacity, i.e., $\sum_i r_i = R$. Part of the flows - "well-behaved" - are transmitting traffic at their guaranteed rate with leaky bucket parameters (r_i, 1518). Another part "misbehaving" comprising of 32 flows that are transmitting above their respective reserved rates and are expected to be constantly backlogged. Figure 4.8 givs the average delay of the "well-behaved" flows versus their reserved rate.

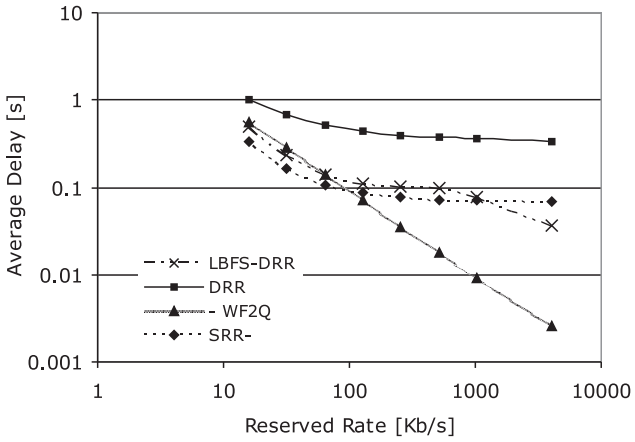


Figure 4.8: Average packet queueing delay vs. the reserved rate of its corresponding flow for traffic load $\approx$ Link Rate

The 32 flows transmitting more than their reserved rate remain backlogged throughout the simulation. The packets from well behaved flows under LBFS-DRR will precede the packets from the 32 constantly backlogged ones and thus the latter do not influence the mean delay of the former. As a result the mean delay experienced by the packets under the LBFS-DRR scheduling is less than under DRR.

The WF^2Q+ scheduler serves the flows at their relative reserved rate thus it can delay packets from flows with lower reserved rate in the presence of backlogged flows with high reserved one. The LBFS-DRR scheduler as a round robin algorithm tends to equalize the delay of the flows in a round regardless of their reserved rate. When a packet from a non-backlogged flow, say i, arrives at the LBFS-DRR scheduler at time t, it is inserted at the front of the current active list. Only packets from non-backlogged flows which arrive after this time instant t influence the packet delay of flow i. When a packet from a flow with low reserved rate arrives at a LBFS-DRR scheduler it will precede packets from flows with higher reserved rates if they have been backlogged for more than a round. While waiting for service, new flows transmitting at higher rate will become backlogged but still in comparison with WF^2Q+ this delay will be lower. For the flows with higher rates the opposite scenario happens. As packets from such flows arrive more often they will precede packets from flows with lower rates. However while waiting for service, packets from flows with lower rate might arrive, which will result in a higher delay than the one experienced under the WF^2Q+ discipline.

The average delay under SRR discipline is slightly lower than the one under LBFS

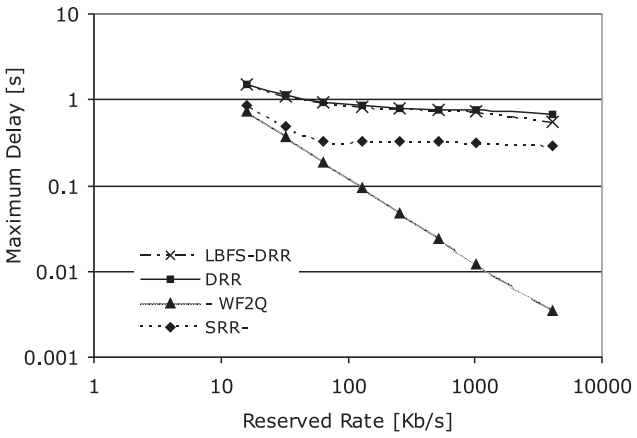


Figure 4.9: The maximum packet queueing delay vs. the reserved rate of its corresponding flow for traffic load ≈ Link Rate

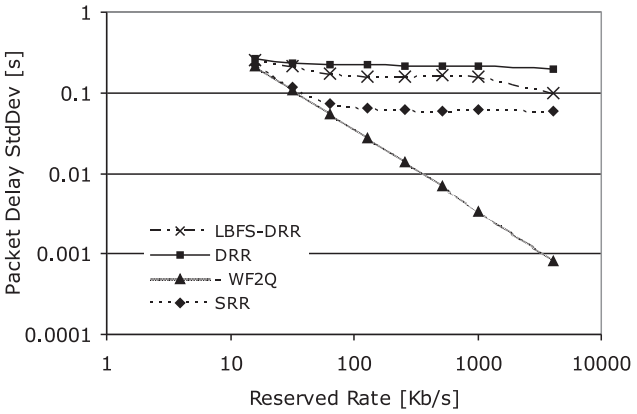


Figure 4.10: The standard deviation of the packet delay vs. the reserved rate of its corresponding flowfor traffic load ≈ Link Rate.

for flows with all reserved rates except for the highest one. The packets from these flows have the highest arrival rate. Under LBFS-DRR they are going to precede the packets from the other flows while under SRR they are going to receive service after some of the flows with lower rate.

The maximum delay measured in this scenario is the same for LBFS-DRR and DRR as presented in Figure 4.9. Under SRR lower maximum delay is achieved as is to be expected from the analysis presented in Section 4.4. To complete the analysis of this

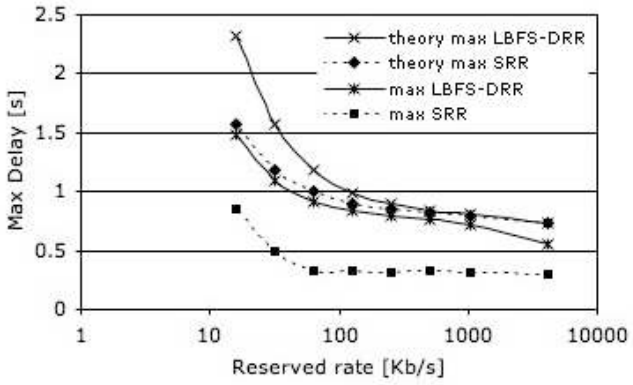


Figure 4.11: Maximum delay bound for leaky bucket shaped traffic

scenario Figure 4.10 shows the standard deviation of the packet delay, which for LBFS-DRR it is in the order of DRR.

Finally, the theoretical and maximum delay bounds obtained respectively via analytical calculations and via simulations are shown on Figure 4.11. The theoretical maximum delay is obtain from the latency of the algorithm and the maximum burst size σ_i, which is 12144 bits by Equation (1.11) from Section 1.3.2. For none of the algorithms the theoretical maximum is reached. In Sections 4.4.2 and 4.4.4 examples were given when this maximum can be achieved. This requires that many conditions are fulfilled which is obviously a rare event.

4.7 Conclusions

In this chapter a novel discipline was presented, namely the Last-Backlogged First-Served (LBFS) scheduling discipline. It was combined with the popular Deficit Round Robin (DRR) algorithm to create an O(1) complexity scheduler. The new LBFS-DRR algorithm serves short flows faster than DRR. As a result the average user experience for the delay over several flows with different typical lengths would be perceived as lower. In the presence of flows transmitting more than their momentous fair share, the well-behaved flows are not affected. They remain isolated and even stronger their packet delay under the LBFS-DRR is close to the delay of a sorted priority scheduling algorithm, like the Worst-Case Fair Weighted Fair Queuing + (WF^2Q+). The fairness and latency bound of the algorithm were found to be exactly the same as the one derived for DRR. Flows with high reserved rate receive differentiated treatment and lower average packet

delay, as is desirable as they would typically pay more for the service.

The implementation for the Surplus Round Robin (SRR) algorithm proposed in this chapter has also O(1) complexity and succeeds in guaranteeing fairness and delay bounds for leaky-bucket shaped traffic. The fairness and the latency bounds were calculated analytically. It was shown that the Golestiani fairness for SRR is the same as the one for DRR. The latency bound for SRR was found to be lower than the one for DRR. This would ultimately result in a lower maximum packet delay, which was demonstrated via simulations. The SRR algorithm also achieves lower average delay than the one of DRR but not as low as LBFS-DRR.

The LBFS discipline can be combined also with SRR and preliminary simulation results have indicated that the average delay of such a scheduler is lower than the one achieved with LBFS-DRR or SRR.

Chapter 5

Packet Scheduling Algorithms for Networks with Multiple Bonded Channels

In this chapter per-flow queuing algorithms designed for packet scheduling on multiple channels are proposed. The algorithms have a distributed architecture, which makes them work conserving even when flows can be distributed only on a selection of the channels. Each algorithm consists of several Deficit Round Robin channel schedulers. The algorithms differ on the type of queuing they use. The output queuing algorithm uses per-channel, per flow queuing and a load distributor, which selects the channel a packet will be transmitted, at the time of the packet arrival. A single channel DRR algorithm schedules between the queues. The input queuing algorithm, named Bonded Deficit Round Robin (BDRR) keeps only one queue per flow and has also different schedulers per channel. However these schedulers are not fully independent but share common variables. It is shown that the BDRR scheduler is a latency rate scheduler and that it results in limited, bounded packet reordering. The performance of both algorithms is further analyzed via simulations for a variety of traffic and load scenarios.

5.1 Background and Overview

Link aggregation is a common technique to increase the available capacity. For point to point networks, it is standardized in [113] and for Ethernet systems in [7]. It is analogous to multi-server systems, which arise also in multiprocessor architectures, multi-path storage I/O and cluster-based Web servers, to name a few. Load balancing algorithms,

also referred to as load sharing or stripping algorithms, are used to share in a fair manner the load amongst the servers or channels. There is a plethora of load balancing algorithms proposed, ranging from simple static policies, such as random distribution or round-robin policy [68], to the ones using hashing [114], incorporating prediction schemes [115] or resulting in ordered output [116]. These load balancing algorithms however, do not perform flow scheduling. For example, for the striping algorithm proposed in [68], the Surplus Round Robin (SRR) scheduler discussed in the previous chapter is used. For each channel it keeps a Surplus Counter, which tracks the amount of bytes dispatched on the channel in a round. Note that it does not schedule amongst flows but amongst channels.

There are several multi-channel algorithms, which not only act as load distributors but schedule amongst the flows competing for the bandwidth [117], [118], [119], [120], [121] and [122]. All these algorithms presume a point-to-point topology and that all flows can be scheduled on any channel. Hence, they are not readily applied to a Hybrid Fibre Coax (HFC) access network.

An HFC access network, as was discussed in Chapter 3, has a point-to-multipoint topology. Recall from Section 1.1.3 that in DOCSIS 3.0 based HFC networks there are multiple channels between the Cable Modem Termination System (CMTS) and the Cable Modems (CM). The channel bonding mechanism is realized at MAC level and allows the traffic to one CM to be distributed over several channels. The CMs can have different capacities with respect to the number of channels they can receive on. This makes it possible that also legacy CMs can use the system. The CM and the CMTS can receive and transmit on all their channels simultaneously. The set of channels on which a CM can receive is called its Bonding Group (BG) and it is assigned by the CMTS. When the packets from the same flow are transmitted on multiple channels they are tagged with a sequence number. In this way the proper packet sequencing is not lost if they are received out of order. The CM restores the original sequence before forwarding the packets to the user devices. DOCSIS 3.0 inherits the quality-of-service (QoS) support from the older specifications. The QoS support in HFC networks is realized via mapping the packets to service flows (SF). The scheduler at the CMTS arbitrates the allocation of the DS bandwidth amongst the different SFs. Thus a DOCSIS 3.0 scheduler should be able to support per flow queuing.

For DOCSIS 3.0, if any of the multi-channels, multi flow schedulers referred to above is applied, it will result in a non work conserving scheduling mechanism. Consider the following example. When a single, multi-channel scheduler selects a packet from a flow for downstream transmission, it has to have a free channel. However, if none of the

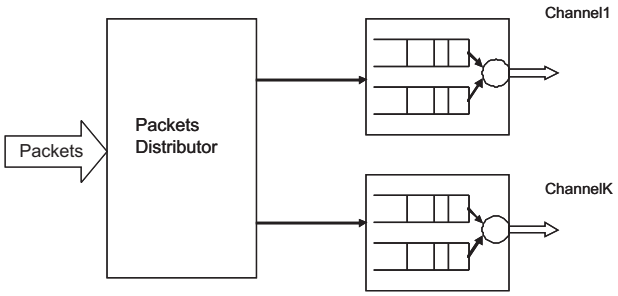


Figure 5.1: Output Queuing

channels on which the flow is assigned is free, the scheduler has to wait until a channel from the bonding group of the selected flow becomes available. Meanwhile, other free channels will remain unused, even if there are packets in the system to flows assigned on them.

This chapter proposes distributed multi-channel per flow scheduling mechanisms, which are applicable to DOCSIS 3.0 networks, perform in constant time and are work conserving. In the following section different types of queuing in the distributed architecture, called simply input and output, are discussed. Next, weight partitioning of the flow's reserved rates is introduced. Section 5.4 introduces a straightforward application of distributed deficit round robin scheduling with output queuing. The distributed DRR with input queuing is proposed in Section 5.5. The next section shows that the latter is a latency rate server and its latency bound is calculated. In Section 5.7 simulation results are presented. The last section concludes and summarizes the topic and results.

5.2 DS packet queuing

Unlike the queuing architecture from Figure 1.4 from Section 1.2 this section considers an architecture with multiple channels, each with a separate scheduler.

Figure 5.1 shows a schematic view of a distributed scheduling architecture, further on referred to as *Output Queuing*. It consists of a packet distributing process and a separate queue per flow for each channel. The distributing process decides upon a packet arrival, to which channel the packet should be transmitted. The channel should belong, of course, to the bonding group of the flow the packet belongs to. Once the channel is determined, the packet becomes the responsibility of the channel scheduler, which can be any per flow queuing algorithm. Thus the packet distributor acts as a stripping algorithm. The rule by which the packets are distributed to the channels will also largely determine the

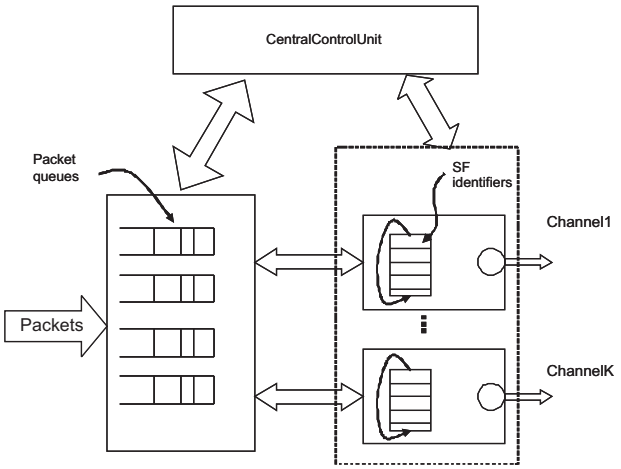


Figure 5.2: Intput Queuing

packet delay.

A disadvantage of output queuing is that it determines the channel, on which a packet is transmitted, at the time of the arrival of the packet in the system. It bases its decision on the information available at this moment or some estimation for the future load on the channels. However the load on the channels can change. Thus, a situation might arise where there are many packets waiting to be transmitted on one channel while other channels, which reach the same flows, have no packets queued for transmission. Another major disadvantage of Output Queuing is that with each packet stored in the queue its sequence number also has to be stored.

The advantage of Output Queuing is that the different channels operate completely independently, they have separate memory and existing scheduling algorithms can be applied without the need for modification.

Figure 5.2 shows a schematic drawing of the distributed architecture for *Input Queuing*. In this scenario the scheduler keeps only one queue per flow where the packets are queued immediately upon arrival. The channel, on which a packet will be transmitted, is determined just before the transmission itself starts. Each channel has a separate per flow-queuing scheduler. However, these channel schedulers are not independent. When a channel becomes free its scheduler selects the next flow based on the available information. Once the flow is selected the common memory pool of the packet queues is accessed. When a scheduler on one channel leaves a queue empty, in order to avoid unnecessary checks by the other channel schedulers, they should be informed. In this sense the channel schedulers are not fully independent. On the figure this dependence is indicated by

sketching a central control unit, which stores and updates the variables, common to all channels and manages the queues. It is also responsible for any concurrency issues.

The advantage of input queuing is that the decision for the channel on which a packet will be transmitted is made at the moment of packet transmission. Thus, it is based on accurate up-to-date information about the state of all channels. Another advantage is that the packet order is preserved until transmission thus there is no need for extra buffer space to store the packet sequence number.

Further on in this chapter two algorithms are proposed, which use as channel schedulers the Deficit Round Robin (DRR) algorithm. Before describing them, preliminary discussion on the quantum selection for each channel is given.

5.3 Rate Partitioning

In a single channel DRR scheduler the flow's quantum is proportional to the flow's weight and the relation is given by Equation (4.2) from Section 4. For channel m the equation the flow's quantums can be written as

$$Q_i^m = w_i^m Q_{min}^m. \tag{5.1}$$

The flow's weights on the channels, w_i^m, are such that their sum should result in the flow's weight in the system, which would correspond to the weight the flow would have if it was served by a single channel scheduler

$$w_i = \sum_{m \in \mathcal{M}_i} w_i^m, \tag{5.2}$$

where $\mathcal{M}_i$ is the set of channels of the flow's bonding group and M_i is the number of channels in the flows bonding group. The weights determine a rate using Equation (4.9).

$$r_i^m = w_i^m r_{min}^m \tag{5.3}$$

The minimum rate corresponds to the minimum weight that can be assigned to a flow on a channel. A natural choice is that the minimum weight is 1 and is the same for all channels. In this way any flow with the minimum weight and a single channel in its bonding group can be assigned on any channel. Thus $r_{min}^m = r_{min}$. From Equation (5.2)

the rates satisfy

$$r_i = \sum_{m \in \mathcal{M}_i} r_i^m \tag{5.4}$$

The minimum quantum in DRR is typically the minimum possible quantum for which the scheduler has O(1) complexity, which is the maximum packet size on the network L_{max}. The minimum quantums on the channels are naturally selected as the minimum possible quantum for which the single channel DRR scheduler has O(1) complexity, which is the same for all channels $Q_{min}^m = Q_{min}$. A frame f^m is one round robin cycle amongst the backlogged flows on channel m. The sum of the quantums determines the frame size and taking into account Equations (5.1) and (5.3) is expressed as

$$f^m = \sum_{i=1}^{N^m} Q_i^m = \sum_{i=1}^{N^m} \frac{r_i^m}{r_{min}} Q_{min}, \tag{5.5}$$

where N^m is the number of backlogged flows assigned to bonding groups which include channel m.

5.4 DRR for channel bonded systems with Output Queuing (OutQ-DRR)

An output queuing algorithm is determined by the type of the scheduler(s) on the channels and by the packet distributing rule. In the considered algorithm the scheduler on each channel is DRR. The quantum for each channel is given by Equation (5.1).

The load balancing rule proposed in this thesis is that the packet distributor forwards the packets to the channel with the least reserved rate at the moment. This is a modification of the rule which forwards packets to the least loaded channels. The reserved rate on a channel is the sum of the channel rates r_i^m of the flows which have backlogged packets on this channel. This sum determines the frame size (see Equation (5.5)) and hence, gives an indication about the delay a packet can experience on the channel.

In Table 5.1 the pseudo code for the packet distributing process is given. For each flow the process keeps the set of channels $\mathcal{M}_i$ from its bonding group. For each channel m it keeps a variable ρ_m indicating the total reserved rate of the flows having packets to be transmitted on this channel. Upon arrival, a packet is classified to a service flow, say i. The classification is based on the destination address or other parameters from the packets headers. The process knows the set of channels in the bonding group of the flow

Table 5.1: PACKETS DISTRIBUTING PROCESS

1. i=p.Flow();
2. $\mathcal{M}_i = i.BondedChannelsSet()$
3. $m = min_{\rho_m}(\mathcal{M}_i)$
4. $IF(i$ NOT IN $BacloggedFlowsList_m)$
5. $\quad \rho_m += r_i^m;$
6.$p.stamp(SequenceNumber)$
7.$SequenceNumber++;$

and from it the channel with lowest ρ_m is selected. If there are several channels with the same ρ_m one is selected randomly. This is an important feature in order to achieve spreading of a burst of packets amongst equally loaded channels from the flows' bonding group.

Provided that the flow is not backlogged on this channel its channel reserved rate r_i^m is added to ρ_m. Before the packet is forwarded to a channel it has to be stamped with the sequence number. The packet distributing process is the one in the system that can keep track of the order of packet arrival. When flow i is no longer backlogged on a channel m, the packet distributing process is informed such that it can reduce the ρ_m variable with r_i^m. As can be seen from the pseudo code all operations are performed in constant time except on line 3 where the channel with the minimum reserved rate is selected, which requires $O(M)$ operations. Here M indicates the number of channels on the system. As the DRR algorithms on the channels perform in constant time, the whole OutQ-DRR algorithm has $O(M)$ complexity.

5.5 DRR for channels bonded systems with Input Queuing

As was discussed in Section 5.2 in order to work properly a distributed algorithm in a input queuing architecture can not have independent schedulers. They are bonded by using a common memory pool for the packet queues and, as will be disclosed further, other common variables. To emphasize this fact the algorithm is named Bonded Deficit Round Robin (BDRR). It is described hereafter.

As an Input Queuing algorithm it keeps one packets queue per flow. For each channel m there is a separate DRR scheduling algorithm with $BackloggedFlows_m$ list and per flow a quantum given by Equation (5.1) and a deficit counter. Consider flow i which requires a minimum rate r_i and is assigned to a bonding group consisting of M_i channels.

When a packet arrives for a flow with an empty queue, a channel from the flow's bonding group is selected randomly and the scheduler for this channel is notified. When a channel is notified the flow's id, i, is added to the $BackloggedFlows$ list of the channel's DRR scheduling algorithm. If a second packet arrives before the first one is served and $M_i > 1$, a second channel from the bonding group is notified. With the further increase of the number of packets in the flow's queue more channels are notified until all the channels from the flow's bonding group are notified. Thus if there are qn_i packets in a flow's queue the number of notified channels n_i is

$$n_i = min(qn_i, M_i). \tag{5.6}$$

This condition guarantees that when a flow is selected for service from a channel scheduler, it will have at least one packet in its queue to be transmitted. The channels' schedulers have to ensure that after a packet is transmitted the condition given by Equation (5.6) is satisfied. For example if the transmission of a packet from the flow's queue will cause the number of packets in the queue qn_i to become less than the number of notified channels minus one $n_i - 1$, the scheduler should no longer schedule packets from this flow. The flow should be removed from the $BackloggedFlows$ list of the channel, thus the number of notified channels will be reduced by one and will become equal to qn_i. A flow can be selected for service on more than one channel at the same time. It is presumed possible to transmit different packets from the same queue on several channels simultaneously. This can be realized when the two packets are stored in different memory blocks which can be read simultaneously.

Next, the algorithm is further clarified by means of its pseudo code but first the variables used are introduced. They are listed in Table 5.2.

For each channel the algorithm keeps a separate $BackloggedFlows_m$ list where the flows, which have packets to be scheduled on the channel, are kept. For each flow there is only one queue $Queue_i$, where the packets classified for the flow are inserted. Per flow there are also the DRR variables, namely the flow's quantum Q_i^m and deficit counter DC_i^m, assigned as discussed in Section 5.3. The BDRR does not notify all channels from the flow's bonding group unless there are sufficient packets in the flow's queue to have at least one packet transmitted per channel. Each flow keeps an indicator $UnusedChannels_i$ of the channels of its bonding group which are not notified yet, i.e., the flow's id is not added in their $BackloggedFlows$ lists. A channel is removed from the set when the flow has notified the scheduler of this channel that it has backlogged packets. When the flow has no more packets for the specified channel it is inserted back

Table 5.2: Bonded DRR Variables

Per channel:
$BackloggedFlows$ lists - for each channel a list with backlogged flows.
Per channel per flow:
$Queue_i$ - pointer to the packets queue;
Q_i^m, DC_i^m - DRR variables for channel m ;
Per flow:
queue - the packets queue;
$UnusedChannels_i$ - set, identifying the ids of the unused channels
n_i - the number of notified channels;
$SequenceNumber_i$ - a counter to track the packets sequence;

into the set. The $UnusedChannels$ set can be realized for example as a list to which channel IDs are added or removed or as a bitmap where a bit is set or reset when a channel is used or not. A variable is kept per flow n_i indicating how many channels at the moment are notified that the flow has packets to transmit, i.e., the flow's id is in their corresponding $BackloggedFlows$ lists. Thus the sum of n_i and the number of channels in the set $UnusedChannels_i$ results in the number of channels in the bonding group of flow i. Finally there is the counter $SequenceNumber$ which is used to tag the packets so their order can be recovered at the destination.

5.5.1 Enqueue Process

The pseudo code of the *Enqueue process* is given in Table 5.3. This function is invoked upon packet arrival at the BDRR scheduler. When a packet arrives at the system it is classified into a flow i and inserted in its corresponding $Queue_i$. If there are channels on which the flow is not considered backlogged, one is selected and notified that the flow has packets to transmit. Consequently, the flow's id is added it to the $BackloggedFlows$ list of the channel and the variables as shown on lines 6 and 7 of the enqueue process are updated. The channel is selected as the next from the channels indicated in the $UnusedChannels$ set, giving preference to those, which have no scheduled transmissions at the moment, resulting in the worst case in O(M) operations. If such a channel is not present, one is selected randomly. At this point the algorithm can be optimized in a future version in order to base the channel selection on some indicators like load or number of backlogged flows. However there is no straightforward way to select the channel on which the flow will receive the fastest service without keeping extra variables.

Table 5.3: BDRR ENQUEUE PROCESS

1. i=p.Flow();
2. $Queue_i.Insert(p)$;
3. $IF(UnusedChannels > 0)$
4. $\quad m = selectChannel(UnusedChannels)$
5. $\quad BackloggedFlows[m].pushBack(i)$;
6. $\quad n_i++$;
7. $\quad DC_i^m = Q_i^m$;

It will be shown later, in the Section 5.7.1, that this choice has a limited effect on the delay. The *selectChannel* function can be implemented also simply as a simple FIFO which would result in O(1) complexity but some efficiency will be lost as it can happen than the selected channel to be informed is currently busy. All other operations perform in constant time. Thus the complexity of the herein implemented enqueue process is O(M).

5.5.2 Dequeue Process

The *Dequeue process* is called independently at each channel scheduler at the events of a flow being added to a channel's empty $BackloggedFlows$ list or when a packet finishes transmission on the channel. The process remains active as long as the list is not empty. The pseudo code is given in Table 5.4. After selecting the next flow i from the $BackloggedFlows$ list its packets are processed until three conditions are met. The first, obvious one, is that there are packets in the queue of the flow. The second one is that the length of the next packet is less than or equal to the deficit counter. These conditions are the same as for the DRR scheduler. The third condition is that there are at least the same number of packets in the queue as the number of channels on which the flow has scheduled transmission, given by n_i. This ensures that when the same flow is selected for service on a different channel there will be at least one packet to be transmitted.

As long as the conditions are met packets are popped from the flow's queue stamped with the sequence number and sent on the channel. The deficit counter for this flow for the channel is reduced by the size of the transmitted packet and the sequence number counter is increased by one. When any of the conditions no longer true holds the process moves on to update the variables.

If the only condition which is not met is the next packet not fitting in the deficit

Table 5.4: BDRR DEQUEUE PROCESS FOR CHANNEL m

```
1. WHILE (Active list for channel m NOT empty)
2.   i = BackloggedFlowsLists[m].popFront();
3.   while(Queue_i NOT empty
            AND nextPacketLength <= DC_i^m
            AND Size(Queue_i) >= n_i)
4.     p = Queue_i.popHead();
5.     p.stamp(SequenceNumber_i)
6.     SequenceNumber_i++
7.     send(p);
8.     DC_i^m = DC_i^m - p.Length();
9.   IF(Queue_i NOT empty
          AND Size(Queue_i) >= n_i)
10.    BackLoggedFlowsList[m].pushBack(i);
11.    DC_i^m += Q_i^m;
12.  ELSE
13.    addChannel(m, UnusedChannels);
14.    n_i--;
15.    DC_i^m = 0;
```

counter, then the flow is inserted at the back of the channels $BackloggedFlows$ list for another round of service and the deficit counter is increased with the quantum for this channel. Otherwise the flow does not require any more service from the channel. This case covers the situation where either there are no more packets in the queue or the number of packets is less than the channels on which the flow is considered backlogged. In both cases the flow should no longer be considered backlogged for this channel thus its the deficit counter is set to 0, the channel is added to the $UnusedChannels$ set and the counter of notified channels is decreased.

Note that, as there are common variables read and updated by the different channel schedulers, concurrency issues must be accounted for in order to avoid deadlocks [123]. To ensure the correct working when a scheduler checks the queue size for flow i on line 3 it should be able to continue to line 4 without any other scheduler accessing the same $Queue_i$. The same should hold for lines 5 and 6. Also the operations between lines 12(9) and 14 are performed without other channel schedulers accessing the n_i variable. The longest operation is the sending of the packet on line 7. Thus the scheduler has to ensure that this operation can be done in parallel, i.e., that when a packet is transmitted on one channel the dequeue process on the other channels can be executed simultaneously.

All the other operations perform in constant time thus any locks on lines $3-6$ will pose very low overhead in comparison with the packet transmission.

The reported algorithm for the dequeue process guarantees that the condition given by Equation (5.6) is satisfied after it finishes. The algorithm has O(1) complexity. Thus the total complexity of the BDRR is O(M).

5.6 Theoretical Analysis

For flows, which can receive packets only on a single channel, both algorithms behave like single channel DRR. In order to be able to guarantee the rate of these flows the following inequality must hold for each channel m

$$\sum_{i=1}^{N^m} r_i^m \leq R^m, \tag{5.7}$$

where R^m is the channel's link rate, N^m is the number of flows which have a reserved rate on the channel as given by Equation (5.4). The frame size from Equation (5.5) is bounded by

$$f^m \leq \frac{R^m}{r_{min}} Q_{min} = F^m, \tag{5.8}$$

where F^m denotes the maximum frame size.

These are the same relations between the parameters as those for DRR scheduling on single channels. Thus selecting the rate partitions as given by Equation (5.7) would guarantee the reserved rates for the flows assigned on only one channel under both algorithms.

Further on the latency and maximum delay bound of the BDRR algorithm are calculated. Note that OutQ-DRR is not a latency-rate server for flows which can be distributed on more than one channel as it does not guarantee their reserved rates. For the OutQ-DRR as described in Section 5.4 it is not possible to derive a lower bound on the rate provided for flows assigned on more than one channel.

Even though the DRR scheduler guarantees a minimum rate on each separate channel a flow is subscribed (see Equation (5.3)), the OutQ-DRR does not guarantee that the flow will be subscribed on all channels of its bonding group. As a result the minimum rate allocated for the flow might be less than the sum of the minimum reserved rates on all assigned channels.

Note also that with OutQ-DRR the packets from one flow are not transmitted in

order. BDRR, on the other hand, transmits the packets on the DS link in the order of their arrival. Still some reordering at the receiver is possible due to the different channel rates and packet sizes. That is, while packet k with length $L_{i,k}$ is being transmitted on a channel, packets from flow i with index greater than k could be transmitted. For a given channel, for these packets to arrive before $L_{i,k}$, their transmission must take less time than the transmission of $L_{i,k}$. In the worst case, $L_{i,k}$ is transmitted over the slowest channel of capacity R_{min}. Thus, it takes $L_{i,k}/R_{min}$ seconds to transmit this packet. During this time, channel m can transmit a total of $\frac{L_{i,k}}{R_{min}}R^m$ bytes of packets whose index is greater than k. Hence, the packets with index greater than k that arrive before $L_{i,k}$ can be at most

$$\frac{L_{i,k}}{R_{min}} \sum_{m=2}^{M_i} R^m \leq L_{max}(\frac{R}{R_{min}} - 1) \tag{5.9}$$

bytes. This bound will be achieved when only packets of the considered flow are transmited on the system.

Following the same consideration and not accounting for the processing delays at both sides, a packet of size $L_{i,k}$ will be delayed in the buffer of the CM due to reordering by at most

$$\frac{L_{max}}{R_{min}} - \frac{L_{i,k}}{R_{max}} \tag{5.10}$$

seconds. The second term is the time it takes for the full packet k to be received and the first term is the maximum time it takes a packet with index less than k to arrive at the receiver.

In the previous chapter the latencies of two single channel algorithms were calculated. For the multi channel BDRR the latency will be derived in the following theorem from the latency of the single channel DRR.

Theorem 5.6.1. *Bonded DRR is a Latency-Rate server with latency determined by*

$$\theta_i^{BondedDRR} = max_{m \in \mathcal{M}_i} \theta_i^{m,DRR} + \frac{(M_i - 1)L_{max}}{r_i},$$

where M_i is the number of channels flow i is assigned to, r_i is its guaranteed rate, L_{max} is the maximum packet size on the network and $\theta_i^{m,DRR}$ is the latency of the DRR scheduler on channel m.

Proof. Consider a channel bonded system with M channels, total bandwidth capacity R

and a BDRR scheduler. Let the bandwidth of channel m be R^m and thus $R = \sum_{m=1}^{M} R^m$. Suppose (t_0, t_1) is flow i's busy period coinciding with the start of a backlogged period at t_0, i.e. $q_i(t_0) = 0$ and t is some time instant within this busy period $t_0 < t \leq t_1$. Recall from Section 1.3.2 that flow i's j-th busy period is defined as the maximum interval in which the traffic for a flow, A_i, arrives at a rate higher than or equal to the flow's reserved rate. The bonding group of flow i consists of M_i channels and its reserved rate r_i is distributed over these channels resulting in channel rates according to Equation (5.4), which are bounded by Equation (5.7).

The number of channels n_i on which the flow is assigned at each moment is given by Equation (5.6). Depending on the number of assigned channels n_i at time t two cases can be distinguished.

Case 1 : $n_i \leq M_i - 1$. From Equation (5.6) follows that there are at most $M_i - 1$ packets in the queue. Thus for the queue state expressed in bits at time t we can write

$$q_i(t) \leq (M_i - 1)L_{max}. \tag{5.11}$$

On the other hand the queue state depends on the arrivals and departures by the relation

$$q_i(t) = A_i(t_0, t) - W_i(t_0, t) + q_i(t_0), \tag{5.12}$$

where $A_i(t_0, t)$ is the number of bits that arrived for flow i during the interval (t_0, t), $W_i(t_0, t)$ is the number of bits that were transmitted, i.e., the amount of service scheduled to flow i in this period and $q_i(t_0)$ is the queue state at time t_0. Expressing W_i from Equation (5.12) and taking into account that the queue state at time t_0 is 0, the service received by flow i in the interval (t_0, t) can be bounded by

$$\begin{aligned} W_i(t_0, t) &= A_i(t_0, t) - q_i(t) + q_i(t_0) \\ &\geq A_i(t_0, t) - (M_i - 1)L_{max} \\ &\geq r_i(t - t_0) - (M_i - 1)L_{max}. \end{aligned} \tag{5.13}$$

The first inequality comes from Equation (5.11) and $q_i \geq 0$. The last inequality comes from the definition of a busy period (see Equation (1.9)).

Case 2 : $n_i = M_i$. In this case the queue state at time t can not be bounded. In the period (t_0, t) there can be a number of intervals which satisfy this condition. Without loss of generality let t_{M_i} be the time instant when the number of packets qn_i in the queue of flow i becomes M_i and remains greater than or equal to M_i until t.

The service received by flow i in the interval (t_0, t) can be split with respect to t_{M_i}

as

$$W_i(t_0,t) = \sum_{m=1}^{M_i} W_i^m(t_0,t_{M_i}) + \sum_{m=1}^{M_i} W_i^m(t_{M_i},t). \tag{5.14}$$

At time $t_{M_i}^-$ the number of packets in the queue is $nq_i \leq M_i - 1$. The same is valid for the number of channels on which the flow is subscribed which corresponds to case 1. Thus the work done in the interval (t_0, t_{M_i}), $W_i(t_0, t_{M_i})$, is bounded as Equation (5.13). Accounting for the considered interval this gives a lower bound on the first term of Equation (5.14) in the form

$$\sum_{m=1}^{M_i} W_i^m(t_0,t_{M_i}) = W_i(t_0,t_{M_i}) \geq r_i(t_{M_i} - t_0 - \frac{(M_i-1)L_{max}}{r_i}). \tag{5.15}$$

In order to find a bound on the second term of Equation (5.14) we have to take into account that the intervals (t_m, t) are backlogged intervals for the corresponding channels. Thus the service on each of the channels can be bounded from the bounds for the scheduler on a single channel if such exists. In the proposed bonded DRR algorithm the scheduler on each channel is DRR. The latency θ_i^{DRR} was derived in [91] and [94] based on a backlogged period. Thus the service received on each channel m in the backlogged period (t_{M_i}, t) is bounded by the DRR latency

$$W_i^m(t_{M_i},t) \geq max(0, r_i^m(t - t_{M_i} - \theta_i^{DRR,m})). \tag{5.16}$$

The service received on channel m in a backlogged period (t_{M_i}, t) is related to the latencies of all channels by the bound

$$\begin{aligned} W_i^m(t_{M_i},t) &\geq max(0, r_i^m(t - t_{M_i} - \theta_i^{DRR,m})) \\ &\geq max(0, r_i^m min_{k\in\mathcal{M}_i}(t - t_{M_i} - \theta_i^{DRR,k})) \\ &\geq max(0, r_i^m(t - t_{M_i} - max_{k\in\mathcal{M}_i}\theta_i^{DRR,k})). \end{aligned} \tag{5.17}$$

Summing Equation (5.17) over all assigned channels $m \in \mathcal{M}_i$ gives

$$\sum_{m=1}^{M_i} W_i^m(t_{M_i},t) \geq max(0, r_i(t - t_{M_i} - max_{k\in M_i}\theta_i^{DRR,k})). \tag{5.18}$$

Replacing the bounds given by Equations (5.18) and (5.15) in Equation (5.14) the lower bound on the service for a period (t_0, t) is obtained to be

$$W_i(t_0, t) \geq max(0, r_i(t - t_0 - \frac{(M_i - 1)L_{max}}{r_i} - max_{m \in \mathcal{M}_i}\theta_i^m)). \tag{5.19}$$

From the definition of LR server (Equation (1.8)) the latency is expressed from Equation (5.19) as the one stated in the theorem. Unlike the latencies which were derived in Chapter 4, which were based on a backlogged period, in this proof the latency was derived from the busy period (t_0, t). For the latencies used for the DRR schedulers in a backlogged period has already been shown that they are a lower bound. Thus the expression for the BDRR latency obtained from Equation (5.19) is the latency of the scheduler. This concludes the proof. □

To show that the latency bound is tight an example where it is actually achieved is given. Consider a system with M channels and a flow i assigned to M_i channels. Let all the channels in the set $\mathcal{M}_i$ guarantee the same latency for flow i $\theta_i^1 = \theta_i^2 = ... = \theta_i^{M_i}$ except channel m which guarantees higher latency, i.e., $\theta_i^m > \theta_i^1$. Figure 5.3 shows an illustration of the considered example for the latency bound of the BDRR algorithm. At time t_0 a packet arrives for flow i. Channel 1 is notified and the flow is inserted in its $BackloggedFlows_1$ list. Suppose that in the interval (t_0, t_m), $M_i - 1$ more packets arrive and the flow receives no service. As a result at time t_m all channels from the DCS of the flow are notified with the last being m at time t_m. $t_0 + \theta_i^1$ marks the time when the flow starts to receive service on channel 1 at its guaranteed rate for the channel r_i^1. $t_m + \theta_i^m$ marks the worst case time instant that the flow starts to receive service at its reserved rate r_i^m for channel m. In the interval $(t_0 + \theta_i^1, t_m + \theta_i^m)$ it starts to receive its reserved rate also on all the other channels. This follows from θ_i^m being the maximum latency. Thus $t_m + \theta_i^m$ is the time instant where the flow starts to receive service at its guaranteed rate r_i. For the time interval (t_0, t_m), $M_i - 1$ packets arrived, which in the worst case is $(M_i - 1)L_{max}$ bits. As this interval is part of a busy period for the flow the minimum arrival rate in this interval is r_i and thus in the worst case

$$t_m - t_0 \leq \frac{(M_i - 1)L_{max}}{r_i}.$$

Thus when this is added to the latency of the m-th channel, which is the channel with maximum latency on the system, it can be easily verified that the latency bound is exactly met.

In Table 4.6 the latency of the DRR scheduler was given as calculated in [94]. Suppose

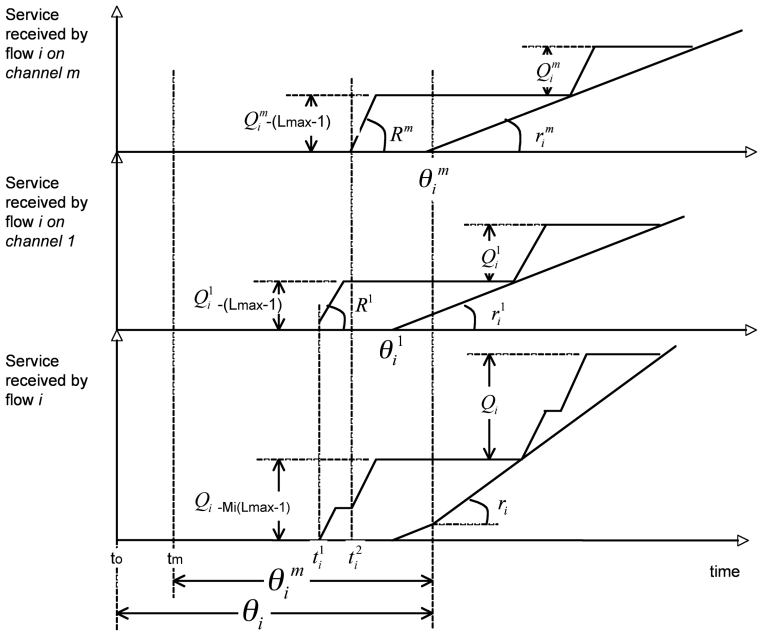


Figure 5.3: Illustration of the latency bound

the maximum channel latency is achieved on channel m, where the DRR parameters are $F^m, Q_i^m, N^m, R^m, r_i^m$. Replacing the latency of the DRR scheduler in the latency expression for BDRR given by Theorem 5.6.1 results in

$$\theta_i^{BDRR} = \frac{F^m - Q_i^m + (N^m - 2)(L_{max} - 1)}{R^m} + \frac{L_{max} - 1}{r_i^m} + \frac{(M_i - 1)L_{max}}{r_i}. \tag{5.20}$$

A number of conclusions can be drawn from the latency bound of the BDRR. Firstly, the latency of the bonded system depends on the latency of the schedulers on the different channels. Thus to minimize the latency of the bonded algorithm the latency on the single channels has to be minimized taking into account the frame length, the channel rate, the number of flows assigned on the channel and the partitioning of the flows' rates on the channels.

Secondly it indicates that the higher the number of channels a flow is assigned to, the higher the latency is. Thus for a flow with reserved rate of say 4Mb/s with respect to delay performance it will be better to assign it to only 1 channel than to spread it on 4 channels with reserved rate on each of them of 1Mb/s. However if assigned on only

one channel the peak rate the flow can get will be restricted. Practically flows from time critical applications like voice or gaming are better assigned on a single channel. Flows from applications with highly variable traffic are better assigned on multiple channels.

The maximum packet delay at the receiver for a leaky bucket shaped traffic with parameters (σ_i, r_i) is obtained from Equation (1.11) from Section 1.3.2 plus the reordering delay from Equation (5.10), plus the packet $L_{i,k}$ transmission delay resulting in

$$D_{max} \leq \frac{\sigma_i}{r_i} + \theta_i^{BDRR} + \frac{L_{max}}{R_{min}} \tag{5.21}$$

5.7 Simulation Results

A variety of simulation results for average and maximum delay and throughput, under different traffic scenarios are presented in this section. The aim is to evaluate the performance of the algorithms and to point out the differences between them. To do so an HFC network with DOCSIS 3.0 MAC protocol is simulated. The simulation program is an enhanced version of the OMNET++ simulator from Section 3.

The simulated system has 4 channels, each having the same downstream (DS) bandwidth of $R^m = 40Mb/s$ resulting in a total system capacity of $R = 160Mb/s$. Specifically in this study a straightforward rule to assign the flows' channel rates is selected. Namely, for flows assigned to a bonded group of M_i channels, the reserved rate on any one of the channels is determined from $r_i^m = r_i/M_i$. It is straightforward to see that the rule from Equation (5.4) is satisfied.

Further on the term "k-channel flow" refers to a SF, that can receive simultaneously on k channels, i.e., its bonding group has size k.

5.7.1 Balanced Load

In this subsection the results for the packet delay from the two algorithms for bursty and leaky-bucket shaped traffic sources when the channels' loads are balanced is reported. The packet delay is measured from the moment a packet enters the scheduler, i.e., after the traffic shaper until it is ready to be transmitted from the SF to its client, i.e., after the packet order is restored at the SF. Thus any transmission and reordering delays are also taken into account.

For the scenarios in this subsection 84 active DS service flows are set up divided in six groups depending on the number of channels in their bonding group and their weights. The groups are summarized in Table 5.5.

Table 5.5: The simulation setup of the flows and their reserved rates

Group Nr	BG size	Nr of flows	r_i [Mb/s]	r_i^m [Mb/s]	Q_i^m [bits]	D_{max} [s]
1	4	8	4	1	12144	0.045
2	2	4	4	2	24288	0.032
3	2	8	2	1	12144	0.045
4	1	48	1	1	12144	0.045
5	1	8	2	2	24288	0.032
6	1	8	4	4	48576	0.026

The first column gives the index of the group, for further reference. The second column indicates the number of channels in the bonding group of the flows from this group. The third column indicates how many flows there are in this group. The forth column indicates the reserved rate r_i per flow. This rate should be divided by the number of channels to achieve the flow's channel reserved rate r_i^m, which is given in the fifth column. The flows from each group are distributed equally amongst the channels. For example the 48 1-channel flows with reserved rate 1Mb/s are assigned 12 per channel. The four, 2-channel flows are assigned as follows: two for channels 0 and 1 and two for channels 2 and 3. On each channel there are $N^m = 30$ flows, which can receive. The described allocation gives total reserved rate per channel of 40Mb/s and thus total reserved rate on the system is 160Mb/s. The minimum quantum Q_{min} on all channels is set to the maximum packet size $L_{max} = 1518$ bytes or 12144 $bits$. The channel quantums of the flows, given in the seventh column, are obtained from Equation (5.1). On each channel the frame length is 485760 bits or 12.144ms. For leaky bucket shaped traffic with parameters for the maximum rate being the reserved rate per flow and the maximum burst size being 12144 bits the maximum delay is calculated from Equation (5.21) taking into account the latency from Equation (5.20). The values are given in the last column of Table 5.5.

The traffic generation process for each flow is ON/OFF with exponentially distributed ON times with mean $2s$ and exponentially distributed OFF times with mean $8s$. The inter-arrival time of the packets is exponentially distributed with values depending on the desired load. The data load is varied from 0.33 up to 0.9 of the total link rate. The rate of the flows is increased correspondingly in order to obtain the desired total load on the channels and is varied in the range [0.33,0.9] times the flow's reserved rate. In this way the load from the 1-channel flows is the same for all channels. The results show the packet queuing delay from the moment the packet arrives at the scheduler until

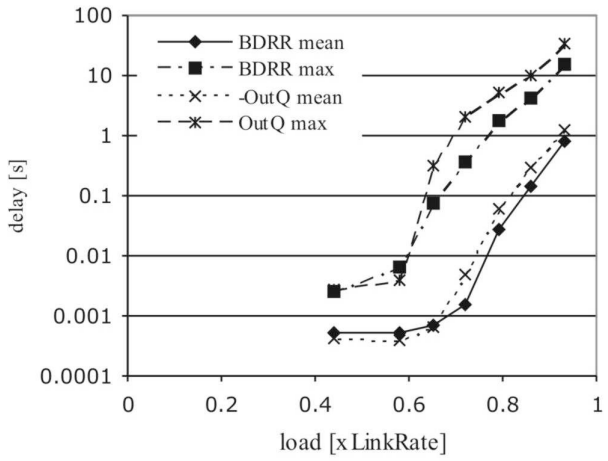


Figure 5.4: Delay vs. load for a 4-channel flow with weight 4

its last bit is transmitted on the DS channel. For each flow the delay is measured over more than a million packets. Each point on the figures is obtained by averaging out the packet delay from all flows in the corresponding group.

Figure 5.4 shows the average and maximum delay for the 4-channel flows with weight 4 (group 1) versus the total load. At loads below $0.6 LinkRate$ the average delay for 4-channel flows for OutQ-DRR is less than for BDRR. This can be explained by the way the two algorithms select the channels on which to serve lightly loaded flows. When a packet arrives at the BDRR scheduler the channel on which the flow will be addded for service is randomly selected. The OutQ-DRR will select the channel with the lowest load. As a result at low loads below $0.6 LinkRate$ the OutQ-DRR achieves lower delay than the BDRR for 2- and 4-channel flows.

With the increase of the load above $0.6 LinkRate$ the average delay achieved under both algorithms increases. The $0.6 LinkRate$ load marks the region where the sum of the loads of the sources in their ON period becomes bigger than the total load. In this region the delay of packets from 4-channel flows served with BDRR is ≈ 2 times less than the one for packets served with OutQ-DRR. This is explained by the different way both algorithms deal with bursts. When a burst of packets arrives in a system served by an OutQ-DRR, the algorithm will add the flow to the $BackloggedFlows$ list of the channel with the lowest reserved load at this moment. If soon after another packet for the same flow arrives, while there is no change of the load state of the channels, it will be added to the same channel. As a result the packets from the many-channel flows will

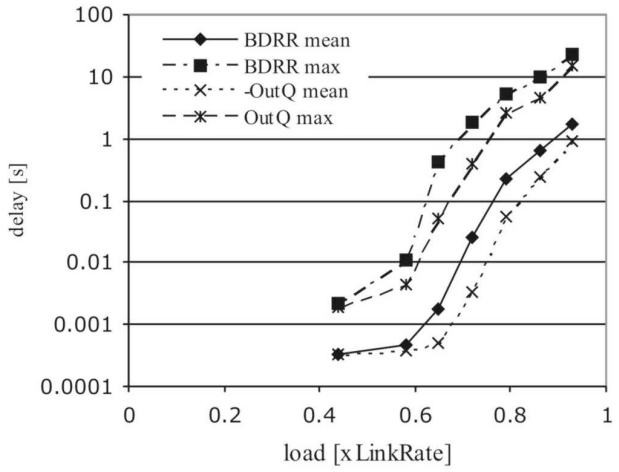


Figure 5.5: Delay vs. load for 1-channel flows with weight 4

be more clustered on one of the channels and more packets will be transmitted out of order. The BDRR algorithm however will spread the burst of packets equally amongst all assigned channels. Even though the average load on the channels from the traffic from the 1-channel flows is equal, there are temporary variations. The BDRR algorithm will use these variation of the channel loads to serve packets on temporally lower loaded channels. As a result the burst will be transmitted faster under BDRR.

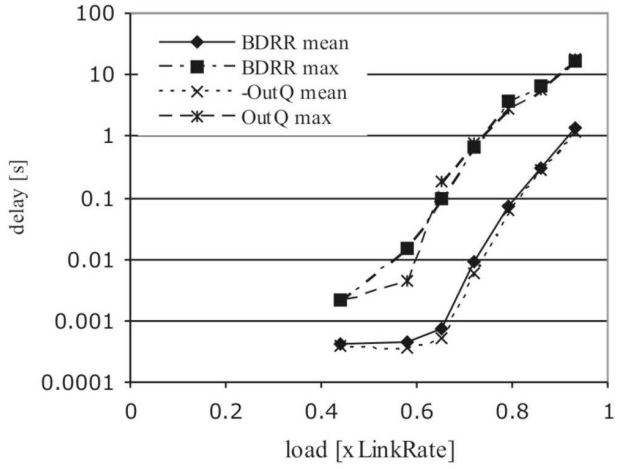


Figure 5.6: Delay vs. load for 2-channel flows weight 4

Figure 5.5 shows the delay for flows assigned to only 1 channel with weight 4. These

results would account for the legacy DOCSIS modems. Note that for the 1-channel flows no reordering of the packets at the SF is needed. At loads below $0.6LinkRate$ the average packet delay for 1-channel flows is similar for both algorithms. For loads above $0.6LinkRate$ the OutQ-DRR provides lower packet delay. This result is due to the fact that under the OutQ-DRR scheduling discipline the packets from flows assigned on more than one channel are more clustered on only one channel. This effect is more pronounced at the 4-channel flows as was discussed above. In this case they receive their fair share according to their channel reserved rate r_i^m. On other channels they will not have sufficient backlog to utilize their share and thus, the fair share for the 1-channel flows on these channels will be bigger than their actual reserved rate. Under BDRR the flows assigned on more channels receive service on all channels they are assigned to during their active periods. As a result the 1-channel flows would of course receive their rate-proportional fair share on the channel but it will be less than the one received under OutQ-DRR.

For flows assigned on 2 channels the results for the delay under both algorithms are very close. They are shown in Figure 5.6. The different treatment of flows assigned on more channels is not pronounced enough and the clustering effect on the 1-channel flows does not influence the 2-channel ones. Results for 2-channel flows with different weights show very similar behavior.

Shaped traffic

In this simulated scenario the traffic from two flows from each group pass a leaky bucket traffic shaper with parameters for the maximum rate being the reserved rate per flow and the maximum burst size being 12144 bits. The values for the theoretical maximum delay for the BDRR algorithm for this traffic are given in the last column of Table 5.5.

Figure 5.7 presents the average and the maximum packet delay for the 1-channel flows with weight 4 (i.e., reserved rate 4Mb/s). Under the both algorithms the delay does not exceed the maximum value. Recall that for 1-channel flows Bonded DRR and OutQ-DRR should behave like regular single channel DRR. Hence, the maximum delay of both algorithms is the same as for single channel DRR.

Figure 5.8 shows the packet delay of a 2-channel flows with weight 4. Under the BDRR the maximum packet delay remains within the calculated theoretical limit for all loads. Under high loads, under the OutQ-DRR scheduling, the total reserved rates on the channels ρ_m does not change significantly in the interval between packet arrival of the high rate leaky bucket shaped flows. As a result the packets from the 2-channel flows

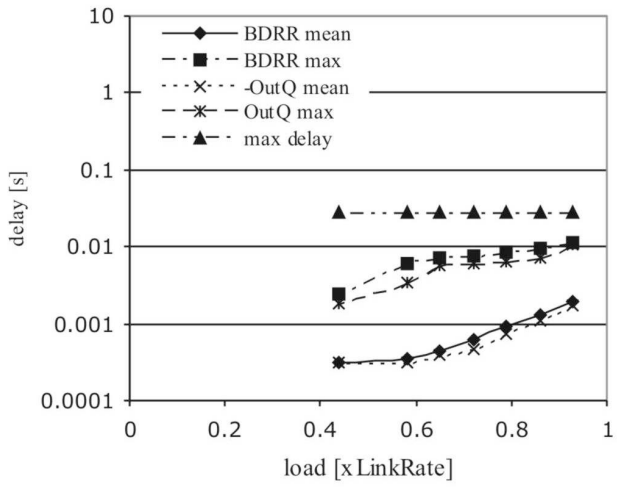


Figure 5.7: Delay vs. load for leaky-bucket shaped traffic 1-channel flows with weight=4

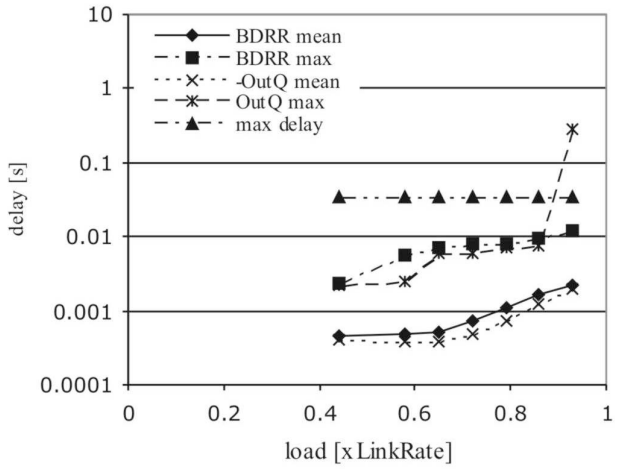


Figure 5.8: Delay vs. load for leaky-bucket shaped traffic 2-channel flows with weight 4

can get clustered on one of the channels, hence more packets will be transmitted out of order. This effect occurs even at lower load for flows, which can be transmitted on more channels like the 4-channel flows shown in Figure 5.9. As can be seen the maximum delay under the BDRR algorithm remains under the calculated upper bound.

Clearly the BDRR performance is better with respect to packet delay for flows which can receive on more than 1 channel. The BDRR can also guarantee maximum delay for

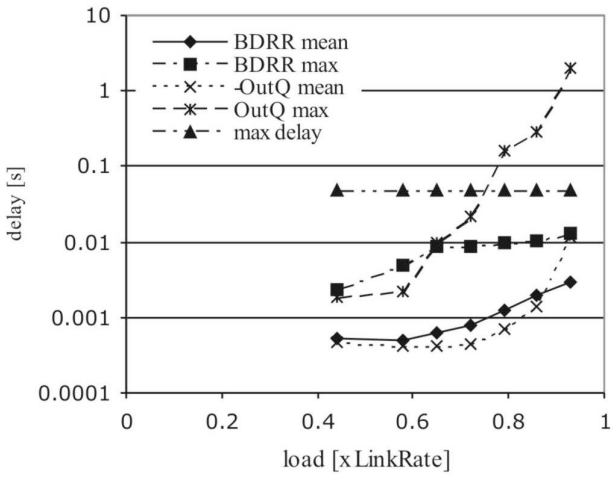


Figure 5.9: Delay vs. load for leaky-bucket shaped traffic for 4-channel flows with weight 4

leaky bucket shaped traffic and bounds the reassembly buffer space (the bound is given by Equation (5.9)). As buffer space is an expensive resource at the subscriber's equipment, such bound could be further explored in the design and dimensioning of access nodes.

5.7.2 Imbalanced load

A real system cannot guarantee that the total load on all channels will be similar. On the contrary it is more likely that for certain periods of time some channels will be more loaded than others. This section studies how an imbalanced load affects the average packet delay under the two algorithms..

For this purpose the same simulation setup as in the previous section is used but without varying the traffic of the 4 and 2-channel flows. They transmit traffic at 0.7 times their respective reserved rate. The 64, 1-channel flows are used to generate background traffic. The traffic from the 1-channel flows assigned on channels 1 and 3 is varied from 0.7 down to 0.1 their reserved rate. These channels are referred to as "light loaded". The traffic from the 1-channel flows assigned on channels 2 and 4 is varied from 0.7 up to 1.3 from their reserved rate. These channels are referred to as "heavy loaded". The total load for each simulation run is approximately 0.7 of the link rate. When the 32 1-channel flows on the light loaded channels transmit at rate $0.1r_i$, the 32 1-channel flows on the "heavy loaded" channels transmit at rate $1.3r_i$. The ratio between the loads on the "heavy" and the "light" loaded channels, due to the traffic from the 1-channel flows,

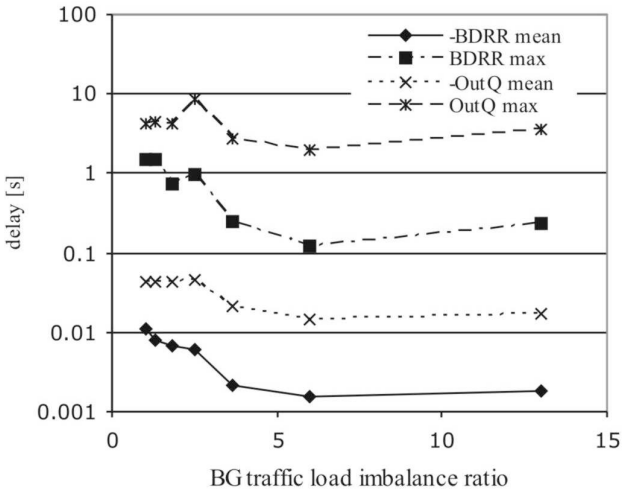


Figure 5.10: Delay vs. BG traffic imbalanced ratio for 4-channel flow with weight 4

is called the background (BG) traffic load imbalance ratio.

Packet Delay

Figure 5.10 shows the results for the packet delay for the 4-channel flows. With increase of the imbalances the average packet delay under both algorithms decreases. However this decrease is more substantial for the BDRR algorithm. The delay of the packets under BDRR is determined from the delay on the light loaded channel. A packet is transmitted on a heavy loaded channel only when the waiting time for transmission there will be less than if it would be transmitted on the light loaded one. For the OutQ-DRR this is not the case. When a burst of packets is scheduled on a channel which is currently low loaded, it can become high loaded soon after and the packets will endure long delays. Thus the most likely load distribution on a real system - the imbalanced one- favours BDRR scheduling.

For the groups which are assigned on more than 1 channel we let the traffic from two flows from each one be generated as leaky bucket traffic with parameters $(12144bits, r_i)$. The results for the average and maximum packet delay for the 4-channel flows with weight 4 are shown on Figure 5.11. The maximum delay for the packets for flows scheduled under BDRR remains always below the calculated maximum theoretical limit. The maximum delay under OutQ-DRR is significantly higher than the one under BDRR for low BG traffic load imbalance ratios. This is in conformance with the results for load

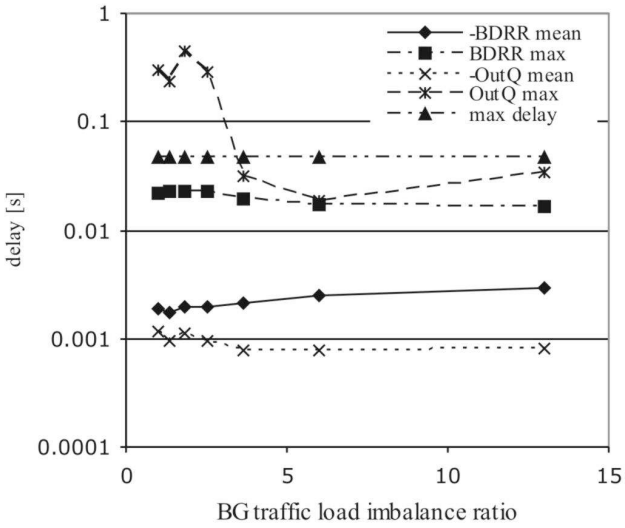


Figure 5.11: Delay vs. BG traffic imbalanced ratio for leaky bucket shaped traffic 4-channel flows with weight 4

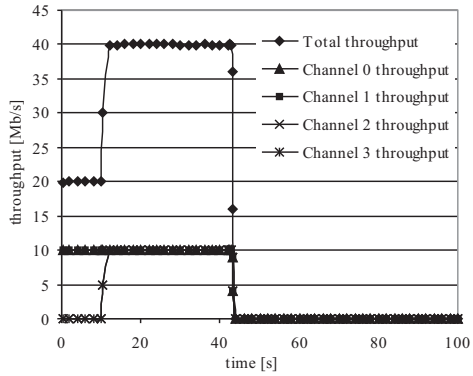
$0.7LinkRate$ when the load is balanced (Figure 5.9). With the increase of the load imbalance however, more and more packets are transmitted on the light-loaded channel and the maximum delay becomes more comparable with the one achieved under BDRR.

Throughput

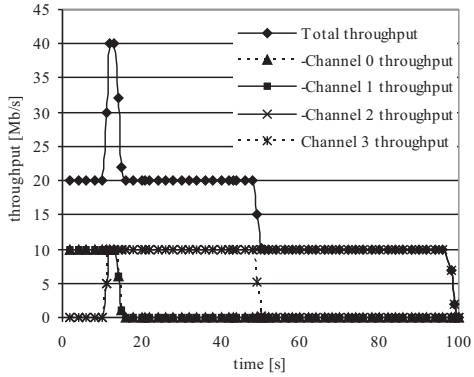
This simulation setup demonstrates how in the presence of long bursts the OutQ-DRR fails to utilize the available resources on all channels. The simulated system has 4 channels each with rate of 10 Mb/s for a total of 40Mb/s. There are 32 1-channel flows distributed 8 per channel. There are also two 4-channel flows assigned on all the channels. The 1-channel flows have each one 2 Mbyte file to be transmitted on the DS. For the flows on channels 0 and 1 this file arrives at the system at time 0s. For the flows on channels 2 and 3 it arrives 20s later. The two 4-channel flows become active at time 10s and have each one large file of 60 Mbytes to transmit.

In Figure 5.12(a) and (b) the resulting total and per channel DS throughput for the BDRR and OutQ-DRR is shown. Note that the throughput on channels 0 and 1 is the same as well as the throughput on channels 2 and 3.

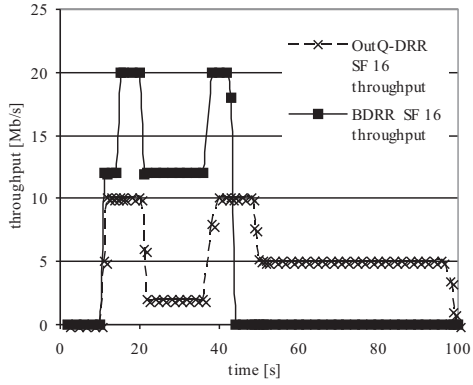
In *the first 10 seconds* the flows assigned on channels 0 and 1 fully use their capacity resulting in total throughput of 20Mb/s for both algorithms. At time 10s the two 4-channel flows begin transmission. Their throughput under the two algorithms is shown



(a)



(b)



(c)

Figure 5.12: The throughput realized (a) under the BDRR algorithm (b) under the OutQ-DRR algorithm (c) for a 4-channel flow under the BDRR and OutQ-DRR algorithms

in Figure 5.12(c). Their throughput under BDRR for *the period 10-12s* is 12 Mb/s. This is as expected and can be calculated as follows. The guaranteed bandwidth of all active flows just before the two 4-channel flows become active is $16 \times 1Mb/s$ resulting in free bandwidth of $40 - 16 = 24Mb/s$. This bandwidth is distributed equally amongst the two 4-channel flows. At 10s, when the two 4-channel flows become active, the OutQ-DRR queues their packets onto the currently free channels 2 and 3. They share equally the bandwidth resulting in 10Mb/s per flow. As long as the flows assigned on channels 0 and 1 have packets to transmit the utilization of the system is 100%. However when the 1-channel flows assigned on channels 0 and 1 have no more packets the utilization under OutQ-DRR falls to 50% because the packets from the 4-channel flows are queued on channels 2 and 3. At *approximately 13.5s* the active 1-channel flows finish the transmission of their file under BDRR. Now the only active flows are the two 4-channel flows. The algorithm distributes the free bandwidth to the 4-channel flows allocating 20Mb/s per flow. During the whole period, the 4-channel flows are being active, their packets are distributed on all channels, resulting in full utilization of the available bandwidth under BDRR.

At *time 20s*, the 1-channel flows, assigned on channels 2 and 3, become active. Thus the packets from all the currently active flows are distributed only on channels 2 and 3. The throughput for the 4-channel flows under OutQ-DRR drops to 2Mb/s as can be seen on Figure 5.12(c). The BDRR algorithm on the other hand transmits the majority of the packets from the 4-channel flows on channels 0 and 1. The same calculations as for the period 10-12s can be applied but this time the flows receive 10Mb/s on channels 0 and 1 and 2Mb/s on channels 2 and 3.

After the flows on channels 2 and 3 finish their transmissions at approximately $38s$ under the OutQ-DRR the 4-channel flows utilize the bandwidth from the two channels. However, when the burst of packets for 4-channel flows was distributed to the channels, more packets were queued at channel 2. As a result, after the packets queued on channel 3 are transmitted, the system utilization drops to 25%. When the 1-channel flows assigned on the channels 2 and 3 finish their transmission under the BDRR the two 4-channel flows again get each half of the total bandwidth until their queues become empty.

This scenario shows that the OutQ-DRR fails to utilize the full capacity of the system. BDRR on the other hand distributes the packets of the 4-channel flows on all channels succeeding to utilize the full link capacity.

5.8 Conclusions and Future Work

In this chapter two scheduling algorithms with distributed architecture, applicable for networks with channel bonding were proposed. Due to the distributed architecture, where each algorithm consists of several Deficit Round Robin single channel algorithms, the scheduling is work conserving even when flows can be distributed only on a selection of the channels. The algorithms differ on the type of queuing they use and on the level of independence of the single channel schedulers.

The output queuing algorithm, termed OutQ-DRR, uses per-channel, per flow queuing and a load distributor, which selects on which channel a packet will be transmitted at the time of the packet arrival. A single channel DRR algorithm schedules between the queues on a channel. The algorithm is unable to fully utilize the available bandwidth but is straightforward to apply as the channels are served independently from each other. It requires extra space for each packet in order to store its sequence number.

The input queuing algorithm, named Bonded Deficit Round Robin (BDRR) keeps only one queue per flow and has also different schedulers per channel. However these schedulers are not fully independent but share common variables. Its packet processing complexity depends only on the number of channels. As an input queuing algorithm it does not use extra buffer space to store the sequence number with the packet. It can fully utilize the available bandwidth. It is shown that the BDRR scheduler is a latency rate scheduler and that it results in a bounded packet reordering. Thus it can bound the maximum delay for leaky bucket shaped traffic. It also bounds the maximum number of bytes that can be transmitted out of order and hence the buffer space at the receiver side.

The performance of both algorithms is further analyzed via simulations for a variety of traffic and load scenarios. The BDRR has better performance in terms of average packet delay when the packets of a flow are distributed on more than one channel. The BDRR also outperforms the output queuing algorithm when the load on the system is not balanced amongst the channels.

In order to address the fairness of the algorithm the different capabilities of the modems, in terms of the number of channels they can receive on, have to be accounted for. As it is defined, the Golestiani fairness is not suitable to study the problem.

The applicability of the proposed algorithms for upstream scheduling in channel bonded access networks also requires further investigation.

Chapter 6

Conclusions and Future Work

This dissertation addressed several of the issues involved in the design and analysis of scheduling algorithms for fixed point-to-multipoint subscriber access networks. Scheduling algorithms are largely responsible for ensuring the different QoS guarantees and providing fairness on the network, which makes them an important design issue. The challenges in the design of these algorithms are ensuring fairness, flow isolation and low complexity in the downstream and efficient bandwidth utilization in the upstream. Emerging technologies for high-speed access using coaxial and optical cable as a physical medium were considered. For each technology, novel algorithms were designed which optimize the bandwidth utilization, the complexity or the delay. Simulation tools were developed to assist in the analyzes of these algorithms. All algorithms were simulated to demonstrate their performance in representative scenarios. Some algorithms were analyzed theoretically in order to establish their latency and fairness bounds.

Hereafter a more detailed summary of the basic contributions of the thesis is given.

6.1 Upstream Bandwidth Allocation in Ethernet Passive Optical Networks (EPON)

While several upstream scheduling algorithms for EPON have been proposed in the literature the contribution of the thesis is in introducing the novel threshold reporting mechanism in the scheduling. Methods to generate reports with thresholds at the ONU and to process them at the OLT in a way suitable to base the scheduling decision on these reports, were designed. An upstream scheduling algorithm for EPON is defined by the scheduling mechanisms in both the ONU and in the OLT. Two types of scheduling at the ONU were considered, namely full priority scheduling, which is the common

strict priority scheduling and interval priority scheduling, where higher priority traffic is transmitted before lower priority only if it was already reported to the OLT. The types of algorithms proposed for the OLT differ by threshold level on which they base their scheduling algorithm - full and single report threshold level - and on their scheduling policy for constant bit-rate traffic. The algorithms are analyzed by means of a detailed simulation program, regarding average packet delay, delay variation and bandwidth utilization. It is shown that by using threshold reporting and interval priority scheduling at the ONU the bandwidth can be fully utilized. Combining it with rate based scheduling at the OLT provides an interesting tradeoff between the efficiency, which is still near to the optimal, and the delay characteristics of time critical applications.

6.2 Simulation framework for DOCSIS 2.0 based HFC networks in OMNET++ and Performance Evaluation of Upstream Scheduling Services

The DOCSIS 2.0 specification offers a wealth of possibilities for the cable operators to provide high speed quality access to their subscriber. It is thus important to have a real-system simulator which allows to study the influence of the different standardized system parameters and scheduling types. A simulator of the DOCSIS 2.0 based HFC networks was implemented in C++ using OMNET++. Unlike other free DOCSIS simulators it models in detail the physical medium dependent layer and many features of the MAC layer. It models in detail various QoS mechanism from the DOCSIS MAC protocol. This includes the upstream scheduling services: unsolicited grant service (UGS), real time polling service (rtPS) and best effort (BE) service. The different polling mechanisms implemented are uni-cast request, bandwidth request in contention slots and piggybacking. The performance of the rtPS and BE scheduling services is evaluated via simulations and an improvement to the common rtPS scheduling is proposed. Optimal values for some contention channel and MAC protocol parameters were obtained.

The simulator can be used to evaluate the performance of a DOCSIS system in different scenarios, to fine tune the values of some parameters and to evaluate different scheduling algorithms, which is of great use for cable operators. It has already been used to simulate and evaluate the performance of the actual scheduler implemented in Motorola BSR 64000 CMTS/Edge routers, which are commercially deployed by a cable operator in Belgium.

6.3 Packet Scheduling Algorithms for Single Channel

The major requirements for downstream packet scheduling algorithms in high-speed networks are the low complexity and guaranteeing flow isolation, fairness, low end-to-end delay and bandwidth utilization. A new scheduler called *Last Backlogged First Served - Deficit Round Robin*, was proposed. In comparison with the deficit round robin algorithm, it provides lower average packet delay, while preserving the advantageous feature like O(1) complexity, fairness and bandwidth guarantees. The lower mean delay is realized by giving service in a round first to flows transmitting bellow their (weighted) fair share. The algorithm exploits the high variability in the typical user traffic pattern resulting in lower mean file transfer delay. The implementation for the *Surplus Round Robin* (SRR) algorithm proposed in this thesis has also O(1) complexity and succeeds in guaranteeing fairness and delay bounds for leaky-bucket shaped traffic. The fairness and latency bound of the algorithm are also derived analytically. Both algorithms are implemented in software and further analyzed by simulations.

6.4 Scheduling Algorithms for Channel Bonded Systems

Channel bonding or the bonding of several channels together to create a larger bandwidth pipe, is defined in DOCSIS 3.0 for HFC access networks. To tackle the scheduling problem two algorithms with distributed architecture are designed. They make use of two different queueing mechanisms. The Bonded Deficit Round Robin (BDRR) uses input queuing and can fully utilize the available downstream bandwidth. It is shown that the BDRR scheduler is a latency rate scheduler and that it results in a bounded packet reordering. Thus it can bound the maximum delay for leaky bucket shaped traffic, which is derived theoretically. This is not the case for the output queueing algorithm - OutQ-DRR. Besides that it does not bound the delay, it is also unable to fully utilize the available bandwidth. However it is more straightforward to apply as the channels are served independently from each other. The performance of both algorithms is further analyzed via simulations for a variety of traffic and load scenarios.

6.5 Future Work

Channel bonding is becoming a widespread technology to increase the throughput of an existing infrastructure. Not only in fixed networks but also for wireless networks, a methodology for bonding several channels together is being standardized. However

scheduling algorithms for such systems have not been sufficiently studied yet.

In this thesis we showed that an algorithm for a multi-channel system, where the flows can be scheduled only on selections of the available channels, which can provide delay guarantees can be designed. The latency of the algorithm was derived from the latency of the algorithm used for scheduling on a single channel. In the future these results can be generalized such that also bonded algorithms based on other latency-rate schedulers for single channels can be analyzed.

Another possible extension of the study on system with channel bonding is designing a hierarchical scheduler. In single channel networks such schedulers are often preferred as they allow the operators to reserve bandwidth not only on a per-flow basis but also per application.

Fairness in multi-channel networks, where the users have different capabilities, has not been sufficiently addressed yet. The current fairness metrics should be enhanced to include not only the reserved rate (allocated share) but also the number of channels a customer can use.

In the thesis several upstream scheduling algorithms were designed where the users are contending for the resources on a single channel. Designing an efficient algorithm for upstream scheduling over multiple channels is a new challenge.

Nederlandse Samenvatting

Access netwerken met punt-tot-multipunt topologie verbinden de telecom operator via een gedeelde kabel en/of optische link met zijn klanten. Zulke netwerken hangen af van bandbreedte toekenning algoritmes om de beschikbare bandbreedte op een efficiente manier te gebruiken en om kwaliteit van diensten te kunnen verzekeren. Wat maakt dat deze algoritmes een belangrijke design kwestie worden. De uitdagingen bij het ontwerpen van deze algoritmes zijn: verzekeren van "fairness", isoleren van verkeer van verschillende gebruikers, zorgen voor een lage complexiteit in de downstream en efficient gebruik van bandbreedte in de upstream. Deze thesis beschrijft en maakt een analyse van verschillende algoritmes voor bandbreedte toekenning in hoge-snelheid optische netwerken en in hoge- snelheid kabel acces netwerken.

Hoofdstuk 1 biedt een overzicht van de geanalyseerde acces technologien, Ethernet Passieve Optische Netwerken en Hybride Fiber Coaxkabel (HFC) Netwerken met de Data-Over-Cable Service Interface Specifications (DOCSIS) protocol versies 2.0 en 3.0. Ook biedt dit hoofdstuk een samenvatting van de bestaande bandbreedte toekennings algoritmes en beschrijft het methodes voor de evaluatie van deze algoritmes. Deze methodes zijn software simulaties en theoretische analyse gebruik makend van de "Latency-Rate" theorie. De "latency" laat toe om algoritmes te vergelijken en de maximale wachttijd die een pakket van een "leaky-bucket" gemoduleerd verkeer kan ondervinden te bepalen.

Ethernet Passieve Optische Netwerken (EPON) zijn acces netwerk technologien die hoge snelheid bieden via optische fiber. De Optische Netwerk Unit (ONU) is de uitrusting bij de gebruiker en de Optische Lijn Terminal (OLT) is de uitrusting bij de kabel operator. Het upstream bandbreedte toekenning algoritme in de OLT is verantwoordelijk voor de toekenning van het upstream kanaal aan de verschillende gebruikers. Ze moeten wel eerst hun benodigde bandbreedte rapporteren. Hoewel verschillende upstream bandbreedte toekenning algoritmes voor EPON zijn voorgesteld in de literatuur, *Hoofdstuk 2* introduceert bandbreedte toekenning algoritmes die gebruik maken van het nieuwe

"drempel rapportering" mechanisme. Methoden worden gedefinieerd om zulke rapporten te genereren in de ONU en om ze te verwerken in de OLT op een zodanige manier dat scheduling beslissingen kunnen genomen worden op basis van deze rapporten. Het upstream bandbreedte toekenning algoritme voor EPON is gedefinieerd door de scheduling mechanismen in zowel de ONU als in de OLT. Verschillende algoritmes zijn ontworpen en gevalueerd door middel van simulaties betreffende gemiddelde pakket vertraging, de variatie van de vertraging en het gebruik van de bandbreedte. Het is aangetoond dat door gebruik van " drempel-rapportering " en de geschikte scheduling in de ONU, de bandbreedte volledig benut kan worden. Bovendien kunnen kleine vertraging en lage variatie van de vertraging worden gegarandeerd voor hoge prioriteit, tijd-kritisch applicaties. De algoritmes en resultaten zijn gerapporteerd in [95], [96], [97] en [98] en werden al meer dan 20 keer geciteerd.

De DOCSIS 2.0 specificatie voorziet een waaier van mogelijkheden voor kabel operatoren om internetacces met hoge snelheid aan te bieden aan hun klanten. *Hoofdstuk 3* beschrijft een simulator van DOCSIS 2.0 HFC netwerken geimplementeerd in C++ gebruik makend van OMNET++. De simulator onderscheidt zich van andere vrij beschikbare simulatoren doordat hij in detail de fysische laag en veel kenmerken van de "Medium Access Control" (MAC) laag modelleert . De simulator kan gebruikt worden voor het evalueren van het systeem, om de waarden van verschillende systeem parameters op punt te stellen en om verschillende scheduling algoritmes te vergelijken, wat van het grootste belang is voor kabel operatoren. De prestaties van verschillende scheduling diensten zijn gevalueerd met simulaties en de resultaten zijn beschreven in dit hoofdstuk. Verbetering van de scheduling is uitgewerkt en de optimale waarden van verschillende protocol parameters zijn berekend als resultaat van de simulatie. De simulator en de resultaten zijn gerapporteerd in [107].

In access netwerken, de pakketten die aankomen van het Internet worden in rijen ingevoegd. Een pakket scheduling algoritme is verantwoordelijk voor de bandbreedte distributie aan de gebruikers in de downstream, waar er een rij per gebruiker is. De belangrijkste vereisten voor downstream pakket scheduling algoritmes in hoge snelheid netwerken zijn: lage complexiteit, fairness, het isoleren van verkeer van verschillende bronnen en korte vertragingen. *Hoofdstuk 4* stelt een nieuwe scheduler voor, genaamd *Last Backlogged First Served - Deficit Round Robin* die een verbetering is van het DRR algoritme. In vergelijking met het DRR, voorziet het nieuwe algoritme lagere gemiddelde vertraging van de pakketten, terwijl het de voordelige eigenschappen zoals constante tijd complexiteit, fairness en bandbreedte garantie behoudt. De lagere gemiddelde vertraging is gerealiseerd door in een ronde eerst de pakketen van gebruikers te bedienen die onder

hun (gewogen) "fair" deel van de beschikbare capaciteit verzenden. Het algoritme maakt gebruik van de hoge variabiliteit in het typische gebruikers verkeerspatroon met als resultaat lagere gemiddelde file transfer vertraging. De implementatie van het *Surplus Round Robin* algoritme voorgesteld in dit hoofdstuk heeft ook constante complexiteit en slaagt er in om fairness en begrensde vertraging voor "leaky-bucket" gemoduleerd verkeer te garanderen. De "fairness" en de "latency" van beide algoritmes worden berekend. Beide algoritmes zijn geimplementeerd en verder geanalyseerd aan de hand van verschillende simulaties. De algoritmes en resultaten zijn gerapporteerd in [124] en [125]. Het eerste artikel is verkozen als *Beste artikel voor Telecommunicatie Systemen* op de conferentie waar het werd gepresenteerd.

Om de beschikbare capaciteit te verhogen gebruiken sommige systemen meerdere kanalen tegelijk. "Channel bonding" of het gebruiken van meerdere kanalen tegelijk om een grotere capaciteit te bereiken is gestandaardiseerd voor HFC netwerken in DOCSIS 3.0. Twee algoritmes zijn ontworpen voor scheduling in zulke netwerken. Ze zijn beschreven in *Hoofdstuk 5* . Ze gebruiken twee verschillende manieren om een rij te vormen. Het *Bonded Deficit Round Robin (BDRR)* algoritme gebruikt input rijen. Het algoritme kan de capaciteit van alle beschikbare kanalen gebruiken. Het is aangetoond dat het BDRR een "Latency Rate" algoritme is en zijn 'latency' is berekend. Het algoritme resulteert in een begrensde herschikking van de orde van de pakketten en geeft dus de grens aan van de vertraging van de pakketten van "leaky-bucket" gemoduleerd verkeer. Dit is niet het geval voor het andere algoritme dat "output" rijen gebruikt. Behalve dat het geen grens geeft voor de maximale pakket vertraging, is het ook niet in staat om in alle gevallen de beschikbare capaciteit van alle kanalen te gebruiken. Nochtans is het simpeler om toe te passen omdat de kanaal schedulers helemaal onafhankelijk van elkaar zijn. De prestaties van beide algoritmes zijn verder geanalyseerd met simulaties voor verschillende verkeerspatronen. De algoritmes en een deel van de resultaten zijn gerapporteerd in [126].

Een samenvatting van de belangrijkste conclusies wordt gegeven in *Hoofdstuk 6* .

Bibliography

[1] *Data-Over-Cable Service Interface Specifications DOCSIS 1.0 Radio Frequency Interface Specification*, CableLabs Std. [Online]. Available: www.cablelabs.com/specifications

[2] *Data-Over-Cable Service Interface Specifications DOCSIS 1.1 Radio Frequency Interface Specification*, CableLabs Std. [Online]. Available: www.cablelabs.com/specifications

[3] *Data-Over-Cable Service Interface Specifications DOCSIS 2.0 Radio Frequency Interface Specification*, CableLabs Std., 2001. [Online]. Available: www.cablelabs.com/specifications

[4] *Data-Over-Cable Service Interface Specifications eDOCSIS Specification*, CableLabs Std. [Online]. Available: www.cablelabs.com/specifications

[5] *Data-Over-Cable Service Interface Specifications DOCSIS 3.0 MAC and Upper Layer Protocols Interface Specification*, CableLabs Std., Rev. I05, 2007. [Online]. Available: www.cablelabs.com/specifications

[6] *Media Access Control Parameters, Physical Layers and Management Parameters for subscriber access networks*, IEEE 802.3ah Std., 2003.

[7] *Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*, ANSI/IEEE802.3 Std., 2000.

[8] *Virtual Bridged Local Area Networks*, IEEE 802.1Q Std., 1998.

[9] J. M. N.Shah, D. Kouvatsos and S. Moser, "A tutorial on DOCSIS: Protocol and performance models," in *Proceedings of the International Working Conference on Performance Modeling and Evaluation of Heterogeneous Networks*, July 2005.

[10] J. Martin and N. Shrivastav, "Modeling the DOCSIS 1.1/2.0 MAC protocol," in *Proceedings of the ICCNC*, October 2003.

[11] N. Shankaranarayanan, Z. Jiang, and P. Mishra, "User-percieved performance of web-browsing and interactive data in HFC cable access networks," in *Proceedings of the International Conference on Communications (ICC)*, June 2001.

[12] S.-H. Cho, J.-H. Kim, and S.-H. Park, "Performance evaluation of the DOCSIS 1.1 MAC protocol according to the structure of a MAP message," in *Proceedings of the International Conference on Communications (ICC)*, June 2001.

[13] G. Chandrasekaran, M. Hawa, and D. Petr, "Preliminary performance evaluation of QoS in DOCSIS1.1," The University of Kansas Center for Research, Inc., Tech. Rep., 2003.

[14] N. Naaman and R. Rom, "Packet scheduling with fragmentation," in *Proceedings of INFOCOM*, vol. 1, 2002, pp. 427–436.

[15] W. Liao, "The behavior of TCP over DOCSIS-based CATV networks," *IEEE Transactions on Communications*, vol. 54, no. 9, pp. 1633–1642, 2006.

[16] J. Martin, "The impact of DOCSIS 1.1/2.0 MAC protocol on TCP and TRFC," in *Proceedings of the Second IEEE Consumer Communications and Networking Conference, CCNC*, 2005.

[17] W. Liao and L. H. jiun Ju, "Adaptive slot allocation in DOCSIS-based CATV networks," *IEEE Transactions on Multimedia*, vol. 6, no. 3, pp. 479–488, 2004.

[18] H. jiun Ju and W. Liao, "Adaptive slot allocation in DOCSIS-based CATV networks," in *Proceedings of ICCCN*, 2002.

[19] ——, "Fast request transmission in DOCSIS-based CATV networks," in *Proceedings of International Conference on Multimedia and Expo (ICME)*, 2002.

[20] J. Lambert, B. Van Houdt, and C. Blondia, "Dimensioning the contention channel of DOCSIS cable modem networks," in *Proceedings of Networking 2005*, ser. Lecture Notes in Computer Science, vol. 3462, APR 2005, pp. 342–357.

[21] ——, "Queues in DOCSIS cable modem networks," *Computers and Operations Research, Special Issue on Queues in Practice*, vol. 35, pp. 2482–2496, 2008.

[22] M. Droubi, N. Idrene, and C. Chen, "Dynamic bandwidth allocation for the HFC DOCSIS MAC protocol," in *Proceedings of ICCCN*, 2000.

[23] S.Golestiani, "A self-clocked fair queueing scheme for broadband applications," in *Proceeding of INFOCOM'94*, April 1994.

[24] M. Hawa and D. Petr, "Quality of service scheduling in cable and broadband wireless access systems," in *Tenth International Workshop on Quality of Service*, May 2002.

[25] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *SIGCOMM '89: Symposium proceedings on Communications architectures & protocols*, 1989, pp. 1–12.

[26] P. Goyal, H. Vin, and H. Cheng, "Start-time fair queuing: A scheduling algorithm for integrated services packet switching networks," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 690–704, 1997.

[27] Y. Dong, Z. Zhang, and D. Du, "Optimized upstream scheduling in broadband cable networks," in *Proceedings of the 10th Symposium on High Performance Interconnects(HotI'02)*, 2002.

[28] M. Droubi, N. Idrene, and C. Chen, "Traffic based bandwidth allocation for DOCSIS cable networks," in *Proceedings of ICCCN*, 2002.

[29] W. Yin, C. Wu, and Y. Lin, "Two-phase minislot scheduling algorithm for HFC QoS service provisioning," in *Proceedings of Globecom*, vol. 1, 2001, pp. 410–414.

[30] K. Kim, "On the evolution of PON-based FTTH solutions," *Information sciences*, no. 149, pp. 21–30, 2003.

[31] G. Kramer and G. Pessavento, "Ethernet Passive Optical Network (EPON): Building a next-generation optical access network," *IEEE Communications Magazine*, February 2002.

[32] G. Kramer, N. Singhal, and S. Dixit, "Fair queuing with service envelops: A cousin fair hierarchical scheduler for subscriber access networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 8, October 2004.

[33] G. Kramer, B. Mukherjee, and G. Pessavento, "Interleaved Polling with Adaptive Cycle Time (IPACT): a dynamic bandwidth distribution scheme in an optical access network," *Photonic Network Comm.*, no. 4, pp. 89–107, 2002.

[34] G. Kramer, B. Mukherjee, S. Dixit, Y. Ye, and R. Hirth, "Supporting differentiated classes of service in Ethernet Passive Optical Networks," *Journal of Optical Networking*, vol. 1, no. (8,9), pp. 280–298, 2002.

[35] S. Choi, "Cyclic polling-based dynamic bandwidth allocation for differentiated classes of service in Ethernet Passive Optical Networks," *Photonic Network Comm*, vol. 7, no. 1, pp. 87–96, 2004.

[36] G. Kramer, B. Mukherjee, and G. Pessavento, "Ethernet PON (ePON): Design and analysis of an optical access network," *Photonic Network Comm*, no. 3, pp. 307–319, July 2001.

[37] C. Assi, Y. Ye, S. Dixit, and M. Ali, "Dynamic bandwidth allocation for Quality-of-Service over Ethernet PONs," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 9, pp. 1467–1477, November 2003.

[38] H. Nasser and H. Mouftah, "A joint-ONU interval-based dynamic scheduling algorithm for Ethernet Passive Optical Networks," *IEEE/ACM Transactions on Networking*, vol. 14, no. 4, August 2006.

[39] M. Ma, Y. Zhu, and T. H. Cheng, "A bandwidth guaranteed polling MAC protocol for Ethernet Passive Optical Network," in *INFOCOM*, 2003.

[40] H.Myoshi, T. Inoue, and K. Yamashita, "Qos-aware dynamic bandwidth allocation scheme in gigabit-ethernet passive optical network," in *International Conference on Communications (ICC'04)*, 2004.

[41] A. Banerjee, G. Kramer, and B. Mukherjee, "Fair sharing using dual-service agreements to achieve open access in a Passive Optical Network," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, August 2006.

[42] A. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated service networks: The single-node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344–357, 1993.

[43] L. Kleinrock, "Time-shared systems: A theoretical treatment," *Journal of the Association for Computing Machinery*, vol. 14, no. 2, pp. 242–261, April 1967.

[44] L. Zhang, "Virtual clock: A new traffic control algorithm for packet switching networks," *ACM Computer Communication Review*, vol. 20, no. 4, September 1990.

[45] J. Bennett and H. Zhang, "WF2Q: Worst-case fair weighted fair queueing," in *INFOCOM*, March 1996, pp. 120–128.

[46] D. Stiliadis and A. Varma, "Efficient fair queuing algorithms for packet switched networks," *IEEE/ACM Transactions on Networking*, vol. 6, no. 2, pp. 175–185, 1998.

[47] ——, "A general methodology for designing efficient traffic scheduling and shaping algorithms," in *INFOCOM*, 1997.

[48] S. Suri, G. Varghese, and G. Chandranmenon, "Leap forward virtual clock: A new fair queuing scheme with guaranteed delays and throughput fairness," in *INFOCOM*, 1997.

[49] M. Karsten, "SI-WF2Q: WF2Q approximation with small constant execution overhead," in *INFOCOM*, 2006.

[50] S. Keshav, "On the efficient implementation of fair queuing," *Journal of Internetworking Research and Experience*, vol. 2, no. 3, pp. 157–173, 1991.

[51] N. Ciulli and S. Giordano, "Analysis and simulation of WF2Q+ based schedulers: comparison, compliance with theoretical bounds and influence on the end-to-end delay jitter," *Computer Networks*, vol. 37, pp. 579–599, 2001.

[52] D. Stevens, J. C. Bennett, and H. Zhang, "Implementing scheduling algorithms in high speed networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 6, pp. 1145–1158, 1999.

[53] J. Xu and R. J. Lipton, "On the fundamental tradeoffs between delay bounds and computational complexity in packet scheduling algorithms," *ACM Computer Communication Review*, vol. 32, no. 4, pp. 279–292, 2002.

[54] D. Knuth, *The art of computer programming.* Reading, MA: Addison Wesley, 1977.

[55] R.Bhagwan and B. Lin, "Fast and scalable priority queue architecture for high-speed networ switches," in *INFOCOM*, 2000.

[56] P. van Emde Boas, R. Kass, and E. Zijlstra, "Design and implementation of an efficient priority queue," *Mathematical Systems Theory*, vol. 10, pp. 99–127, 1977.

[57] P. Valente, "Exact GPS simulation with logarithmic complexity, and its application to an optimally fair scheduler," *ACM Computer Communication Review*, vol. 34, no. 4, pp. 269–280, 2004.

[58] Q. Zhao and J. Xu, "On the computational complexity of maintaining GPS clock in packet scheduling," in *INFOCOM*, 2004.

[59] J. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 675–689, 1997.

[60] Motorola. [Online]. Available: www.motorola.com

[61] CISCO 12000 series routers. [Online]. Available: www.cisco.com

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

[63] S. Kanhere and H. Sethu, "Fair, efficient and low-latency packet scheduling using nested deficit round robin'," in *Proceeding of the IEEE Workshop on High Performance Switching and Routing*, Dallas,Texas, USA, May 2001.

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

[65] S. S. Kanhere and H. Sethu, "Prioritized elastic round robin: An efficient and low-latency packet scheduler with improved fairness," Department of Computer Science, Drexel University, Tech. Rep., 2003.

[66] L. Lenzini, E. Mingozzi, and G. Stea, "Eligibility-based round robin for fair and efficient packet scheduling in wormhole switching networks." *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 3, 2004.

[67] H. Adisshu, G. Parulkar, and G. Varghese, "A reliable and scalable striping protocol," in *Proc. ACM SIGCOMM*, August 1996.

[68] A. Hari, G. Varghese, and G. Parulkar, "An architecture for packet-striping protocols," *ACM Transactions on Computer Systems*, vol. 17, no. 4, November 1999.

[69] A. Kortebi, S. Oueslat, and J. Roberts, "Implicit service differentiation using deficit round robin," in *Proc. ITC19*, Beijing, 2005.

[70] A. Kortebi, L. Muscariello, S. Oueslat, and J. Roberts, "Minimizing the overhead in implementing flow-aware networking," in *Proc. ANCS'05*, Prinston, New Jersey, USA, October 2005.

[71] S. Cheung and C. Pencea, "BSFQ: Bin-sort fair queuing," in *INFOCOM*, 2002.

[72] C. R. Kalmanek, H. Kanakia, and S. Keshav, "Rate controlled servers for very high speed networks," in *Globecom*, 1990.

[73] L. Lenzini, E. Mingozzi, and G. Stea, "Tradeoffs between low complexity, low latency, and fairness with deficit round robin schedulers," *IEEE/ACM Transactions on Networking*, vol. 12, no. 4, August 2004.

[74] S.-C. Tsao and Y.-D. J. Lin, "Pre-order deficit round robin: a new scheduling algorithm for packet-switched networks." *Computer Networks*, vol. 35, no. 2-3, pp. 287–305, 2001.

[75] S. Ramabhadran and J. Pasquale, "The stratified round robin scheduler: design, analysis and implementation." *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, pp. 1362–1373, 2006.

[76] X. Yuan and Z. Duan, "FRR: a proportional and worst-case fair round robin scheduler," in *INFOCOM*, 2005.

[77] B. Caprita, W. C. Chan, J. Nieh, C. Stein, and H. Zheng, "Group ratio round robin: O(1) proportional share scheduling for uniprocessor and multiprocessor systems," in *USENIX Annual Technical Conference*, 2005.

[78] G. Chuanxiong, "SRR: An o(1) time complexity packet scheduler for flows in multi-service packet networks'," in *Proc. SIGCOMM'01*, 2001.

[79] R. Garg, "RRR: Recursive round robin scheduler," in *Globecom*, 1998.

[80] OPNET. [Online]. Available: www.opnet.com

[81] The network simulator ns-2. [Online]. Available: www.isi.edu/nsnam/ns/

[82] OMNET++ simulator. [Online]. Available: www.omnetpp.org

[83] B. Stroustrup, *The C++ programming language.* Reading, MA: Addison-Wesley, 1999.

[84] D. Gross and C. M. Harris, *Fundamentals of Queuing Theory.* New York, NY: John Willey and Sons, Inc, 1998.

[85] P. Nain, "Basic elements of queuing theory," 1998, lecture Notes. [Online]. Available: http://www.s3.kth.se/lcn/courses/2E1624/nain-qt-basics.pdf

[86] J. Beckers, I. Hendrawan, R. Kooij, and R. van der Mei, "Generalized processor sharing models for internet access lines," in *9th IFIP Conference on performance modeling and evaluation of ATM and IP networks*, 2001.

[87] S. Aalto, U. Ayesta, and E. Nyberg-Oksanen, "M/G/1/MLPS compared to M/G/1/PS," *Oper. Res. Lett.*, vol. 33, no. 5, 2005.

[88] A. Parekh, "A generalized processor sharing approach to flow control in integrated service networks," Ph.D. dissertation, Massachusetts Institute of Technology, February 1992.

[89] A. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated service networks: The multiple node case," *IEEE/ACM Transactions on Networking*, vol. 2, no. 2, pp. 137–150, 1994.

[90] D. Stiliadis and A. Varma, "Latency-rate servers: a general model for analysis of traffic scheduling algorithms," *IEEE/ACM Transactions on Networking*, vol. 6, no. 5, pp. 611–624, 1998.

[91] D. Stiliadis, "Traffic scheduling in packet-switched networks: Analysis, design, and implementation," Ph.D. dissertation, University of California at Santa Cruz, USA, June 1996.

[92] S. Kanhere and H. Sethu, "On the latency bound of pre-order deficit round robin'," in *Proceeding of the 27th IEEE Conference on Local Computer Networks (LCN 2002)*, Tampa, Florida, USA, November 2002.

[93] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round robin," in *Proc. SIGCOMM'95*, 1995.

[94] S. Kanhere and H. Sethu, "On the latency bound of deficit round robin'," in *Proc. ICCCN*, Miami, Florida, USA, October 2002.

[95] D. Nikolova, B. Van Houdt, and C. Blondia, "Dynamic bandwidth allocation algorithms in EPON: a simulation study," in *Proc. of OptiComm*, 2003, pp. 369–380.

[96] ——, "QoS issues in EPON," in *Workshop on Community Networks and FTTH/P/x (CNFT)*, 2003.

[97] ——, "Dynamic bandwidth allocation algorithms in Ethernet Passive Optical Networks (EPON) with threshold reporting," *Telecommunication Systems*, vol. 28, no. 1, pp. 30–52, 2005.

[98] ——, "Ethernet Passive Optical Network," University of Antwerp, Tech. Rep., 2003.

[99] *A Broadband optical access system with increased service capabilities using dynamic bandwidth assignment*, ITU-T G 983.4 Std., 2001.

[100] E. Ringoot, N. Janssens, M. Tassent, J. Angeloupoulos, C. Blondia, and P. Vetter, "Demonstration of dynamic medium access control for APON and SuperPON," in *Globecom*, 2001.

[101] B. V. Houdt, C. Blondia, O. Casals, and J. García, "Performance analysis of a MAC protocol for broadband wireless ATM networks with Quality-of-Service provisioning," *Journal of Interconnection Networks (JOIN)*, vol. 2, no. 1, pp. 103–130, 2001.

[102] C. Blondia, "A discrete-time batch markovian arrival process as b-isdn traffic model," *Belgian Journal of Operations Research, Statistics and Computer Science*, vol. 32, no. 3-4, 1993.

[103] National laboratory for applied network research, now CAIDA: The cooperative association for internet data analysis. [Online]. Available: www.nlanr.net, http://www.caida.org/home/

[104] *RFC 793 - Transmission Control Protocol*, Std., September 1981.

[105] R. Braden, *RFC 1122, Requirements for Internet Hosts - Communication Layers*, Std., October 1989.

[106] *Recommendation H.222.0, Information technology - generic coding of moving pictures and associated audio information systems*, ITU-T Std., 2000.

[107] D. Nikolova and S. Van Roempaey, "EuroDOCSIS protocol for hybrid fiber coax (HFC) access network," University of Antwerp, Tech. Rep., 2004.

[108] Telenet. [Online]. Available: www.telenet.be

[109] A. Wierman, "Fairness and classifications," *ACM SIGMETRICS Performance Evaluation Review*, vol. 34, no. 4, March 2007.

[110] Guo and Mata, "The war between mice and elephants," in *Proc. ICNP 2001*, Riverside, CA, November 2001.

[111] I. A. Rai, G. Urvoy-Keller, M. K. Vernon, and E. W. Biersack, "Performance analysis of LAS-based scheduling disciplines in a packet switched network." in *SIGMETRICS*, 2004, pp. 106–117.

[112] K. Thompson, G. Miller, and R. Wilder, "Wide-area internet traffic patterns and characteristics," *IEEE Network*, vol. 1, no. 6, pp. 10–23, 1997.

[113] *RFC 1990 - The PPP Multilink Protocol (MP)*, IETF Std., August 1996.

[114] W. Shi, M. H. MacGregor, and P. Gburzynski, "Load balancing for parallel forwarding," *IEEE/ACM Transactions on Networking*, vol. 13, no. 4, pp. 790–801, 2005.

[115] Ji, Guo, and L. N. Bhuyan, "Load balancing in a cluster-based web server for multimedia applications," *IEEE/ACM Transactions on Parallel and Distributed Systems*, vol. 17, no. 11, november 2006.

[116] W. Shi and L. Kencl, "Sequence-preserving adaptive load balancers," in *Proceedings of the 2006 ACM/IEEE Symposium on Architecture for Networking and Communications Systems, ANCS 2006*, December 2006.

[117] Josep M. Blanquer and Banu Ozden, "Fair queueing for aggregated multiple links," in *Proceedings of the ACM SIGCOMM*, October 2001.

[118] J. Cobb and M. Lin, "A theory of multi-channel schedulers for quality of service," *Journal of High Speed Networks*, vol. 12, no. 12, 2002.

[119] H. Xiao and Y. Jiang, "Analysis of multi-server round robin scheduling disciplines," *IEICE Transactions on Communications*, vol. E87-B, no. 12, December 2004.

[120] Satya R. Mohanty and Laxmi N. Bhuyan, "On fair scheduling in heterogeneous link aggregated services," in *Proceedings of the ICCCN*, 2005.

[121] J. Guo, J. Yao, and L. Bhuyan, "Fair queueing for aggregated multiple links," in *Proceedings of INFOCOM*, 2005.

[122] J. Yao, J. Guo, and L. Bhuyan, "Fair link striping with FIFO delivery on heterogeneous channels," *Computer Communications*, vol. 31, pp. 3427–3437, 2008.

[123] Programming in C, lecture notes by Dave Marshall. [Online]. Available: http://www.cs.cf.ac.uk/Dave/C/

[124] D. Nikolova and C. Blondia, "Evaluation of surplus round robin scheduling algorithm," in *Proc. SPECTS'06*, Calgary, Canada, August 2006.

[125] ——, "Last-backlogged first-served deficit round robin (LBFS-DRR) packet scheduling algorithm," in *Proceedings of the 15th International Conference on Networks (ICON2007)*, November 2007.

[126] ——, "Bonded deficit round-robin: Packet scheduling algorithm for systems with channel bonding," *IEEE/ACM Transactions on Parallel and Distributed Systems*, revised version submitted.