Resource management in in-home digital networks using Dantzig-Wolfe decomposition

Resource Management in In-Home Digital Networks using Dantzig-Wolfe Decomposition

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Boef, Edgar den

Resource management in in-home digital networks using Dantzig-Wolfe decomposition / by Edgar den Boef. - Eindhoven : Technische Universiteit Eindhoven, 2005
Proefschrift. - ISBN 90-386-0524-2
NUR 919
Subject headings : resource management / linear programming / digital communication systems
CR Subject Classification (1998) : G.1.6, C.2.3

Cover design by Jan-Willem Luiten

Resource Management in

In-Home Digital Networks

using Dantzig-Wolfe Decomposition

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de Rector Magnificus, prof.dr.ir. C.J. van Duijn, voor een commissie aangewezen door het College voor Promoties in het openbaar te verdedigen op donderdag 28 april 2005 om 16.00 uur

door

Edgar den Boef

geboren te Tilburg

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr. E.H.L. Aarts
en
prof.dr.ir. D. den Hertog

Copromotor:
dr.ir. W.F.J. Verhaegh

Preface

Five years ago, when I was finishing my Master's Thesis, I was wondering what the next step in my career would be. Until that time I was hesitant about pursuing a Ph.D., but a discussion with Dick den Hertog, professor at Tilburg University, changed my mind. The search for an interesting topic to spend the next four years on, led me to Emile Aarts, who gave me the opportunity to perform the research for this thesis at Philips Research Laboratories. Although these years as a Ph.D. student have not always been easy, I have never regretted the choice I made. The final result, that you are now holding, has been made possible thanks to the support of many people.

First of all, I would like to thank Wim Verhaegh and Jan Korst, my daily supervisors at Philips Research. The pleasant discussions I had with them, gave me enough ideas to continue my research. Furthermore, their comments on my manuscripts and this thesis really helped me with improving them. Next, I would like to thank Emile Aarts for giving me this opportunity and for his helpful advice on both thesis and non-thesis related issues during the past years. I would like to thank Dick den Hertog for convincing me to work towards a Ph.D., and for our fruitful co-operation on line-searching for convex functions, that resulted in a joint paper. I also owe thanks to Clemens Wüst for his help on deriving the video traces that I required for the experiments in this thesis, and for his help on several implementation issues that I encountered.

A pleasant working environment is an important ingredient for a good thesis. This was provided to me at both the (Eindhoven) Embedded Systems Institute at the Technische Universiteit Eindhoven and the Media Interaction group of Philips Research Laboratories. For this, I would like to thank all my colleagues from both places and especially Ingrid for her interest and support on our Tuesdays at the university, and Marc, Stieneke, Johan, Evelien, Zharko, Gijs, and all other roommates I had during these years at Philips for their interest and the pleasant distractions away from work. Furthermore, I would like to thank everyone from the 'Mafia-gang' at the MAPSP-workshops for the enjoyable evenings I had playing this wonderful game after a day full of presentations.

Finally, I would like to thank my family and friends for their continuous interest and support. I owe special thanks to Frank and Arno for their friendship over the

years and for assisting me during the defense of my thesis. But above all, I thank my parents for their support and for providing me with a warm home to return to after a day's work.

Eindhoven, February 2005 Edgar den Boef

Contents

1

Introduction

1.1 In-home digital network

About twenty years ago digital devices such as the personal computer (PC) and the compact disk (CD) player started to make their appearance in the home. Nowadays, most people have several PCs and CD players in their homes together with new digital devices such as a DVD player and in the near future a high-definition television (HDTV) set. Some of these devices have similar functionality, e.g., a DVD can be played with a stand-alone DVD player, but also with a DVD player built in a PC. If all digital devices in the home are interconnected by a so-called in-home digital network (IHDN), functionality can be shared and new and exciting features and applications will become possible. For example, you could watch a video program on any (TV) screen in the home without the need to know the exact place where the program is stored, or you could surf the Internet in any desired room, or you could listen to your favorite music that moves along with you when you walk through your home. The communication between the devices could take place using existing wiring, e.g. power or phone lines, or new wired and wireless connections could be installed.

The new applications using the IHDN give rise to a number of resource management problems. Resources such as processors and storage devices may be used by more than one application, A suitable allocation of these resources needs to be determined such that the involved applications possess the quality level that

is required or that is the best possible. Furthermore, the communication of each application using the IHDN needs to be handled efficiently. When several people at the same time watch a video program or surf the Internet, then multiple data streams have to be transmitted simultaneously over the network, and consequently the bandwidth on one or more connections has to be shared. To prevent data congestion and resulting data loss, data can be prefetched and buffered, thereby smoothing the data supply to the communication links during times with peak traffic in the network.

An issue related to resource management is admission control, i.e., whether an application can be allowed on the network and use the network's resources. Once a stream is admitted to the network, it should be guaranteed that it can use the resources that have been assigned to it. This decision can be made by trying to make suitable resource reservations for a new application, and allowing the application on the network only if a reservation of the required resources is possible. The feasibility of such a reservation depends on how much of each resource the application requires and also on the resource reservations made by previous applications still using them.

In this thesis we consider a problem concerning the transmission of data for applications using the network. The solution approach that we describe, determines resource reservations for the data transmission, which can be used to decide whether or not the data stream of an application can be allowed on the network, as mentioned above. We will mainly focus on video streams as these generally have a high bit rate and thus require substantial resource reservations. Audio and other data streams mostly can be handled in the margin of the network capacity.

1.2 Video streams

Video streams are usually compressed before they are transmitted over a network, otherwise they would require too much bandwidth. An example of video compression is MPEG-2 encoding; for an extensive description of MPEG encoding we refer to Mitchell et al. [1997] and more specifically for MPEG-2 encoding to Haskell et al. [1997]. Generally, when compressing a video stream there are two possibilities concerning its bit rate and quality. A video stream can be variable-bit-rate (VBR) encoded with a constant perceived quality or it can be encoded with a constant bit rate (CBR), leading to a variable quality. To obtain a constant quality, more bits are spent on relatively difficult scenes and fewer bits on relatively easy ones. However, this may lead to significant variability in bit rate; the peak rate can be up to five or ten times the average rate. A clear advantage of a CBR stream is the constant requirement it imposes on the bandwidth. However, in order to deliver the same perceived video quality to the user, CBR encoding of a video stream leads to

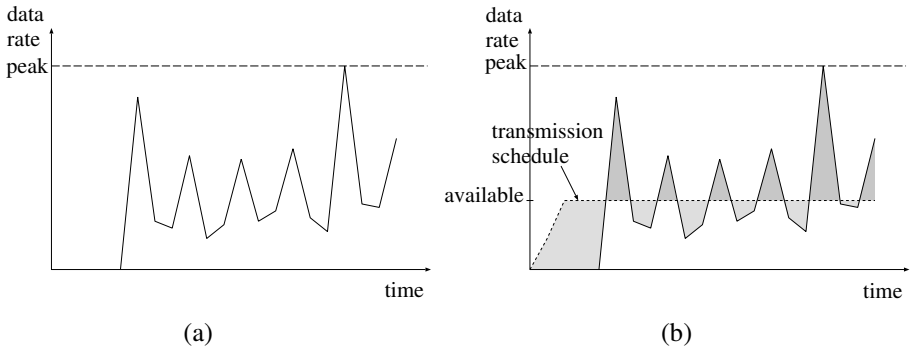

Figure 1.1. (a) Example of a VBR stream. If bandwidth is reserved at peak rate then a large part of the bandwidth will remain unused. (b) However, suppose the available rate for transmission is set much lower than the peak rate. Data that exceeds the available rate (given by the dark gray areas) then needs to be transmitted early (given by the light gray areas) and buffered until it is required. This results in a smoother transmission schedule.

a significantly higher bit rate than the average rate of the corresponding VBR encoding [Dalgic & Tobagi, 1996]. As a reservation of bandwidth for a stream does not necessarily have to be at the stream's peak rate as we will show later, VBR encoding generally enables more streams to use a communication link than CBR encoding. Even when using CBR reservations, VBR encoding is more efficient. Therefore, we only consider VBR streams in this thesis.

MPEG-encoded streams also exhibit small-time-scale variability due to their encoding in three different types of frames. These are I-frames, P-frames, and B-frames. Generally I-frames are larger than P-frames and P-frames are larger than B-frames. The I-, P-, and B-frames are grouped into so-called Groups-Of-Pictures (GOPs). A GOP starts with an I-frame and contains after this I-frame a number of P- and B-frames. A sequence of frames in a GOP could be given by IBBPBBPBBPBB or IBPBBBPBBPBPBB.

Thus, the rate at which a VBR stream transmits data varies heavily over time. If everything is transmitted just-in-time, a bandwidth reservation based on a stream's peak rate should be made. However, this leads to an unnecessarily high reservation and low utilization of the bandwidth. By buffering data before and after transmission, the bandwidth reservation can be decreased and a smoother transmission schedule is obtained for a VBR stream; see Figure 1.1. Algorithms that determine such transmission schedules are often referred to as *bandwidth smoothing* algorithms.

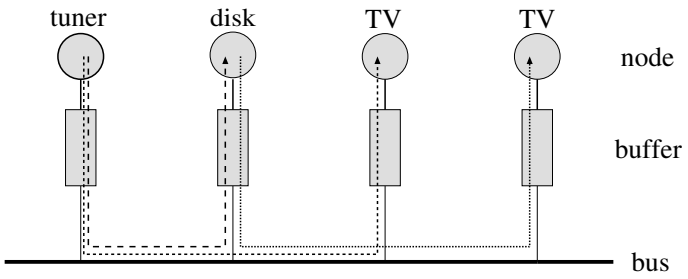

Figure 1.2. An example network with four nodes and three streams running.

1.3 Informal problem statement

In this thesis we concentrate on the following problem. Let an IHDN consist of a single bus with finite capacity, i.e., the bandwidth of the bus, to which several nodes are connected. These nodes may represent, e.g. a set-top box, a storage device, a multimedia PC, or a TV set. Between each node and the bus there is a finite buffer. Furthermore, there is a set of data streams, with each stream running from one node to another node. Figure 1.2 shows an example network. The streams have a variable bit rate. Furthermore, we distinguish two types of streams, namely prerecorded streams and live streams. For a prerecorded stream the exact supply and demand of data is given, while for a live stream only an upper bound on the supply and demand is given. More specifically we consider for a live stream the case in which a stream is controlled by one or more so-called leaky buckets [Turner, 1986] which shape the stream before it enters the network. In the remainder of this thesis we refer to prerecorded streams as fully-specified streams and to live streams as leaky-bucket-controlled streams.

The problem that we consider is to determine for each stream its reserved share of the bus capacity (bandwidth) and its reserved share of the corresponding buffers, as well as a transmission strategy indicating how the data should be transmitted over time, such that the total bus and buffer capacities are not exceeded and buffers neither underflow nor overflow. One of the advantages of allocating fixed shares to the streams is the possibility to treat the streams almost independently. Furthermore, with fixed shares of the bandwidth for all streams it requires less administrative work to give each stream a guaranteed part of the bandwidth to use than with continuously changing shares. Finally, bandwidth that is not used by a stream is not necessarily lost. It can be used by best-effort streams in the network, e.g., data that is copied from one location in the network to another location. The solution of the above described resource management problem can be used to decide whether for a given set of streams all streams can be admitted to the network. The required calculations can be done by a central control node, having enough computational

power, but also may be performed in a more distributed manner.

We consider both an off-line and an on-line setting of the above problem. In the off-line setting we assume that all streams are known before their transmission is to be scheduled. In the on-line setting the starting time of a stream is unknown. For the off-line setting we assume the reserved shares are fixed for the whole runtime of a stream. However, for the on-line setting we allow that each time a stream starts, the shares of the already running streams can be adjusted when determining bandwidth and buffer shares for the new stream.

1.4 Solution approach

In this thesis we will describe a method to tackle the above described problem for both the off-line and the on-line setting. As mentioned, we consider two different types of streams, namely fully-specified streams for which data supply and demand is exactly known in advance, and leaky-bucket-controlled streams. Ideally, the approach that is used is able to handle both types of streams. For both types it has to be determined how large the shares of the bandwidth and relevant buffers should be for a stream. A suitable approach divides the bandwidth and buffer capacities over the streams in the same way for both types, making it easy to solve problem instances in which both types occur. The required sizes of the bandwidth and buffer shares as well as the transmission strategy of a stream can then be determined independently for each stream. In this way, the total problem is decomposed into two stages: the allocation of the resources over the different streams and the determination of a transmission strategy for each stream.

The solution method that we propose and that meets the above requirement, is based on the Dantzig-Wolfe decomposition for linear programming (LP) models. It is specifically suitable for LP models in which a large number of constraints only involve limited sets of variables. It exploits this structure by decomposing the LP problem into a master problem and a number of sub-problems. In this way the run time can be decreased considerably. Furthermore, it allows for the use of efficient algorithms specific for the sub-problems, as we will show in this thesis.

Besides being able to handle both types of streams, the solution approach can use an existing solution for current streams on the network, when a new stream requests to be allocated bandwidth and buffer shares. This saves time calculating solutions for current streams, that simultaneously can more easily maintain their solutions. Finally, the solution approach is suitable to be used as a diagnostic, i.e., when no solution exists for the given set of streams and capacities, it can indicate for which additional resource requirement a feasible solution does exist. This can be used to easily identify bottlenecks and to give a cost to the infeasible solution.

1.5 Related work

The area of in-home digital networks is fairly new with publications in this field just emerging in recent years. One of the oldest publications on in-home digital networking is by Chen [1997] who identified emerging needs for a digital home network, such as home automation, home computers, digital video and audio equipment, and digital access systems, and proposed a distributed home network architecture.

More recently, some journals came with special issues dedicated to in-home networks. IEEE Communications Magazine reserved parts of its April issues in 2002 and 2003 for in-home networking. Topics of discussion in these issues are among other things the perspective of the end-user on home networking, a service delivery solution, and service gateway architectures for the home. The July/August 2003 issue of Telecommunication Systems was devoted to multimedia home telecommunication systems. Besides a paper comprising a part of Chapter 2 of this thesis, it contains papers concerning a management scheme to provide differentiated services, a wireless LAN protocol which aims to provide QoS over in-home wireless networks, and residential gateway architectures.

Some other publications concerning IHDNs are by Scholten et al. [2002], Giovanelli et al. [2003], and Salazar [2003]. Scholten et al. describe an in-home digital network architecture that deploys a distributed token mechanism for communication. The architecture supports both real-time and non-real-time communication and guaranteed QoS is offered. Giovanelli et al. describe a bandwidth reservation scheme for UPnP-based in-home digital networks. The main difference between their and our approach is that they assume that all devices in the IHDN support the UPnP protocol, while we describe an approach independent of the used protocols. Finally, Salazar presents a mathematical model for an IHDN, which is used together with queuing analysis to evaluate network performance in terms of mean response time. Unlike the probabilistic approach that is used in this paper, we only focus on deterministic methods and results in this thesis.

Publications concerning the transmission of VBR video are more abundant. On the topic of bandwidth smoothing algorithms for VBR video several publications have appeared. Most of them present an algorithm that smoothes a single stream that has to be transmitted from a server to a client with limited buffer size. Salehi et al. [1998] present a smoothing algorithm that minimizes the peak rate and the variance of the transmission schedule for a given buffer size. Sanjay & Raghavan [1999] show how this schedule can be determined quickly for different buffer sizes. Feng [1997] presents an algorithm that minimizes the buffer requirements for a given bandwidth. Feng et al. [1997] and Zhang & Hui [1998] both present an algorithm that leads to a transmission schedule with the smallest number of band-

width increases, the smallest peak bandwidth, and the largest minimum bandwidth requirement. Furthermore, Feng et al. [1997] also give an algorithm to obtain a schedule without bandwidth increases, assuming there is no buffer size restriction. Chang et al. [1997], Kang & Yeom [1999], and Zhang & Hui [1997] all present algorithms to obtain so-called on-off schedules. These schedules alternate between transmitting at peak rate (on) and not transmitting at all (off). Chang et al. [2002] also distinguish so-called lazy and aggressive on-off-transmission schemes which they use to derive ready times and deadlines for the transmission of data packets. McManus & Ross [1998] present a dynamic programming method to determine a transmission schedule. They give several possible objectives for which this method can be used. Jiang & Kleinrock [1998] present an optimal smoothing algorithm assuming there is a small number of possible transmission rates for each time unit. Zhang & Fu [2000] consider the transmission of a VBR stream which is divided into a number of segments. They show how to determine the required bandwidth and buffer size for each of the segments. Finally, Feng & Rexford [1999] give an overview of various smoothing algorithms and compare them using several criteria.

Rexford & Towsley [1999] consider the transmission of VBR video in an internetworking environment, where a service provider does not control the whole transmission path of the video. They present efficient algorithms for transmitting VBR video across a portion of the route, from an incoming node to an outgoing node. They show how to minimize both the total buffer allocation at these nodes and the start-up delay. They also describe how to compute a transmission schedule that minimizes buffer allocation for a sequence of nodes. Kang & Yeom [2000] consider the transmission of multiple streams to a single client with a buffer. They show how the network bandwidth utilization can be optimized by first exploiting the statistical multiplexing gain through aggregation of the streams, and then smoothing the aggregate.

Chang et al. [1998] consider the transmission of a prerecorded VBR stream for which both the bandwidth and the receiving buffer size need to be minimized. This problem is considered more extensively in Section 2.2.4 as one of the subproblems for a fully-specified stream. To determine the set of solutions that minimize both the bandwidth and the receiving buffer size, they construct a bandwidth-buffer trade-off curve. Using this algorithm they also show how to make an optimal trade-off between bandwidth and buffer size if a VBR stream needs to be transmitted over multiple relay-servers [Chang et al., 1999b]. Zhao et al. [1998] show how to perform an optimal trade-off between the bandwidth and the initial delay between the start of transmission and the start of video playback.

Besides smoothing video streams by sending data in advance as is done by the above approaches, it also possible to smooth streams by adjusting their video quality, using the fact that small quality differences are hardly noticeable to the user.

Duffield et al. [1998] give an algorithm that crops the frame sizes during encoding when the available bandwidth is not enough to transmit the ideal size of the frame. Ng [1999] presents an algorithm that deletes B-frames to obtain a smooth schedule. Another way to decrease the bandwidth requirements of a VBR stream at the cost of quality loss involves so-called scalable video encoding. Van der Schaar-Mitrea [2001] describes a simple video-coding structure that consists of a non-scalable base-layer and a scalable enhancement layer.

The previous-mentioned works concerning bandwidth smoothing, all considered VBR streams of which the amount of data to be transmitted is known in advance. However, in an on-line setting only limited knowledge of the data supply of a stream is available, which requires a different approach. Rexford et al. [1997] developed the first models and algorithms for on-line smoothing. These algorithms are based on time windows of limited size, containing a number of consecutive frames. At the beginning of each window an off-line smoothing method is used to determine the required bandwidth for the stream during that window. They consider two types of algorithms using these windows, viz. hopping-window algorithms, which operate on consecutive, non-overlapping windows, and sliding-window algorithms, which execute the smoothing on overlapping windows. Chang [2000b] extends on the sliding-window algorithms by describing a scheme that dynamically adjusts the sliding distance of the window. He shows that this approach results in a peak bandwidth of at most that of the sliding-window algorithm with sliding distance one, while the computation cost is the same as that of the fastest sliding-window algorithm, namely the algorithm with sliding distance equal to the window size. Chang et al. [1999a] compare these two sliding-window algorithms to the new dynamic approach using some experimental results. Finally, Cao et al. [2003] formulate an on-line smoothing model and propose a benchmark on-line smoothing algorithm based on this model. They also describe some new on-line smoothing heuristics, which are evaluated using their benchmark algorithm.

Whereas in most of the above-mentioned literature data transmission to just one receiver with a buffer is considered, we consider multiple streams. Each stream may originate at one of several possible senders and needs to be transmitted to one of several possible receivers, where each sender and receiver has its own buffer. Furthermore, we show how single-stream smoothing algorithms can be used to determine bandwidth and buffer shares for each stream such that a maximum number of streams can be serviced.

We conclude this section with some references concerning leaky-bucket-controlled streams. For leaky-bucket-controlled streams described by the (σ, ρ)-characterization an extensive calculus has been developed, originally by Cruz [1991a, 1991b], which was later extended by several other authors. For a clear description and overview of this calculus we refer to the book by Chang [2000a].

	Chapter 2	Chapter 3
Streams	Fully-specified	Leaky-bucket-controlled
Input	Given supply and demand schemes or controllable supply or demand	Upper bound on supply and demand
Multiple streams	2.1	3.2
Sub-problems	2.2	3.3
Bandwidth min.	2.2.1	3.3.2
Buffer min.	2.2.2	3.3.2
Two-buffer min.	2.2.3	3.3.3
Trade-off	2.2.4 & 2.2.5	3.3.4
Results	2.3	3.4
On-line settings	Chapter 4	
Input	Supply and demand schemes unknown until stream starts	Upper bound unknown until stream starts
Multiple streams	4.1	
Sub-problems	4.2.1	4.2.2
Transmission strategy	4.2.3	
Results	4.3	

Figure 1.3. Structure of the thesis.

The developed calculus can be used to determine resource allocations for streams that are controlled by (σ, ρ)-leaky buckets. However, we could not use it to model the situation in which data needs to be buffered at the receiving side, if it arrives before it is demanded.

1.6 Thesis overview

The structure of this thesis is as follows; see also Figure 1.3 for an overview. First, we formally define the problem for fully-specified streams in Chapter 2. We also present an LP formulation for the problem and we describe how the Dantzig-Wolfe decomposition can be applied. This decomposition leads to a sub-problem per stream in which a weighted sum of the bandwidth and buffer shares needs to be minimized. Depending on the sign of the cost coefficients, different methods are applied to solve the sub-problems. We describe for a fully-specified stream how to minimize the bandwidth share, a single buffer share, and two buffer shares jointly. Furthermore, we describe a specific method to perform a trade-off between the bandwidth and one buffer. For a trade-off between the bandwidth and both buffers we show how to reformulate the problem into finding the minimum of a piecewise-linear, convex function. We show how this minimum can be found efficiently using new line-search methods that exploit the convexity and piecewise linearity of the function. We conclude the chapter with experimental results.

In Chapter 3 we concentrate on leaky-bucket-controlled streams. First, we describe a (σ, ρ)-leaky-bucket controller. Then, we define the problem for leaky-bucket-controlled streams and show how it can be reduced to the problem for fully-specified streams. We continue with describing how to efficiently solve the sub-problem for a leaky-bucket-controlled stream. For this, we derive new constraints for the sub-problem and show that they are sufficient. The methods that we describe to solve the different cases of the sub-problem all use these new constraints. We also conclude this chapter with experimental results.

In Chapter 4 we introduce the on-line settings of the problems for fully-specified and leaky-bucket-controlled streams. We describe how to solve the on-line settings using the Dantzig-Wolfe decomposition approach for the off-line settings. Furthermore, we introduce some objectives that can be used for the LP model. Next, we show how to adapt the methods for the sub-problems of the off-line settings to the on-line settings. We also discuss a strategy concerning transmission and buffer usage to obtain better results in an on-line setting. We conclude again with experimental results.

Finally, we summarize our results and state the conclusions of this thesis in Chapter 5.

2

Fully-specified streams

In this chapter we focus on the problem with fully-specified streams, i.e., we consider the problem of determining a fixed share of the bandwidth and fixed shares of the relevant buffers for streams for which exact supply and demand is given in advance. Besides given supply and demand we also consider the possibility that either supply or demand is not given but can be chosen freely while only constrained by a maximum rate, e.g., when a stream is read from or written to a disk. When watching a video stream, a user may at one time undertake an action such as pausing, rewinding or fast forwarding the playback. Of these actions our model can cover pausing a video stream as it effectively leads to a demand of zero during a certain time period. However, we leave actions as rewinding and fast forwarding for future considerations. The solution methods for the fully-specified-stream problem that we present in this chapter, can serve as a basis for solution methods for a model including actions as rewinding and fast-forwarding as well as for other related problems such as the problem with leaky-bucket-controlled streams and on-line variants of the problem. Furthermore, the problem can also be considered in the context of analyzing the effectiveness of algorithms that address an on-line setting.

The structure of this chapter is as follows. We start with a formal description of the problem concerning fully-specified streams in Section 2.1. This section also contains an LP model for the problem and a description of the Dantzig-Wolfe

decomposition that we apply. In Section 2.2 we describe the solution methods for the single-stream problems that result from the Dantzig-Wolfe decomposition. Finally, we present experimental results in Section 2.3.

2.1 Problem

Before we formally define the problem, we first give our assumptions in Section 2.1.1. Next, we introduce the notation used in this thesis for the fully-specified-streams problem in Section 2.1.2. In Section 2.1.3 we define the problem which concerns fully-specified streams as the Multiple Streams Smoothing Problem and give the constraints which have to be satisfied. Then we show in Section 2.1.4 how it can be modelled as a linear program. Finally, in Section 2.1.5 we show how to apply the Dantzig-Wolfe decomposition to the resulting LP.

2.1.1 Model assumptions

The first assumption is that we split up the time axis into a finite set of identical time units. A time unit can be chosen long, e.g. a minute, or short, e.g. 1/25 s, depending on the application of the model. We assume that during a time unit, data of more than one stream can be transmitted over the bus, and that the switching time between transmissions for different streams is either negligible or beforehand subtracted from the available bandwidth. Furthermore, we assume that there is no loss of data during transmission.

We next assume that an application supplies data of a stream at the sending node. For a fully-specified stream, the supply is mostly given by a supply scheme, e.g. a video stream that comes from a tuner and is pushed into the network. However, sometimes the supply is controllable, e.g. a stream that is read from a hard disk, for which the amount of data submitted into the network can be controlled. In this case, the data can bypass the buffer, so it is transferred immediately from the node to the bus. The only constraint is a maximum rate at which the sending node can supply data to the bus. The demand of data takes place at the receiving node by an application. Just as supply, demand of a fully-specified stream can be either given by a demand scheme, or controllable with a maximum demand rate. We assume that for each fully-specified stream either supply or demand or both are given.

With respect to buffering, we take worst-case assumptions, i.e., we assume that data is supplied at the beginning of a time unit, and then buffered at the sending node before it is transmitted. After transmission somewhere within the time unit, it is buffered at the receiving node before it is demanded at the end of the time unit; cf. Figure 2.1. However, other buffering assumptions can be used as well with the solution method we present in this thesis; they will only lead to marginally

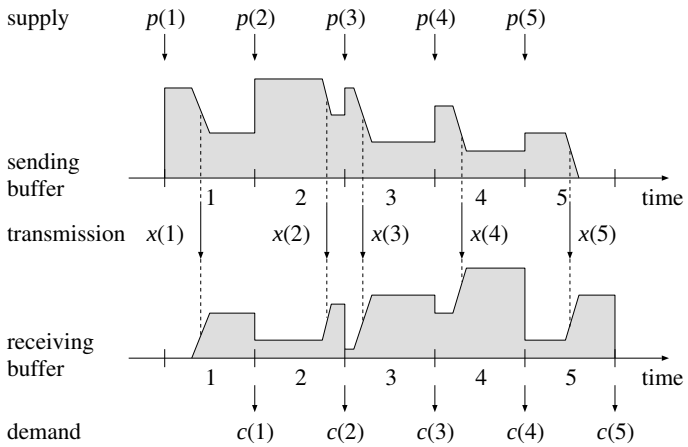

Figure 2.1. The impact of data arrival and departure on the buffer fillings. We assume supply takes place at the beginning of a time unit, demand at the end, and transmission over the bus somewhere in-between.

different constraints for the transmission schedule of a stream. We only take into consideration the total amount of data that is stored in a buffer for each stream; we do not consider how data is exactly stored into memory locations.

2.1.2 Notation

Let $\mathcal{T} = \{1, 2, \ldots, T\}$ denote the set of time units. The moments in time between two consecutive time units are referred to as *time points*. We denote the set of time points by $\mathcal{T}' = \{0, 1, 2, \ldots, T\}$. Then, a time unit $t \in \mathcal{T}$ is equivalent to the interval bounded by the time points $t-1$ and t, i.e., $(t-1, t]$. Let B represent the maximum amount of data that can be transmitted over the bus during a time unit. Furthermore, let $\mathcal{N}$ denote the set of nodes that are connected to the bus. The capacity of the buffer between node $n \in \mathcal{N}$ and the bus is given by M_n. We denote the set of data streams by $\mathcal{D}$ where each stream $d \in \mathcal{D}$ transmits data from a sending node $s_d \in \mathcal{N}$ to a receiving node $r_d \in \mathcal{N}$.

The sets $\mathcal{D}^{\text{sup}} \subseteq \mathcal{D}$ and $\mathcal{D}^{\text{dem}} \subseteq \mathcal{D}$ consist of the fully-specified streams with given supply and demand schemes, respectively. As assumed, $\mathcal{D}^{\text{sup}} \cup \mathcal{D}^{\text{dem}} = \mathcal{D}$, however in general $\mathcal{D}^{\text{sup}} \cap \mathcal{D}^{\text{dem}} \neq \emptyset$. For each stream $d \in \mathcal{D}^{\text{sup}}$ the supply scheme is for all $t \in \mathcal{T}$ given by the amount $p_d(t)$ of data that is supplied at the start of time unit t; see also Figure 2.1. For each stream $d \in \mathcal{D}^{\text{dem}}$ the demand scheme is for all $t \in \mathcal{T}$ given by the amount $c_d(t)$ of data that is demanded at the end of time unit t. The maximum supply rate of a stream $d \in \mathcal{D} \backslash \mathcal{D}^{\text{sup}}$ with controllable supply is given by $P_d^{\max}$; the maximum demand rate of a stream $d \in \mathcal{D} \backslash \mathcal{D}^{\text{dem}}$ with controllable

demand is given by $C_d^{\max}$. A possible delay between supply and demand of a fully-specified stream is implicit in $p_d(t)$ and $c_d(t)$, e.g. if demand equals supply with delay δ, then $c_d(t) = p_d(t-\delta)$.

For each stream $d \in \mathcal{D}$ the decision variables are given by the reserved bus bandwidth b_d, the reserved buffer sizes $m_{s_d,d}$ and $m_{r_d,d}$ at the sending node and at the receiving node, respectively, and the transmission schedule $x_d(t)$, $t \in \mathcal{T}$.

For the remainder of this thesis we define for each $d \in \mathcal{D}$ and for all $t \in \mathcal{T}$

$$\begin{aligned} C_d(t) &= \sum_{k=1}^{t} c_d(k), \\ P_d(t) &= \sum_{k=1}^{t} p_d(k), \\ X_d(t) &= \sum_{k=1}^{t} x_d(k), \end{aligned}$$

as the cumulative demand, supply, and transmission schedules, respectively. For convenience we define $P_d(0) = C_d(0) = X_d(0) = 0$.

2.1.3 Multiple Streams Smoothing Problem

We can now define the following problem for a set of fully-specified streams.

Definition 2.1. Multiple Streams Smoothing Problem (MSSP). Given are a set $\mathcal{N}$ of nodes, with for each node $n \in \mathcal{N}$ a buffer capacity M_n, a set $\mathcal{T}$ of time units, and a bandwidth B. Furthermore, a set $\mathcal{D}$ of streams is given with for each stream $d \in \mathcal{D}$ a sending node $s_d \in \mathcal{N}$ with either a supply scheme $p_d(t)$, $t \in \mathcal{T}$ or a maximum supply rate $P_d^{\max}$, and a receiving node $r_d \in \mathcal{N}$ with either a demand scheme $c_d(t)$, $t \in \mathcal{T}$ or a maximum demand rate $C_d^{\max}$.

Determine for all streams $d \in \mathcal{D}$ values for the bandwidth share $b_d \geq 0$ and buffer shares $m_{s_d,d}, m_{r_d,d} \geq 0$, and a transmission schedule $x_d(t)$ for all $t \in \mathcal{T}$, such that the bandwidth B and buffer capacities M_n are not exceeded and buffer shares neither overflow nor underflow. □

Next, we describe the constraints which the decision variables of MSSP must satisfy. As MSSP is a feasibility problem, i.e., the question is whether a solution exists for the given input, we initially give only the constraints that describe the feasible area.

MSSP concerns allocating fixed shares of the bus and buffer capacities to the different streams, such that feasible transmission schedules for all the streams exist. For this, the following constraints have to be met. First of all, the available bandwidth and buffer capacities may not be exceeded. So, for the bandwidth share

reservations

$$\sum_{d \in \mathcal{D}} b_d \leq B \tag{2.1}$$

must hold, and for each buffer, i.e.,

$$\sum_{d \in \mathcal{D} \mid s_d = n} m_{n,d} + \sum_{d \in \mathcal{D} \mid r_d = n} m_{n,d} \leq M_n \,, \qquad \forall\, n \in \mathcal{N} \,, \tag{2.2}$$

must hold.

The next constraints concern the amount of data transmitted during each time unit. The transmission schedule of a stream should obey the reserved bandwidth and buffer shares, and it should meet the supply and demand schemes. Thus, the amount of data transmitted for stream d during time unit t may not exceed the reserved bandwidth share of stream d, i.e.,

$$x_d(t) \leq b_d \,, \qquad \forall\, d \in \mathcal{D} \,,\, t \in \mathcal{T} \,. \tag{2.3}$$

The following constraints regard buffer overflow and underflow; cf. Figure 2.1. The reserved buffer space at the sending node may not underflow, i.e.,

$$P_d(t) - X_d(t) \geq 0 \,, \qquad \forall\, d \in \mathcal{D}^{\text{sup}} \,,\, t \in \mathcal{T} \,. \tag{2.4}$$

The reserved buffer space at the sending node may not be exceeded, i.e.,

$$P_d(t) - X_d(t-1) \leq m_{s_d,d} \,, \qquad \forall\, d \in \mathcal{D}^{\text{sup}} \,,\, t \in \mathcal{T} \,. \tag{2.5}$$

The reserved buffer space at the receiving node may not underflow, i.e.,

$$X_d(t) - C_d(t) \geq 0 \,, \qquad \forall\, d \in \mathcal{D}^{\text{dem}} \,,\, t \in \mathcal{T} \,. \tag{2.6}$$

The reserved buffer space at the receiving node may not be exceeded, i.e.,

$$X_d(t) - C_d(t-1) \leq m_{r_d,d} \,, \qquad \forall\, d \in \mathcal{D}^{\text{dem}} \,,\, t \in \mathcal{T} \,. \tag{2.7}$$

If the supply of a fully-specified stream is controllable, then each $x_d(t)$ may not exceed the amount of data that can be supplied during a time unit. This can be accomplished in combination with (2.3) by demanding that the reserved bandwidth does not exceed the amount of data that can be supplied during a time unit. Furthermore, the reserved buffer size at the sending node can be set to 0, i.e.,

$$b_d \leq P_d^{\max} \,\wedge\, m_{s_d,d} = 0 \,, \qquad \forall\, d \in \mathcal{D} \backslash \mathcal{D}^{\text{sup}} \,. \tag{2.8}$$

If the demand is controllable, then analogously,

$$b_d \leq C_d^{\max} \,\wedge\, m_{r_d,d} = 0 \,, \qquad \forall\, d \in \mathcal{D} \backslash \mathcal{D}^{\text{dem}} \,. \tag{2.9}$$

This concludes the description of MSSP. In Section 2.1.4 we show how to model this problem as an LP problem. In Section 2.1.5 we describe a solution method by applying the Dantzig-Wolfe decomposition to the LP problem.

2.1.4 LP model

Although fractional values are not allowed for the decision variables in MSSP, it can nevertheless be formulated as an LP, and solved using LP methods. Any fractional values that are obtained for the variables can be rounded off without really affecting the solution. The reason for this is that everything is expressed in bytes, giving very large values for the constants and decision variables (in the order of 10,000 or more), so rounding off a variable has a negligible effect.

We want to know whether there is a feasible solution to (2.1), (2.2), and (2.3)–(2.9) for all $d \in \mathcal{D}$. We translate this set of constraints into an optimization program which always has a feasible solution, by adding slack and penalty variables to (2.1) and (2.2), resulting in revised constraints. This gives for the reserved bandwidth shares

$$\sum_{d \in \mathcal{D}} b_d + s = B + p \,, \tag{2.10}$$

with slack variable $s \geq 0$ and penalty variable $p \geq 0$. Furthermore, we have for the reserved buffer shares

$$\sum_{d \in \mathcal{D} \mid s_d = n} m_{n,d} + \sum_{d \in \mathcal{D} \mid r_d = n} m_{n,d} + s_n = M_n + p_n \,, \qquad \forall\, n \in \mathcal{N} \,, \tag{2.11}$$

with slack variables $s_n \geq 0$ and penalty variables $p_n \geq 0$. The objective becomes to minimize

$$p + \sum_{n \in \mathcal{N}} p_n \,.$$

This objective, together with (2.10), (2.11), and (2.3)–(2.9) for all $d \in \mathcal{D}$ forms an LP that determines whether a feasible solution for MSSP exists. If a solution is found with an objective value that equals zero, it is also a feasible solution for MSSP. Otherwise, the values of the penalty variables indicate how much the bandwidth and buffer capacities should be increased before the solution is feasible for MSSP.

2.1.5 Dantzig-Wolfe decomposition

To solve the above LP problem efficiently, we use the fact that (2.3)–(2.9) can be considered per individual stream. This makes this LP problem suitable to apply a Dantzig-Wolfe decomposition. We assume that the reader has some familiarity with LP models and the simplex algorithm. For the linear programming theory we refer to Papadimitriou & Steiglitz [1982]. For a general description of the Dantzig-Wolfe decomposition we refer to Dantzig & Wolfe [1960, 1961], and again to Papadimitriou & Steiglitz [1982]. Ho & Loute [1981] describe a manner for the implementation of the Dantzig-Wolfe decomposition algorithm.

First, we divide the constraints of the LP problem into the following two sets.

- The central constraints consisting of (2.10) and (2.11).
- The stream constraints, one set per stream $d \in \mathcal{D}$, consisting of (2.3)–(2.9) for all $d \in \mathcal{D}$, and the following constraints (2.12), (2.13), and (2.14) to ensure that each set of stream constraints describes a bounded polyhedron:

$$b_d \leq B\,, \tag{2.12}$$
$$m_{s_d,d} \leq M_{s_d}\,, \tag{2.13}$$
$$m_{r_d,d} \leq M_{r_d}\,. \tag{2.14}$$

  Although Dantzig-Wolfe decomposition can also be applied when the stream constraints describe a polyhedral cone, we avoid unnecessary complexity with the use of (2.12)–(2.14).

Next, we define for each stream d a stream solution set $\mathcal{P}_d$ containing solutions $(b_d, m_{s_d,d}, m_{r_d,d}, x_d(0), \ldots, x_d(T-1))$ that meet the corresponding stream constraints. As all these constraints are linear, we can express each solution in a stream solution set as a convex combination of the extreme points of the set, i.e., if the stream solution set $\mathcal{P}_d$ has k_d extreme points $\boldsymbol{y}_{d,1}, ..., \boldsymbol{y}_{d,k_d}$, then we can write any solution as

$$(b_d, m_{s_d,d}, m_{r_d,d}, x_d(0), \ldots, x_d(T-1)) = \lambda_{d,1}\boldsymbol{y}_{d,1} + ... + \lambda_{d,k_d}\boldsymbol{y}_{d,k_d}\,,$$

with

$$\sum_{q=1}^{k_d} \lambda_{d,q} = 1\,,$$
$$\lambda_{d,q} \geq 0\,.$$

We denote the elements of an extreme point $\boldsymbol{y}_{d,q}$ by

$$\boldsymbol{y}_{d,q} = \left(b_d^q, m_{s_d,d}^q, m_{r_d,d}^q, x_d^q(0), \ldots, x_d^q(T-1)\right)\,.$$

By replacing the variables in the central constraints using the above expression, we get the following restricted master LP.

$$\begin{array}{lll}
\text{minimize} & p + \sum_{n\in\mathcal{N}} p_n & \\
\text{subject to} & \sum_{d\in\mathcal{D}} \sum_{q=1}^{k_d} \lambda_{d,q} b_d^q + s = B + p\,, & \\
& \sum_{d\in\mathcal{D} \mid s_d=n \vee r_d=n} \sum_{q=1}^{k_d} \lambda_{d,q} m_{n,d}^q + s_n = M_n + p_n\,, & \forall\, n \in \mathcal{N}\,, \\
& \sum_{q=1}^{k_d} \lambda_{d,q} = 1\,, & \forall\, d \in \mathcal{D}\,, \\
& \lambda_{d,q}, p, p_n, s, s_n \geq 0\,. &
\end{array}$$

Since the solution sets $\mathcal{P}_d$ may contain many extreme points, and since it is generally not easy to determine all these points, we do not calculate all extreme points of each stream solution set. Instead, we use column generation [Gilmore & Gomory], [1961, 1963] to generate the column in the restricted master LP that

	s	s_1	s_2	s_3	s_4	p	p_1	p_2	p_3	p_4	$\lambda_{1,1}$	$\lambda_{2,1}$	$\lambda_{3,1}$
						1	1	1	1	1			
B	1					-1					b_1^1	b_2^1	b_3^1
M_1		1					-1				$m_{1,1}^1$	$m_{1,2}^1$	
M_2			1					-1			$m_{2,1}^1$		$m_{2,3}^1$
M_3				1					-1			$m_{3,2}^1$	
M_4					1					-1			$m_{4,3}^1$
1											1		
1												1	
1													1

Figure 2.2. Starting LP tableau of the master LP for the example of Figure 1.2. On top are the coefficients for each decision variable in the objective function, followed by the rows corresponding to the constraints concerning the total bandwidth reserved, the total buffer size reserved at each node, and the sum of the values of the λ's. The bandwidth and buffer shares of a solution $\boldsymbol{y}_{d,q}$ are denoted as b_d^q and $m_{n,d}^q$, respectively.

corresponds to a specific $\lambda_{d,q}$ when it is needed. Thus, we can start with any solution for each stream solution set, and then iteratively generate new solutions. Figure 2.2 shows an example of the starting LP tableau.

More formally, the initial tableau of the master problem contains for each stream d only one column corresponding to a solution $\boldsymbol{y}_d \in \mathcal{P}_d$. An initial solution $\boldsymbol{y}_d$ does not need to be an extreme point of $\mathcal{P}_d$; any feasible solution will suffice. In other words, only $|\mathcal{D}|$ solutions $\boldsymbol{y}_d$ are considered, corresponding to $|\mathcal{D}|$ variables $\lambda_{d,1}$. After optimizing this restricted master problem, the following step is to determine whether there is a not-yet-considered variable $\lambda_{d,q}$ with negative reduced cost, i.e., a $\lambda_{d,q}$ which can improve the solution of the master LP if it is considered. The reduced cost of a variable can be found in the objective row of the optimized restricted LP tableau. For a non-basic variable the reduced cost is given by $\hat{c} - \hat{\boldsymbol{c}}_B \hat{B}^{-1} \boldsymbol{a}$, with $\hat{\boldsymbol{c}}_B$ the coefficients of the basic variables in the objective function, $\hat{B}^{-1}$ the inverse of the matrix consisting of the columns belonging to the current basic variables, $\boldsymbol{a}$ the column belonging to the non-basic variable under consideration, and $\hat{c}$ the coefficient in the objective function belonging to this non-basic variable. For the columns to be generated, belonging to some $\lambda_{d,q}$, we have $\hat{c} = 0$ and $\boldsymbol{a}$ consists of components b_d, $m_{s_d,d}$, $m_{r_d,d}$, and a 1. This leads to the following minimization problem in which to determine for each stream solution set the element $y \in \mathcal{P}_d$ with the most negative reduced costs, i.e., we have to solve for each stream d the following LP sub-problem[1]:

[1] When supply or demand is controllable, the appropriate buffer size is set to 0 and consequently

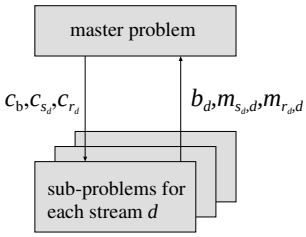

Figure 2.3. A visualization of the decomposition of the multiple-stream problem, where the master problem gives the cost coefficients to the single-stream problems, which return bandwidth and buffer shares for one single stream d, if there is one that improves the solution of the master problem. Otherwise, an optimal solution has been found.

$$\begin{array}{ll} \text{minimize} & c_b b_d + c_{s_d} m_{s_d,d} + c_{r_d} m_{r_d,d} + c_e \\ \text{subject to} & (2.3)-(2.9) \text{ and } (2.12)-(2.14) \text{ for stream } d. \end{array} \tag{2.15}$$

In this LP sub-problem, the coefficients c_b, c_{s_d}, c_{r_d}, and c_e, which are given by the current dual solution $-\hat{\boldsymbol{c}}_B \hat{B}^{-1}$, may take on any real value.

When a non-basic variable with the minimum reduced cost is determined and this reduced cost is negative, then it is entered into the basis of the master LP. After the master LP is re-optimized, we again search for a non-basic variable $\lambda_{d,q}$ with negative reduced cost. If for none of the streams d the sub-problem (2.15) gives a negative reduced cost, then an optimal solution of MSSP has been found. See Figure 2.3 for a picture of the approach.

Although an LP problem can be solved with polynomial-time algorithms, we use the simplex algorithm, which is generally very efficient, but not guaranteed to run in polynomial time. Therefore, the number of sub-problems that are solved, need not be polynomially bounded. However, for real-life instances of data streams, the number of solutions for a stream that may be generated for the master LP is generally low, e.g., we observed that about five solutions per stream were generated in an experiment with four streams. More experimental results concerning different column generation strategies are presented in Section 2.3. Gondzio et al. [1997] address some of the issues that arise when the master LP in a decomposition approach such as Dantzig-Wolfe decomposition, is solved using an interior point method.

does not appear in the LP sub-problem.

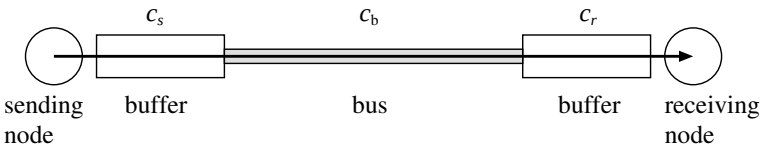

Figure 2.4. The sub-problem for one stream: the stream needs to be transmitted from its sending node to its receiving node for which it uses the buffer between the sending node and the bus, the bandwidth of the bus, and the buffer between the bus and the receiving node. Problem is to reserve shares of the bandwidth and the buffers such that the total costs associated with these reservations are minimized.

2.2 Single-stream methods

Now that we have applied the Dantzig-Wolfe decomposition, we can solve the master problem easily using the simplex algorithm and we only need to determine how to solve sub-problem (2.15). Although (2.15) is an LP problem in itself, the number of variables and constraints concerning the transmission schedule may be very large, e.g., each of the Lord of the Rings movies consist of more than $250,000$ frames, which gives more than $250,000$ variables and $1,250,000$ constraints, if time units are chosen to be equal to the inter-frame times. Thus, we want to use specific algorithms to solve (2.15) more efficiently than standard LP algorithms do. For ease of notation we omit the subscript d in this section as only one stream is considered when solving a sub-problem.

The problem is to minimize $c_\mathrm{b}b + c_s m_s + c_r m_r$ subject to the stream constraints (2.3)–(2.9) and (2.12)–(2.14). Thus, for a single stream we need to determine shares of its sending buffer, bandwidth, and its receiving buffer such that its supply and demand schemes are met; see also Figure 2.4. Notice that c_e can be left out from (2.15), as it is constant. We assume that for $b = B$, $m_s = M_s$, and $m_r = M_r$ a feasible transmission schedule exists, otherwise MSSP cannot be solved. If $b = B$, $m_s = M_s$, and $m_r = M_r$ is infeasible, then each of the methods that we describe in this section, can detect the infeasibility. The values of c_b, c_s, and c_r can be either positive or non-positive, depending on the current solution of the master problem. For each of the seven possible combinations of the signs of the cost coefficients with at least one positive cost coefficient, we present an efficient algorithm. Notice that if a cost coefficient is non-positive, its corresponding resource share can be set to its maximum, since a maximum share only increases the solution space for the transmission of a stream. Thus, if all cost coefficients are non-positive, then an optimal solution is given by $b = B$, $m_s = M_s$, and $m_r = M_r$. We initially describe all algorithms assuming both supply and demand are given. If either supply or demand is controllable, only one buffer will be present in the sub-problem. We will indicate

for the relevant algorithms how they can be adapted to correctly handle this.

In Section 2.2.1 we consider the case for which only the bandwidth needs to be minimized, i.e., only $c_b > 0$. In Section 2.2.2 we consider the cases for which only either the sending buffer share or the receiving buffer share needs to be minimized, i.e., either $c_s > 0$ or $c_r > 0$. In Section 2.2.3 we consider the case for which both the sending buffer share and the receiving buffer share need to be minimized, i.e., $c_s, c_r > 0$. In Section 2.2.4 we show how to solve the case for which an optimal trade-off has to be made between the bandwidth and one buffer, i.e., $c_b > 0$ and either $c_s > 0$ or $c_r > 0$. Finally, in Section 2.2.5 we consider the case for which a trade-off between the bandwidth and both buffer shares has to be made, i.e., $c_b, c_s, c_r > 0$.

2.2.1 Bandwidth minimization

When only c_b is positive, the sub-problem is solved by minimizing the bandwidth share; the values of the buffer shares are given by their maximum, i.e., $m_s = M_s$ and $m_r = M_r$. To minimize bandwidth for given values of the sending and receiving buffer shares, we use an adaptation of the *minimum variability bandwidth allocation* (MVBA) algorithm described by Salehi et al. [1998]. It works as follows. It takes as input a set of constraints $(L(t), U(t))$, which denote a lower bound $L(t)$ for the cumulative transmission schedule $X(t)$, and an upper bound $U(t)$. Output consists of the minimum amount of bandwidth needed and a transmission schedule. Constraints (2.3)–(2.7) give the following upper and lower bounds that a feasible cumulative transmission schedule $X(t)$ has to satisfy to avoid buffer underflow and overflow. For all $t \in \mathcal{T}$,

$$L(t) = \max\{C(t), P(t+1) - m_s\} \leq X(t) \leq \min\{P(t), C(t-1) + m_r\} = U(t).$$

Notice that for controllable supply, no supply scheme is given, and the lower and upper bound thus reduce to $L(t) = C(t)$ and $U(t) = C(t-1) + m_r$, respectively. Similarly, for controllable demand we have $L(t) = P(t+1) - m_s$ and $U(t) = P(t)$.

The cumulative transmission schedule that meets these requirements and also has the lowest peak rate and minimum variance in rate is given by the shortest path from the starting point at time point 0 to the end point at time point T, which remains between $L(t)$ and $U(t)$; see Figure 2.5. It is a piecewise, constant bit-rate schedule, with rate changes whenever we have a full or empty buffer, i.e., when $X(t)$ touches either $L(t)$ or $U(t)$. The time points $t \in \mathcal{T}'$ where the rate in the schedule changes are referred to as *critical points*.

Definition 2.2. Let $[x(1), \ldots, x(T)]$ be an MVBA transmission schedule. A time point $t \in \mathcal{T}' \backslash \{0, T\}$ is said to be a *critical point* if for its adjacent time units t and $t+1$, $x(t) \neq x(t+1)$ holds. A critical point t is said to be *convex* when $x(t) < x(t+1)$, and *concave* when $x(t) > x(t+1)$. The time points 0 and T are defined as

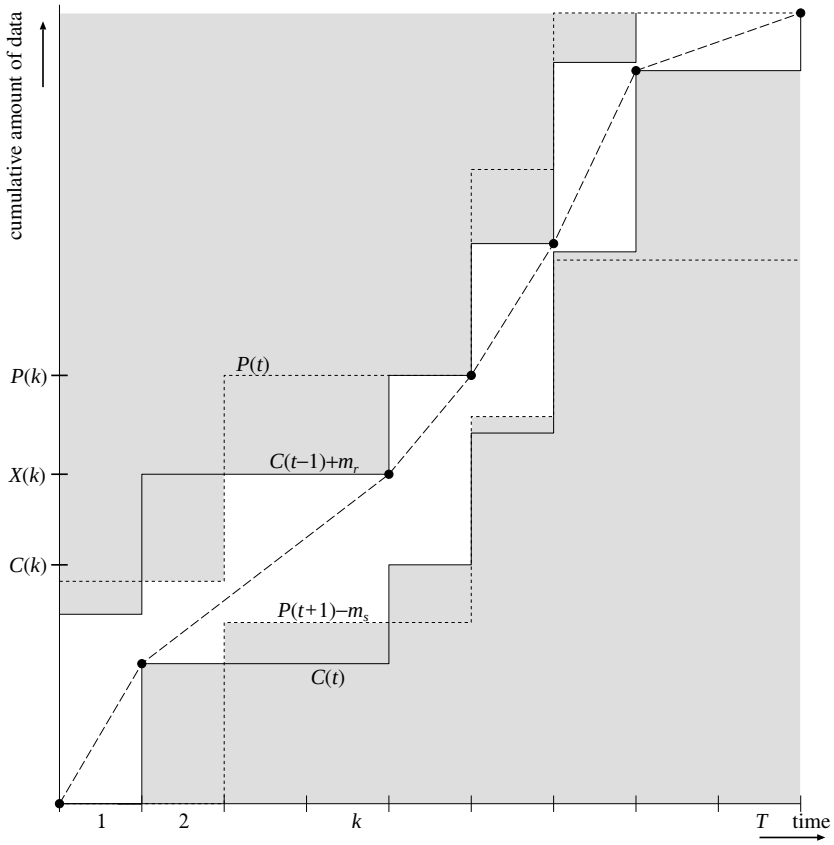

Figure 2.5. Example of an MVBA cumulative transmission schedule. $P(k)$, $X(k)$, and $C(k)$ denote the value of $P(t)$, $X(t)$, and $C(t)$ at the end of time unit k, respectively. The lower bound $L(t)$ and upper bound $U(t)$ are indicated by the gray areas in the figure.

concave critical points. □

We denote the resulting minimum bandwidth by $MVBA(L,U)$. The time complexity of the algorithm presented by Salehi et al. is $\mathcal{O}(T^2)$, but they also mention an $\mathcal{O}(T)$ implementation.

2.2.2 Buffer minimization

When either only c_s or c_r is positive, the sub-problem is solved by minimizing the corresponding buffer share; the values of the bandwidth share and of the buffer share with non-positive cost coefficient are given by their maximum, i.e., $b = B$, $m_s = M_s$ if $c_s \leq 0$, and $m_r = M_r$ if $c_r \leq 0$. For minimization of either the sending buffer share or the receiving buffer share we use an adaptation of the rate-constrained bandwidth smoothing (RCBS) algorithm by Feng [1997]. We distinguish two variants of RCBS: 'backward RCBS' minimizes the receiving buffer size given a demand scheme as described by Feng, and 'forward RCBS' minimizes the sending buffer size given a supply scheme. We describe backward RCBS denoted by $RCBS^{\text{back}}$; forward RCBS, denoted by $RCBS^{\text{for}}$ works analogously. We first describe the algorithm with given demand and controllable supply. Then we show how to adjust the algorithm to take a supply scheme and a limited buffer size at the sending side into account.

$RCBS^{\text{back}}$ uses as input a demand scheme $c(t)$, $t \in \mathcal{T}$, and a bandwidth b. It determines a transmission schedule from the sending side to the receiving side that does not exceed the bandwidth and minimizes the buffer at the receiving side without underflowing or overflowing it. Let $x(t)$ denote the transmission schedule, and let $m(t)$ denote the amount of data that is buffered at the end of time unit t after demand $c(t)$ has been removed from the buffer, with $m(T) = 0$. Then $m(t) = m(t-1) + x(t) - c(t)$. Hence, $m(t-1) = c(t) + m(t) - x(t)$, and so the required buffer size for time unit $t-1$ is minimized if the required buffer size of time unit t is minimized and the transmission in time unit t is maximized. Working backwards from $t = T$ to $t = 1$ the algorithm iteratively checks whether $c(t) + m(t)$ data can be transmitted in one time unit, i.e., whether $c(t) + m(t) \leq b$. If so, this amount is transmitted, otherwise, b is transmitted. The part that cannot be transmitted gives $m(t-1)$. The transmission schedule and amount of data to be buffered between time units are thus determined as follows. For $t = T, T-1, \ldots, 1$,

$$\begin{aligned} x(t) &= \min\{c(t) + m(t), b\}\,, \\ m(t-1) &= c(t) + m(t) - x(t)\,. \end{aligned}$$

Figure 2.6 gives an example of an RCBS transmission schedule. As $m(t)$ denotes the amount of data in the buffer after $c(t)$ is consumed, the actual required buffer size is given by $\max_{t\in\mathcal{T}} m(t) + c(t)$. Obviously, if $m(0) > 0$, then a feasible trans-

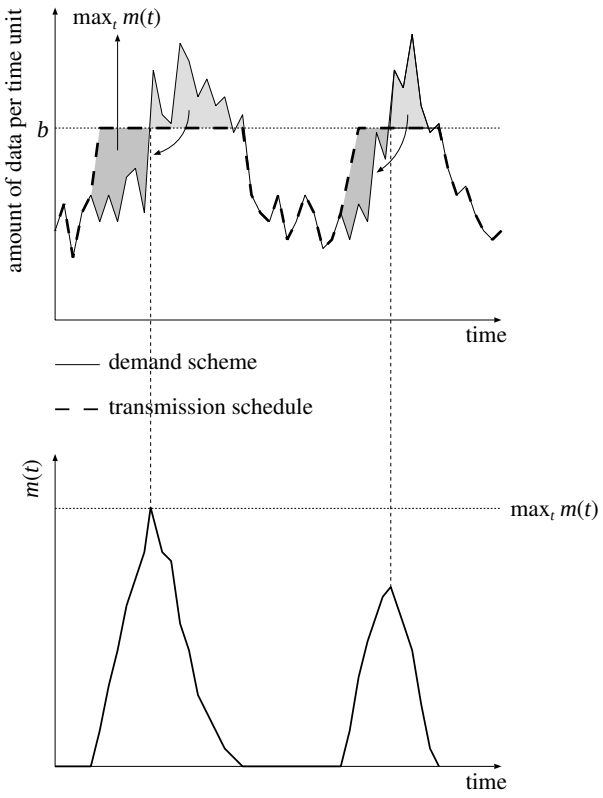

Figure 2.6. Example of an RCBS transmission schedule. The gray areas above the line b depict the amount of data that has to be transmitted earlier than the time unit in which it is demanded. In this case, the largest of these areas is equal to the maximum amount buffered between two time units, i.e., $\max_t m(t)$.

mission schedule does not exist.

Now consider the case in which a supply scheme $p(t)$ and a buffer size m_s at the sending side are given as well. Then $X(t) \geq P(t+1) - m_s$ is an extra constraint for the cumulative transmission schedule; cf. constraint (2.5). The lower bound for $X(t)$ is then given by $L(t) = \max\{C(t), P(t+1) - m_s\}$. To ensure that the cumulative transmission schedule satisfies this lower bound as constraint, we introduce an alternative demand scheme given by $c'(t) = L(t) - L(t-1)$ with $L(0) = 0$. Applying the method as described above, results in a transmission schedule $x(t)$, and an alternative buffered amount $m'(t)$ with $m'(T) = 0$. To determine the amount that is actually buffered between time units, the original demand scheme has to be used.

Algorithm 1 Optimal buffer allocation for any given b and $c_r \geq c_s > 0$

1. $L(t) = \max\{C(t), P(t+1) - M_s\}, t \in \mathcal{T}$
   $c'(t) = L(t) - L(t-1), t \in \mathcal{T}$
2. $m_r = RCBS^{\text{back}}(b, c, c')$
3. $U(t) = \min\{P(t), C(t-1) + m_r\}, t \in \mathcal{T}$
   $p'(t) = U(t) - U(t-1), t \in \mathcal{T}$
4. $m_s = RCBS^{\text{for}}(b, p, p')$

So, for $t = T, T-1, \ldots, 1$,

$$\begin{aligned} x(t) &= \min\{c'(t) + m'(t), b\}\ , \\ m'(t-1) &= c'(t) + m'(t) - x(t)\ , \\ m(t-1) &= c(t) + m(t) - x(t)\ . \end{aligned}$$

Again, the minimum buffer size needed is given by $\max_{t\in\mathcal{T}} m(t) + c(t)$. For later use, we denote this resulting minimum buffer size by $RCBS^{\text{back}}(b, c, c')$. The time complexity of RCBS is $\mathcal{O}(T)$.

2.2.3 Two-buffer minimization

When both c_s and c_r are positive, the sub-problem is solved by minimizing the two buffers jointly; the value of the bandwidth share b is given its maximum B. Notice that this case cannot occur when either supply or demand is controllable as then only one buffer share appears in the LP sub-problem. We assume that $c_r \geq c_s$. If $c_s \geq c_r$ holds, an analogous solution method can be used.

Algorithm 1 gives for a given value of b the optimal values of m_s and m_r. It first uses $RCBS^{\text{back}}$ to minimize the 'expensive' buffer m_r, and then uses $RCBS^{\text{for}}$ to minimize the 'cheap' buffer m_s. A similar algorithm was also presented by Rexford & Towsley [1999] for the joint minimization of the start-up delay and the total buffer allocation.

Theorem 2.1. *The solution (m_s, m_r) that is obtained by execution of Algorithm 1 minimizes total costs $z = c_s m_s + c_r m_r$ if $c_r \geq c_s > 0$.*

Proof. Assume $c_r \geq c_s > 0$. Suppose a solution (m'_s, m'_r) with a feasible transmission schedule exists that results in costs $z' = c_s m'_s + c_r m'_r$ with $z' < z$. As step 2 of Algorithm 1 implies that m_r is the smallest feasible value for the buffer at the receiving node, it follows that $m'_r \geq m_r$. If $m'_r = m_r$, then $m'_s < m_s$ since $z' < z$; however this contradicts step 4 which minimizes m_s for a given m_r. So we know that $m'_r > m_r$, and as $z' < z$, $m_s > m'_s$ must hold. Now we can make the following

derivation:

$$\begin{aligned} m_s + m_r &= \frac{1}{c_s} c_s \left(m_s - m_s'\right) + m_s' + \frac{1}{c_r} c_r \left(m_r - m_r'\right) + m_r' \\ &\geq \frac{1}{c_r} \left(c_s m_s - c_s m_s' + c_r m_r - c_r m_r'\right) + m_s' + m_r' \\ &= \frac{1}{c_r} \left(z - z'\right) + m_s' + m_r' \\ &> m_s' + m_r' \,, \end{aligned}$$

where the first inequality follows from the fact that $c_r \geq c_s > 0$. So the total buffer amount of (m_s', m_r') is smaller than the total buffer amount of (m_s, m_r). We next show that this is impossible.

Consider the greedy transmission schedule $X(t)$, given by $x(t) = \min\{b, P(t) - X(t-1), C(t-1) + m_r - X(t-1)\}$, for the solution (m_s, m_r), and a feasible transmission schedule $X'(t)$ for the solution (m_s', m_r'). Let $\mathcal{T}_s \subset \mathcal{T}$ be the set of all time units t for which $X(t) = P(t+1) - m_s$ holds. As m_s is minimal given m_r, we know that $\mathcal{T}_s \neq \emptyset$. Now we consider two cases. Either there exists a $t_s \in \mathcal{T}_s$ for which $X(t_s)$ is also restricted by the upper bound, i.e., $X(t_s) = \min\{P(t_s), C(t_s - 1) + m_r\}$, or not.

Suppose such a $t_s \in \mathcal{T}_s$ exists. Then either $X(t_s) = C(t_s - 1) + m_r$ or $X(t_s) = P(t_s)$. In the first case, we have $P(t_s + 1) - m_s = X(t_s) = C(t_s - 1) + m_r$, and therefore $m_s + m_r = P(t_s + 1) - C(t_s - 1)$. As $P(t_s + 1) - m_s' \leq X'(t_s) \leq C(t_s - 1) + m_r'$ must also hold, we have $m_s' + m_r' \geq P(t_s + 1) - C(t_s - 1) = m_s + m_r$ which contradicts $m_s + m_r > m_s' + m_r'$.

In the second case, we have $P(t_s + 1) - m_s = X(t_s) = P(t_s)$, and therefore $m_s = P(t_s + 1) - P(t_s)$. As $P(t_s) \geq X'(t_s) \geq P(t_s + 1) - m_s'$, we have $m_s' \geq P(t_s + 1) - P(t_s) = m_s$ which contradicts $m_s' < m_s$.

Now suppose there does not exist a $t_s \in \mathcal{T}_s$ which is restricted by the upper bound. As $X(t)$ is the greedy transmission schedule, it follows that for all $t_s \in \mathcal{T}_s$ we have $x(t_s) = b$. Consider one $t_s \in \mathcal{T}_s$. Let t_e be the last time unit before t_s for which $X(t_e)$ is restricted by the upper bound, i.e., $t_e = \max\{t \in \mathcal{T} \cup \{0\} \mid t < t_s \wedge X(t) = \min\{P(t), C(t-1) + m_r\}\}$. Again, as $X(t)$ is the greedy transmission schedule, it follows that for all t with $t_e < t \leq t_s$, $x(t) = b$ holds.

Now, we first consider $X(t_e) = P(t_e)$. As $m_s' < m_s$, we would like to decrease m_s by a small amount $\varepsilon > 0$. Then the total amount of data transmitted between t_e and t_s, i.e., $\sum_{t=t_e+1}^{t_s} x(t)$ has to be increased by ε; see Figure 2.7(a). However, as for each $t_e < t \leq t_s$, $x(t) = b$, this is impossible, and there cannot exist a solution (m_s', m_r') with $m_s' < m_s$ in this case.

Next we consider $X(t_e) = C(t_e - 1) + m_r$. Again we would like to decrease m_s by a small amount $\varepsilon > 0$ as $m_s' < m_s$; see Figure 2.7(b). Then $X(t_s)$ has to increase

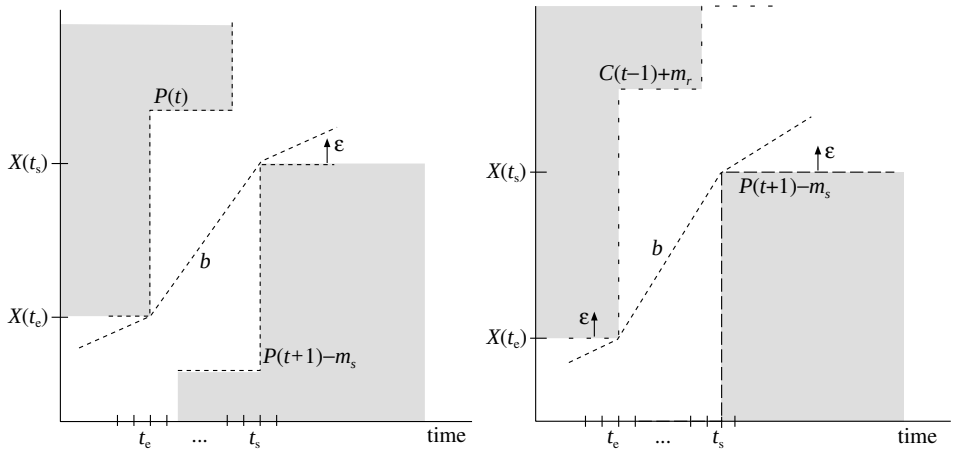

Figure 2.7. Example of possible situations in a schedule when $X(t_s)$ is only restricted by its lower bound. (a) The situation in which $X(t_e)$ is restricted by $P(t_e)$. (b) The situation in which $X(t_e)$ is restricted by $C(t_e-1)+m_r$.

by ε. However, since data is transmitted at maximum bandwidth b during $(t_e, t_s]$, i.e., $\sum_{t=t_e+1}^{t_s} x(t) = (t_s - t_e)\,b$, $X(t_e)$ also has to increase by ε. This means m_r has to be increased by at least ε. So, a solution (m_s', m_r') with $m_s' = m_s - \varepsilon$ must have $m_r' \geq m_r + \varepsilon$ in this case, giving $m_s' + m_r' \geq m_s + m_r$. This contradicts the fact that the total buffer amount of (m_s', m_r') is smaller than the total buffer amount of (m_s, m_r). Therefore, there does not exist a solution (m_s', m_r') with costs $z' < z$, and hence the obtained solution (m_s, m_r) is optimal. □

2.2.4 Trade-off with one buffer

When c_b is positive together with either c_s or c_r, the sub-problem is solved by making a trade-off between the bandwidth and the relevant buffer share; the value of the remaining buffer share is again given by its maximum. In this section we present an algorithm for this trade-off between the bandwidth and one buffer share, called the bandwidth-buffer trade-off algorithm. It determines an optimal balance between bandwidth b and buffer size at the receiving side m_r, given their cost coefficients c_b and c_r. The trade-off between b and m_s can be performed analogously. We begin with presenting the algorithm and how it works. Then we show its correctness and finally we discuss the time complexity of the algorithm. When the supply is controllable, the same algorithm as we present here can be used. The only difference lies in the upper and lower bounds used for the transmission schedule, which then do not involve the supply scheme and sending buffer.

Another algorithm that determines the bandwidth-buffer trade-off curve is de-

Algorithm 2 Global overview of the bandwidth-buffer trade-off algorithm.

Determine initial solution:

1. $L(t) = \max\{C(t), P(t+1) - M_s\}$
   $c'(t) = L(t) - L(t-1)$
2. $m_r = RCBS^{\text{back}}(B, c, c')$
3. $U(t) = \min\{P(t), C(t-1) + m_r\}$
4. $b = MVBA(L, U)$

Initialize trade-off:

5. Execute Algorithm 3.

Perform trade-off:

6. Execute Algorithm 4.

scribed by Chang et al. [1998]. Den Boef et al. [2004] give a comparison of the method by Chang et al. and the bandwidth-buffer trade-off algorithm. They show that the run times of both algorithms are more or less equal. However, the memory requirements of the bandwidth-buffer trade-off algorithm are only half the memory requirements of the method by Chang et al.

The bandwidth-buffer trade-off algorithm.

Algorithm 2 gives a global overview of how the problem is solved. The algorithm begins with determining an initial solution using RCBS to minimize m_r for which b is set to the maximum bandwidth B (steps 1 and 2). After this, the algorithm uses MVBA to minimize b for the obtained m_r and to determine an initial MVBA transmission schedule (steps 3 and 4). Then, it performs a trade-off between the reserved bandwidth b and the buffer size m_r in which it iteratively tries to increase the buffer size m_r and determines what effect it has on the smallest possible choice of b, i.e., on $\max_t x(t)$ (steps 5 and 6).

Algorithm 3 initializes the trade-off. It partitions the time horizon into several distinct intervals V_i, where the intervals are separated from each other by the concave critical points, as is shown in Figure 2.8. Interval V_i lies between critical points v_i and v_{i+1}.

Next, Algorithm 3 determines the following parameters for each interval V_i; see Figure 2.8. The maximum transmission rate during V_i is denoted by $b_i^{\max}$ and occurs at the end of interval V_i. The minimum transmission rate during V_i is denoted by $b_i^{\min}$ and occurs at the beginning of the interval. If $b_i^{\max} > b_i^{\min}$, then the algorithm also determines the best achievable maximum transmission rate, b_i^{high}, considering only the interval V_i and assuming there is a buffer of maximum size M_r at the receiving side. In other words, b_i^{high} is the maximum transmission rate during the interval V_i of the MVBA schedule that starts at v_i and finishes at v_{i+1} with

Algorithm 3 Initialization for the trade-off between bandwidth and buffer size

Determine the set of concave critical points $\mathcal{C} = \{v_1, v_2, \ldots, v_k\} \subseteq \mathcal{T}'$.

$V_i = (v_i, v_{i+1}], i \in \mathcal{I} = \{1, \ldots, k-1\}$;
For all $i \in \mathcal{I}$ determine
  $b_i^{\max} = \max_{t \subseteq V_i} x(t) = x(v_{i+1})$; $\quad b_i^{\min} = \min_{t \subseteq V_i} x(t) = x(v_i + 1)$;
  If $b_i^{\max} > b_i^{\min}$ **then** determine
    $L(t) = \max\{C(t), P(t+1) - M_s\}$; $\quad U(t) = \min\{P(t), C(t-1) + M_r\}$;
    $(b_i^{\text{high}}, b_i^{\text{low}}) = MVBA_i^{\text{int}}(L, U)$
    If $b_i^{\max} > b_i^{\text{high}}$ **then** determine
      $b_i^{\max-1} = \max_{t \subseteq V_i, x(t) \neq b_i^{\max}} x(t)$; $\quad b_i^{\min+1} = \min_{t \subseteq V_i, x(t) \neq b_i^{\min}} x(t)$;
      $y_i^{\max} = |\{t \in \mathcal{T} \vee t \subseteq V_i | x(t) = b_i^{\max}\}|$; $\quad y_i^{\min} = |\{t \in \mathcal{T} \vee t \subseteq V_i | x(t) = b_i^{\min}\}|$;
      $r_i^{\max} = 1/y_i^{\max}$; $\quad r_i^{\min} = 1/y_i^{\min}$;
      $$\triangle_i m^{\max} = \begin{cases} y_i^{\max}(b_i^{\max} - b_i^{\max-1}) & \text{if } b_i^{\max-1} \geq b_i^{\text{high}} \\ y_i^{\max}(b_i^{\max} - b_i^{\text{high}}) & \text{otherwise.} \end{cases};$$
      $$\triangle_i m^{\min} = \begin{cases} y_i^{\min}(b_i^{\min+1} - b_i^{\min}) & \text{if } b_i^{\min+1} \leq b_i^{\text{low}} \\ y_i^{\min}(b_i^{\text{low}} - b_i^{\min}) & \text{otherwise.} \end{cases};$$
    endif
  endif
  If $b_i^{\max} = b_i^{\text{high}}$ **then**
    $r_i^{\max} = 0$; $\quad \triangle_i m^{\max} = M_r - m_r$;
  endif
  If $b_i^{\min} = b_i^{\text{low}}$ **then**
    $r_i^{\min} = 0$; $\quad \triangle_i m^{\min} = M_r - m_r$;
  endif
endfor

$\mathcal{I}^1 = \{i \in \mathcal{I} | b_i^{\max} \geq b_j^{\max}, \forall j \in \mathcal{I}\}$
$\mathcal{I}^2 = \{i \in \mathcal{I}^1 | r_i^{\max} \leq r_j^{\max}, \forall j \in \mathcal{I}^1\}$
$\mathcal{I}^3 = \{i \in \mathcal{I}^2 | \triangle_i m^{\max} \leq \triangle_j m^{\max}, \forall j \in \mathcal{I}^2\}$
Pick $i^* \in \mathcal{I}^3$
For all V_i **do**
  if $r_{i^*}^{\max} > r_i^{\max}$ **then**
    $\gamma_i = \frac{b_{i^*}^{\max} - b_i^{\max}}{r_{i^*}^{\max} - r_i^{\max}}$;
  else $\gamma_i = M_r - m_r$;
  if $r_i^{\max} + r_{i+1}^{\min} > 0$ **then**
    $\varepsilon_i = \frac{b_i^{\max} - b_{i+1}^{\min}}{r_i^{\max} + r_{i+1}^{\min}}$;
  else $\varepsilon_i = M_r - m_r$;
od

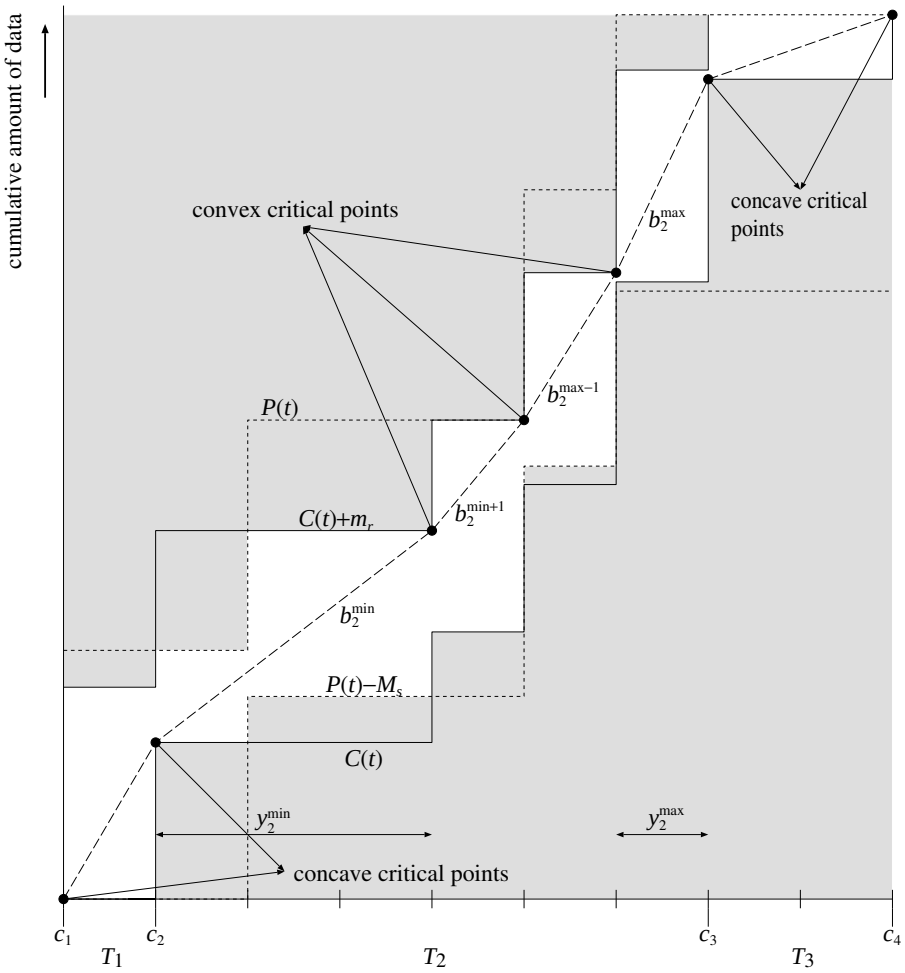

Figure 2.8. Parameters of an MVBA-schedule for the trade-off algorithm.

lower bound $\max\{C(t), P(t+1) - M_s\}$ and upper bound $\min\{P(t), C(t-1) + M_r\}$ for all time units $t \in \mathcal{T}$ with $t \subseteq V_i$. The algorithm also determines the lowest transmission rate, b_i^{low}, that occurs in this MVBA schedule. To determine b_i^{high} and b_i^{low} the algorithm uses a special *interval version* of MVBA defined as follows. For an example, see Figure 2.9.

Definition 2.3. $MVBA_i^{\text{int}}(L,U)$ returns the maximum and minimum transmission rate of an MVBA schedule for an interval V_i assuming V_i is the total time period and given a lower bound function L and an upper bound function U. It starts and ends at the concave critical points v_i and v_{i+1}. □

If $b_i^{\max} > b_i^{\text{high}}$, then Algorithm 3 determines the second highest transmission rate, denoted by $b_i^{\max-1}$, and the second lowest transmission rate, denoted by $b_i^{\min+1}$. Finally, it determines how many time units during V_i have $b_i^{\max}$ and $b_i^{\min}$ as transmission rate, which is denoted by $y_i^{\max}$ and $y_i^{\min}$, respectively; see Figure 2.8.

When the buffer m_r is increased, this affects the transmission rates at the intervals for which $b_i^{\max} > b_i^{\text{high}}$. The maximum transmission rate decreases for each unit of buffer increase by a rate $r_i^{\max} = 1/y_i^{\max}$. This rate is valid for only a limited amount of buffer increase, $\triangle_i m^{\max}$, which gives the buffer increase until either $b_i^{\max} = b_i^{\max-1}$ or $b_i^{\max} = b_i^{\text{high}}$, whichever holds first. Analogously, the rate by which $b_i^{\min}$ increases for each unit of buffer increase is given by $r_i^{\min} = 1/y_i^{\min}$, and the maximum amount that m_r may increase for $r_i^{\min}$ to remain valid is given by $\triangle_i m^{\min}$. Notice that either $r_i^{\max}\triangle_i m^{\max} = b_i^{\max} - b_i^{\max-1}$ or $r_i^{\max}\triangle_i m^{\max} = b_i^{\max} - b_i^{\text{high}}$. For each interval V_i for which $b_i^{\max} = b_i^{\text{high}}$, the algorithm sets $r_i^{\max} = 0$, and $\triangle_i m^{\max} = M_r - m_r$ since $b_i^{\max}$ in this case does not change. Analogously, if $b_i^{\min} = b_i^{\text{low}}$, then the algorithm sets $r_i^{\min} = 0$ and $\triangle_i m^{\min} = M_r - m_r$. This also holds for intervals with $b_i^{\max} = b_i^{\min}$, since then $b_i^{\max} = b_i^{\text{high}} = b_i^{\text{low}} = b_i^{\min}$.

Now, we know for each interval V_i that when we increase m_r by a small amount $\triangle m$ its maximum transmission rate $b_i^{\max}$ decreases by $r_i^{\max}\triangle m$. As $b = \max_i b_i^{\max}$, the algorithm next selects the interval that has the highest transmission rate, the lowest $r_i^{\max}$, and the lowest $\triangle_i m^{\max}$, i.e., it chooses i^* that lexicographically maximizes $(b_i^{\max}, -r_i^{\max}, -\triangle_i m^{\max})$. Then, V_{i^*} is the interval with the highest transmission rate $b_i^{\max}$, also after a small increase $\triangle m$ of m_r. Furthermore, if there is more than one such interval, V_{i^*} is the interval for which $r_i^{\max}$ is valid for the smallest buffer increase $\triangle_i m^{\max}$. So b decreases with $r_{i^*}^{\max}\triangle m$, and thus the total costs change by an amount $c_r\triangle m - c_b r_{i^*}^{\max}\triangle m = (c_r - c_b r_{i^*}^{\max})\triangle m$. Therefore, if $c_r < c_b r_{i^*}^{\max}$ then the total costs decrease when m_r is increased by a small amount $\triangle m$.

Before the trade-off can start we first determine the values of parameters γ_i and ε_i as indicated in the algorithm. These parameters give the amount by which the

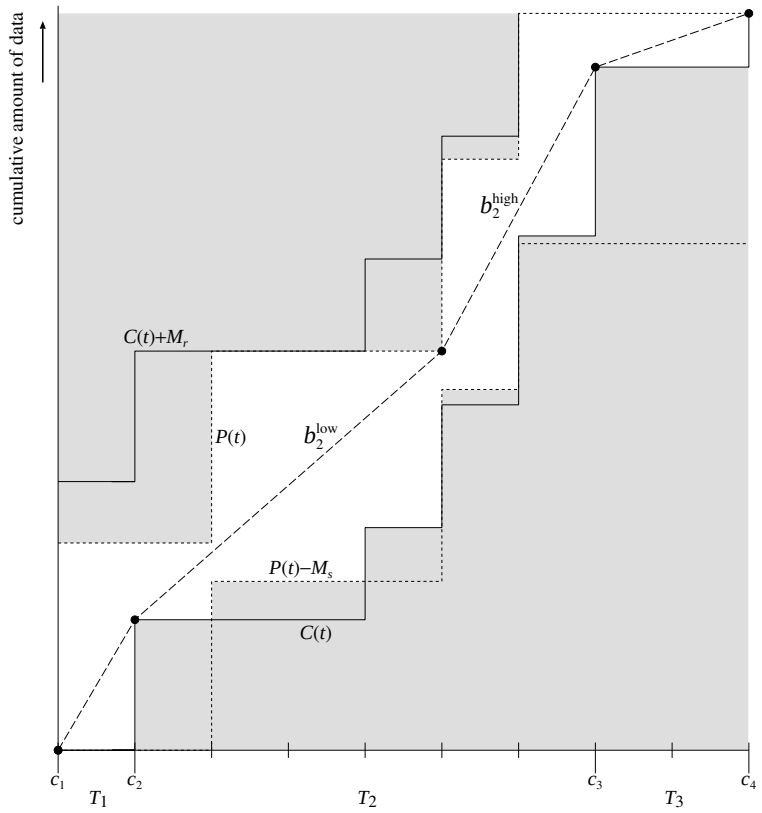

Figure 2.9. Example of an MVBA-schedule for an interval in the trade-off algorithm which leads to b_i^{high} and b_i^{low}.

buffer size m_r can be increased before an event takes place that directly or indirectly influences the rate at which the highest transmission rate decreases. Below, we describe all these events, and how large the increase $\triangle m$ is for them to happen.

- The highest transmission rate b can shift from V_{i^*} to another interval V_i. This can happen for V_i with $r_i^{\max} < r_{i^*}^{\max}$ as then $b_i^{\max}$ decreases slower than $b = b_{i^*}^{\max}$. We define γ_i as the amount by which m_r has to be increased such that $b_i^{\max}$ becomes equal to $b_{i^*}^{\max}$. For this, $b_{i^*}^{\max} - r_{i^*}^{\max}\gamma_i = b_i^{\max} - r_i^{\max}\gamma_i$, so $\gamma_i = \frac{b_{i^*}^{\max} - b_i^{\max}}{r_{i^*}^{\max} - r_i^{\max}}$. For i with $r_i^{\max} \geq r_{i^*}^{\max}$, we set $\gamma_i = M_r - m_r$.
- A convex critical point may no longer be a critical point or a new convex critical point can emerge, which may affect $r_i^{\max}$ or $r_i^{\min}$. This happens when the buffer is increased by $\triangle_i m^{\max}$ or $\triangle_i m^{\min}$.
- A concave critical point may no longer be a critical point, which means that two intervals merge into one. This happens when the buffer is increased by ε_i, which is the amount by which m_r has to be increased such that $b_i^{\max}$ becomes equal to $b_{i+1}^{\min}$. For this, $b_i^{\max} - b_{i+1}^{\min}\varepsilon_i = b_{i+1}^{\min} + r_{i+1}^{\min}\varepsilon_i$, so $\varepsilon_i = \frac{b_i^{\max} - b_{i+1}^{\min}}{r_i^{\max} + r_{i+1}^{\min}}$. For i with $r_i^{\max} + r_i^{\min} = 0$, we set $\varepsilon_i = M_r - m_r$.

Now, Algorithm 4 iteratively increases the buffer m_r as long as the rate by which the bandwidth decreases per unit of buffer increase is high enough for an increase in the buffer to be cost efficient, i.e., while $c_r - c_b r_{i^*}^{\max} < 0$. The algorithm chooses the increase in m_r equal to $\triangle m = \min_i\{\gamma_i, \triangle_i m^{\max}, \triangle_i m^{\min}, \varepsilon_i\}$. This leads to one of the described events, after which the actions shown in Algorithm 4 are taken to adjust all the parameters for the newly obtained transmission schedule. These actions are merely book-keeping of all the parameter values. Of course, all parameters of each interval have to be adjusted for the increase in buffer size. If b shifts to another interval, then also all γ_i values have to be redetermined, since they signal when this may happen. If the number of convex critical points changes, there are four possibilities. If either b_i^{high} or b_i^{low} has been reached on an interval, then $r_i^{\max}$ and $r_i^{\min}$ are set to 0, respectively, and the event-detecting parameters γ_i and ε_i in case b_i^{high} has been reached, and ε_{i-1} in case b_i^{low} has been reached, have to be redetermined. If $b_i^{\max} = b_i^{\max-1}$ then $y_i^{\max}$, the length of the part of the interval with this maximum transmission rate, has to be redetermined, together with $r_i^{\max}$ and $\triangle_i m^{\max}$. Also, all event-detecting parameters of this interval then have to be redetermined. Furthermore, if it concerns interval V_{i^*}, then all γ_i have to be redetermined. If $b_i^{\min} = b_i^{\min+1}$ we have to take similar adjusting actions. Finally, if two intervals merge, then all parameters after the merge are renumbered by decreasing their index by one, since there is one interval fewer. For the new interval, b_i^{high} and b_i^{low} have to be redetermined, and thus also $\triangle_i m^{\max}$.

Algorithm 4 The trade-off execution

while $c_r - c_b r_{i^*}^{\max} < 0$ **do**
$\triangle m = \min_i\{\gamma_i, \triangle_i m^{\max}, \triangle_i m^{\min}, \varepsilon_i\}$

Actions to adjust parameters of transmission schedule after an event

For all $j \in \mathcal{I}$ **do**
$\gamma_j = \gamma_j - \triangle m$; $\varepsilon_j = \varepsilon_j - \triangle m$;
$\triangle_j m^{\max} = \triangle_j m^{\max} - \triangle m$; $\triangle_j m^{\min} = \triangle_j m^{\min} - \triangle m$;
$b_j^{\max} = b_j^{\max} - r_j^{\max}\triangle m$; $b_j^{\min} = b_j^{\min} + r_j^{\min}\triangle m$;
endfor
If b shifts to interval V_i **then**
$i^* = i$;
For all $j \neq i^*$ **do**
If $r_j^{\max} < r_{i^*}^{\max}$ **then** Redetermine γ_j;
else $\gamma_j = M_r - m_r$;
endfor
endif
If the number of convex critical points in interval V_i changes **then**
If $b_i^{\max} = b_i^{\text{high}}$ **then**
$r_i^{\max} = 0$; $\triangle_i m^{\max} = M_r - m_r$;
Redetermine γ_i and ε_i;
else if $b_i^{\max} = b_i^{\max-1}$ **then**
Redetermine $y_i^{\max}, r_i^{\max}, \triangle_i m^{\max}$, and $b_i^{\max-1}$;
Redetermine γ_i and ε_i;
If $i = i^*$ **then for all** $j \neq i^*$ Redetermine γ_j;
endif
If $b_i^{\min} = b_i^{\text{low}}$ **then**
$r_i^{\min} = 0$; $\triangle_i m^{\min} = M_r - m_r$;
Redetermine ε_{i-1};
else if $b_i^{\min} = b_i^{\min+1}$ **then**
Redetermine $y_i^{\min}, r_i^{\min}, \triangle_i m^{\min}$, and $b_i^{\min+1}$;
Redetermine ε_{i-1};
endif
endif
If the number of concave critical points decreases, i.e., V_i merges with V_{i+1} **then**
Let $V_i = (v_k, v_l]$ and $V_{i+1} = (v_l, v_m]$ then $V_i = (v_k, v_m]$;
$y_i^{\max} = y_{i+1}^{\max}$; $r_i^{\max} = r_{i+1}^{\max}$;
$(b_i^{\text{high}}, b_i^{\text{low}}) = MVBA_i^{\text{int}}(L, U)$
Adjust $\triangle_i m^{\max}$;
$\gamma_i = \gamma_{i+1}$; $\varepsilon_i = \varepsilon_{i+1}$;
For all $j > i$ **do** Reassign parameters$_j$:= parameters$_{j+1}$;
endif
endwhile

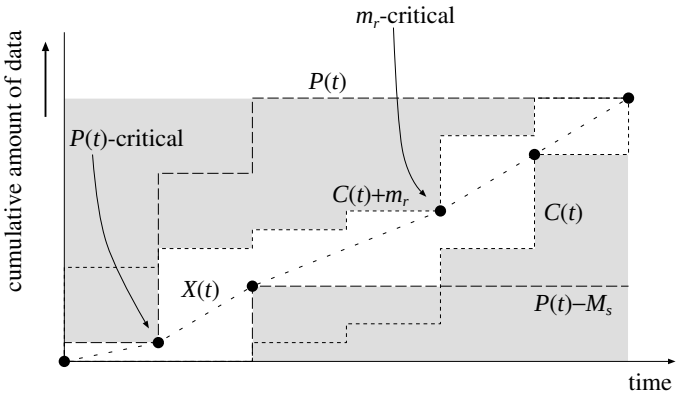

Figure 2.10. $P(t)$- and m_r-critical points.

The algorithm continues with increasing the receiving buffer while $c_r - c_b r_{i^*}^{\max} < 0$. When $c_r - c_b r_{i^*}^{\max} \geq 0$, the optimal value for b equals the maximum transmission rate in the transmission schedule found.

Correctness proof.

We next prove the correctness of the algorithm. For this, we distinguish two types of convex critical points. First, at a $P(t)$-critical point $t_0 \in \mathcal{T}'$, $X(t)$ touches $P(t)$, i.e., $X(t_0) = P(t_0)$ with t_0 here the corresponding time unit in $\mathcal{T}$. Secondly, at an m_r-critical point $t_0 \in \mathcal{T}'$, $X(t)$ touches $C(t) + m_r$ but it does not touch $P(t)$, i.e., $X(t_0) = C(t_0) + m_r < P(t_0)$; see Figure 2.10 for an example. Notice that for controllable supply only m_r-critical points exist.

Furthermore, we give the *refinement theorem* and *domination theorem* of Sanjay & Raghavan [1999]. We use them to show some results concerning the critical points when the buffer m_r is increased. Then, we show that the algorithm correctly transforms the minimum and maximum transmission rates during the intervals of the transmission schedule into the minimum and maximum transmission rates during the intervals of a new MVBA transmission schedule. We continue the proof by showing that all possible events that may influence the maximum transmission rate are correctly detected. Finally, we show the correctness of the stopping criterion of the algorithm.

First, we give the refinement theorem of which the proof is given by Sanjay & Raghavan [1999].

Theorem 2.2. *Let X_1 and X_2 be two MVBA transmission schedules with constraint sets (L_1, U_1) and (L_2, U_2), respectively. Furthermore, let $L_1(t) = L_2(t)$ and $U_1(t) \leq U_2(t)$, for all $t \in \mathcal{T}$. Then the following properties hold.*

(a) The concave critical points of X_1 are a superset of the concave critical points of X_2.

(b) If $U_2(t) \leq U_1(t) + k$, for all $t \in \mathcal{T}$, then $X_2(t) \leq X_1(t) + k$, for all $t \in \mathcal{T}$.

(c) If during an interval $[q_1, q_2], U_2(t) = U_1(t) + k$, for all $t \in [q_1, q_2]$, then the convex critical points of X_1 in $[q_1, q_2]$ are a superset of the convex critical points of X_2 in $[q_1, q_2]$.

□

The refinement theorem leads to the following corollary.

Corollary 2.1. *An increase in the buffer size m_r may have the effect that an m_r-critical point disappears, but no m_r-critical point will emerge.*

Proof. It is trivial that an m_r-critical point may disappear, so we only show that an m_r-critical point cannot emerge. Let X be an MVBA transmission schedule and let $t_0 \in \mathcal{T}$. If an m_r-critical point has to emerge at the end of time unit t_0, so at time point $t_0 \in \mathcal{T}'$, then the supply scheme must lie above the receiving buffer constraint and t_0 is not yet an m_r-critical point, i.e., we must have

$$C(t_0 - 1) + m_r < P(t_0) \ \wedge \ X(t_0) < C(t_0 - 1) + m_r.$$

So $U(t_0) = \min\{P(t_0), C(t_0 - 1) + m_r\} = C(t_0 - 1) + m_r$. If we denote the increase in m_r by $\triangle m$, which is chosen such that $C(t_0 - 1) + m_r + \triangle m \leq P(t_0)$, then for the new MVBA schedule X' for the buffer size $m_r + \triangle m$,

$$U'(t_0) = \min\{P(t_0), C(t_0 - 1) + m_r + \triangle m\} = C(t_0 - 1) + m_r + \triangle m = U(t_0) + \triangle m$$

holds. Then, part (c) of Theorem 2.2 states that the convex critical points of X in $[t_0, t_0]$ are a superset of the convex critical points of X' in $[t_0, t_0]$. Since t_0 is not a convex critical point in X, it cannot be a convex critical point in X' either. Therefore, no m_r-critical point will emerge when the buffer size m_r is increased. □

A consequence of the above corollary is that an m_r-critical point that has disappeared during the execution of the algorithm will not emerge again.

Another theorem given by Sanjay & Raghavan [1999] is the domination theorem.

Theorem 2.3. *Let X_1 and X_2 be two MVBA transmission schedules with constraint sets (L_1, U_1) and (L_2, U_2), respectively. Furthermore, let $L_2(t) \geq L_1(t)$ and $U_2(t) \geq U_1(t)$, for all $t \in [q_1, q_2]$. If $X_2(q_1) \geq X_1(q_1)$ and $X_2(q_2) \geq X_1(q_2)$, then $X_2(t) \geq X_1(t)$, for all $t \in [q_1, q_2]$.* □

With the domination theorem, we can prove the following corollary.

Corollary 2.2. *An increase in the buffer size m_r will not cause a $P(t)$-critical point to disappear and can only cause a $P(t)$-critical point to emerge if it is present in*

the MVBA schedule X^∞ with $U^\infty(t) = P(t)$ and $L^\infty = L$.

Proof. Let X be an MVBA transmission schedule with constraint set (L,U). Suppose $t_0 \in \mathcal{T}'$ is a $P(t)$-critical point of X, i.e., $X(t_0) = P(t_0)$. Now let $\triangle m$ denote the increase of buffer size m_r, and X' the resulting MVBA transmission schedule. Then X' has constraint set (L',U') with $L' = L$ and $U'(t) \geq U(t)$, for all $t \in \mathcal{T}$. According to Theorem 2.3, $X'(t) \geq X(t)$, for all $t \in \mathcal{T}$. Since $X'(t_0) \leq P(t_0) = X(t_0)$, this implies $X'(t_0) = X(t_0) = P(t_0)$, so the $P(t)$-critical point t_0 will not disappear.

Now suppose t_0 is not a $P(t)$-critical point in X^∞. Then $X^\infty(t_0) < P(t_0) = X(t_0)$. But $U^\infty(t) \geq U(t)$, for all $t \in \mathcal{T}$, therefore, $X^\infty(t_0) \geq X(t_0)$, which gives a contradiction. Thus, t_0 has to be a $P(t)$-critical point in X^∞, and therefore, a $P(t)$-critical point can only emerge if it is present in X^∞. □

For the concave critical points we can prove the following lemma.

Lemma 2.1. *An increase in the buffer size m_r may have the effect that several consecutive intervals merge, but none of the intervals will split into more intervals.*

Proof. Let X be an MVBA transmission schedule with constraint set (L,U). Let $\triangle m$ denote the increase of buffer size m_r, and X' the resulting MVBA transmission schedule. Then X' has constraint set (L',U') with $L' = L$ and $U'(t) \geq U(t)$, for all $t \in \mathcal{T}$. Then Theorem 2.2 part (a) states that the concave critical points of $X(t)$ are a superset of the concave critical points of $X'(t)$. Therefore, several consecutive intervals may merge, but none of the intervals will split. □

We continue by showing that the maximum and minimum transmission rates during each interval of the MVBA transmission schedule, i.e., $b_i^{\max}$ and $b_i^{\min}$, are correctly transformed into their respective values during the intervals of a new MVBA transmission schedule, as long as critical points do not emerge or disappear. Let X be an MVBA transmission schedule with constraint set (L,U). Now let the increase of m_r be denoted by $\triangle m$. Then, only at m_r-critical points the upper bound shifts upwards with $\triangle m$, whereas at $P(t)$-critical points the upper bound and at concave critical points the lower bound do not change in value. As an MVBA transmission schedule has a constant transmission rate between two successive critical points, the transmission rate between two critical points only changes when the upper bound at one of them shifts upwards, so at an m_r-critical point, and when the upper or lower bound does not change at the other one, so at a $P(t)$-critical point or a concave critical point, respectively.

Let t_0 be an m_r-critical point and $t_1 > t_0$ a concave critical point adjacent to t_0, i.e., there are no other critical points between t_0 and t_1. Let $(t_0,t_1] \subset V_i$. Then, obviously, $b_i^{\max}$ is the transmission rate during $(t_0,t_1]$. The amount by which $b_i^{\max}$

changes with each unit of buffer size increase is given by the change in the slope of the cumulative transmission schedule. This is equal to $-1/(t_1 - t_0)$, i.e., minus the increase in buffer size, or the decrease in amount of data transmitted during $(t_0, t_1]$, divided by the length of the interval, which is equal to the number of time units in the interval with the maximum transmission rate, i.e., $y_i^{\max}$. The change in slope is then given by $r_i^{\max} = 1/y_i^{\max}$. If t_0 is a concave critical point and $t_1 > t_0$ an m_r-critical point adjacent to t_0, an analogous reasoning can be held for $b_i^{\min}$ and $r_i^{\min}$.

Now suppose t_0 is a $P(t)$-critical point and $t_1 > t_0$ is again a concave critical point adjacent to t_0. Then the upper bound at t_0 will not change and t_0 will not disappear; cf. Corollary 2.2. Thus, an increase of m_r will not change the transmission rate in $(t_0, t_1]$, and hence $r_i^{\max} = 0$ in that case. This corresponds to $b_i^{\max} = b_i^{\text{high}}$. A similar reasoning holds for $r_i^{\min}$ with t_0 a concave critical point and $t_1 > t_0$ a $P(t)$-critical point adjacent to t_0.

So, as long as critical points do not emerge or disappear, not considering the 'interior' of the intervals, the maximum and minimum transmission rates are correctly updated by the algorithm. The algorithm distinguishes three possible events, when the buffer size m_r is increased, which may directly or indirectly influence the maximum transmission rate of the transmission schedule. Obviously, the algorithm has to know what the maximum transmission rate is so it can determine whether an increase is interesting. Therefore, it has to keep track of the interval with the maximum transmission rate, and thus it has to know when this shifts to another interval. Furthermore, the disappearance or emergence of a critical point may affect the transmission rates in the intervals, and therefore have to be detected. Lemma 2.1 and Corollaries 2.1 and 2.2 state that concave critical points and m_r-critical points can only disappear, whereas $P(t)$-critical points can only emerge.

For the maximum transmission rate to shift to another interval, the buffer size m_r has to be increased by an amount that causes the maximum transmission rate on an interval V_i to be equal to the maximum transmission rate on the interval V_{i^*}. This can only happen if $r_i^{\max} < r_{i^*}^{\max}$. Let γ_i denote the required increase in buffer size for the maximum transmission rate to shift to interval V_i. Then as mentioned previously $b_{i^*}^{\max} - \gamma_i r_{i^*}^{\max} = b_i^{\max} - \gamma_i r_i^{\max}$ holds, and we have $\gamma_i = \frac{b_{i^*}^{\max} - b_i^{\max}}{r_{i^*}^{\max} - r_i^{\max}}$.

For a concave critical point $t \notin \{0, T\}$, $x(t) > x(t+1)$ holds. So, a concave critical point v_i disappears as soon as $x(v_i) \leq x(v_i + 1)$. Let ε_i denote the required increase in buffer size for concave critical point v_{i+1} between V_i and V_{i+1} to disappear. Then as mentioned previously $b_i^{\max} - \varepsilon_i r_i^{\max} = b_{i+1}^{\min} + \varepsilon_i r_{i+1}^{\min}$ holds, and we have $\varepsilon_i = \frac{b_i^{\max} - b_{i+1}^{\min}}{r_i^{\max} + r_{i+1}^{\min}}$.

Now we show that $\triangle_i m^{\max}$ and $\triangle_i m^{\min}$ correctly signal the disappearance and emergence of a convex critical point when it may influence the rate at which $b_i^{\max}$

decreases. The rate $r_i^{\max}$ with which $b_i^{\max}$ decreases, changes when either the last convex critical point of V_i disappears or a $P(t)$-critical point emerges after the last convex critical point of V_i. In the latter case, $b_i^{\max}$ has become equal to b_i^{high}, since the $P(t)$-critical point will not disappear; cf. Corollary 2.2.

In the former case, i.e., the last convex critical point has disappeared, there are two possibilities. Either it was the only convex critical point of V_i, in which case $b_i^{\max}$ has become equal to b_i^{high}, or there is another convex critical point in V_i. If the new last convex critical point is a $P(t)$-critical point then $b_i^{\max}$ has become equal to b_i^{high}. If it is an m_r-critical point, then $b_i^{\max-1}$ did not change with the buffer increase, since the upper bound shifted upwards at both its surrounding critical points. Therefore, $b_i^{\max}$ then has become equal to $b_i^{\max-1}$.

Suppose a $P(t)$-critical point emerges before the last convex critical point. Then it has to be a $P(t)$-critical point in X_i^{int}, the MVBA$^{\text{int}}$ schedule for V_i. Therefore, if it influences $b_i^{\max-1}$, $b_i^{\text{high}} > b_i^{\max-1}$ must hold, and $b_i^{\max}$ will become equal to b_i^{high} when the last convex critical point disappears. If it does not influence $b_i^{\max-1}$, then $b_i^{\max} = b_i^{\max-1}$ will hold when the last convex critical point disappears[2].

Now we have to determine what increase in buffer size leads to $b_i^{\max} = b_i^{\max-1}$, or $b_i^{\max} = b_i^{\text{high}}$ if $b_i^{\text{high}} > b_i^{\max-1}$. Let $\triangle_i m^{\max}$ denote this increase, then we have

$$b_i^{\max} - r_i^{\max}\triangle_i m^{\max} = \begin{cases} b_i^{\max-1} & \text{if } b_i^{\max-1} \geq b_i^{\text{high}} \\ b_i^{\text{high}} & \text{otherwise.} \end{cases}$$

Rewriting gives

$$\triangle_i m^{\max} = \begin{cases} y_i^{\max}(b_i^{\max} - b_i^{\max-1}) & \text{if } b_i^{\max-1} \geq b_i^{\text{high}} \\ y_i^{\max}(b_i^{\max} - b_i^{\text{high}}) & \text{otherwise.} \end{cases}$$

In a similar manner we can prove that

$$\triangle_i m^{\min} = \begin{cases} y_i^{\min}(b_i^{\min+1} - b_i^{\min}) & \text{if } b_i^{\min+1} \leq b_i^{\text{low}} \\ y_i^{\min}(b_i^{\text{low}} - b_i^{\min}) & \text{otherwise} \end{cases}$$

gives the increase in buffer size that leads to $b_i^{\min} = b_i^{\min+1}$, or $b_i^{\min} = b_i^{\text{low}}$ if $b_i^{\text{low}} < b_i^{\min+1}$.

We now prove that it is optimal to stop whenever $c_r - c_b r_{i^*}^{\max} \geq 0$, or equivalently when $r_{i^*}^{\max} \leq \frac{c_r}{c_b}$. First, notice that $\triangle b^{\max} = -\triangle m r_{i^*}^{\max}$, for all $\triangle m \leq \min_i\{\gamma_i, \triangle_i m^{\max}, \triangle_i m^{\min}, \varepsilon_i\}$. So the change in the objective function equals $-c_b\triangle m r_{i^*}^{\max} + c_r\triangle m$. A small increase in m_r is only interesting if this change is negative, so if $\triangle m(-c_b r_{i^*}^{\max} + c_r) < 0$, or, as $\triangle m > 0$, if $c_r - c_b r_{i^*}^{\max} < 0$. Thus,

[2] Under the assumption that no other $P(t)$-critical point emerges.

if $c_r - c_b r_{i^*}^{\max} \geq 0$ then a small increase in m_r does not lead to an improvement of the solution.

Now we show that $r_{i^*}^{\max}$ never increases during the execution of the algorithm, thereby proving that it is optimal to stop as soon as $c_r - c_b r_{i^*}^{\max} \geq 0$. The algorithm starts with the interval V_i with the maximum transmission rate $b^{\max}$ and which has the lowest $r_i^{\max}$ (of all intervals with $b^{\max}$). Now $b^{\max}$ can only shift to another interval if that interval has a lower $r_i^{\max}$. Furthermore, during the execution of the algorithm each $y_i^{\max}$ does not decrease, therefore each $r_i^{\max}$ does not increase. This means that $r_{i^*}^{\max}$ does not increase during the execution of the algorithm. This ends our proof of the correctness of the algorithm.

Time complexity

We finish by discussing the time complexity of the presented algorithm. The initialization consists of one call to RCBS and one to MVBA, and thus has time complexity $\mathcal{O}(T)$. Also, the initialization of the trade-off has time complexity $\mathcal{O}(T)$ since each time unit has to be considered when determining the intervals and several of their parameters such as $b_i^{\max}$ and b_i^{high}. The time complexity of the trade-off loop depends on the amount of events that take place. $b^{\max}$ can only shift to another interval with a lower $r_i^{\max}$, and it can only shift back to an interval when its $r_i^{\max}$ has changed, which can happen only when a convex critical point disappears or emerges. Denoting the initial number of intervals with K, this means the former event can take place at most $K+T$ times. The actions to adjust the parameters after this event have time complexity $\mathcal{O}(K)$. At each time unit at most one convex critical point may emerge or disappear, thus the second event can happen at most T times. Also for this event, the time complexity of the actions to adjust the parameters is $\mathcal{O}(K)$. Finally, since there are K intervals, at most $K-1$ times two intervals may merge. Adjusting the parameters after this event, however, has time complexity $\mathcal{O}(T)$ since for the new interval an MVBA schedule has to be determined. The total time complexity of the trade-off loop thus becomes $\mathcal{O}(K+T)\cdot\mathcal{O}(K)+T\cdot\mathcal{O}(K)+K\cdot\mathcal{O}(T)=\mathcal{O}(KT)$. Therefore, the time complexity of the total algorithm is $\mathcal{O}(KT)$.

2.2.5 Trade-off with two buffers

When all cost coefficients are positive, the sub-problem is solved by making a trade-off between the bandwidth and both buffer shares. In this section we describe how we solve this trade-off. We assume that $c_r \geq c_s$; the case with $c_s > c_r$ can be solved analogously. Notice again that this case cannot occur when either supply or demand is controllable.

Recall that for any given bandwidth b, we can determine optimal values of m_s and m_r by means of Algorithm 1 on page 25. This algorithm typically takes

only a fraction of a second to compute m_s and m_r. We use it to reformulate the sub-problem with only positive cost coefficients as follows.

Let h be a function on bandwidth b with function values that represent the minimum total costs. Now, let $m_s(b)$ and $m_r(b)$ be the optimal values of m_s and m_r, respectively, for a given b and cost coefficients c_s and c_r. Then h returns the total costs $c_b b + c_s m_s(b) + c_r m_r(b)$. The sub-problem for this case is then equivalent to finding the minimum of h. To obtain the results as fast as possible, we also want to minimize the number of function evaluations needed to find the minimum of h.

Since for a given b the problem can be formulated as an LP problem with b appearing in the right-hand-sides of the constraints, we know that h is a convex, piecewise-linear function [Roos et al., 1997]. To search the minimum of a one-dimensional function which has exactly one stationary value (a so-called unimodal function), the golden section method [Gill et al., 1988] has the best guaranteed performance. Basically, it reduces the *interval of uncertainty* in which the point with the minimum can be, by a constant factor $\kappa = (\sqrt{5}-1)/2$ after each function evaluation. However, it does not exploit the convexity property when choosing a point for function evaluation. Therefore, we next show how convexity can be used to decrease the interval of uncertainty even further.

Figure 2.11(a) gives an example of a convex function h for which six function evaluations are known. As h is convex, $\alpha h(x) + (1-\alpha)h(y) \geq h(\alpha x + (1-\alpha)y)$ for all $\alpha \in [0,1]$, and x and y in the domain of h. Using this property we obtain the piecewise-linear upper bound of h given in Figure 2.11(b). Now, we consider the line segments BC and DE and extend them until they intersect at point K as shown in Figure 2.11(c). Then, the line segments CK and KD give a lower bound for the function h between C and D, again because of the convexity of h. This can be done for any four consecutive points, resulting in the lower bound on h given in Figure 2.11(c) by the dashed lines. Finally, combining these upper and lower bounds with the lowest function value found so far gives the area in which the minimum of h must be located. This is given in an enlarged view in Figure 2.11(d) by the gray areas. The interval of uncertainty that remains after applying the convexity property is given by $[L,U]$.

Using the above approach, Den Boef & Den Hertog [2004] describe two methods to determine the minimum of a univariate convex black-box function, viz. the *improved golden section method* and the *triangle section method*. We describe both methods in this thesis, beginning with the improved golden section method.

Improved golden section method.

The improved golden section method is basically the golden section method improved with the reduction that follows from the convexity property, as described above. The golden section method chooses new points for function evaluation in

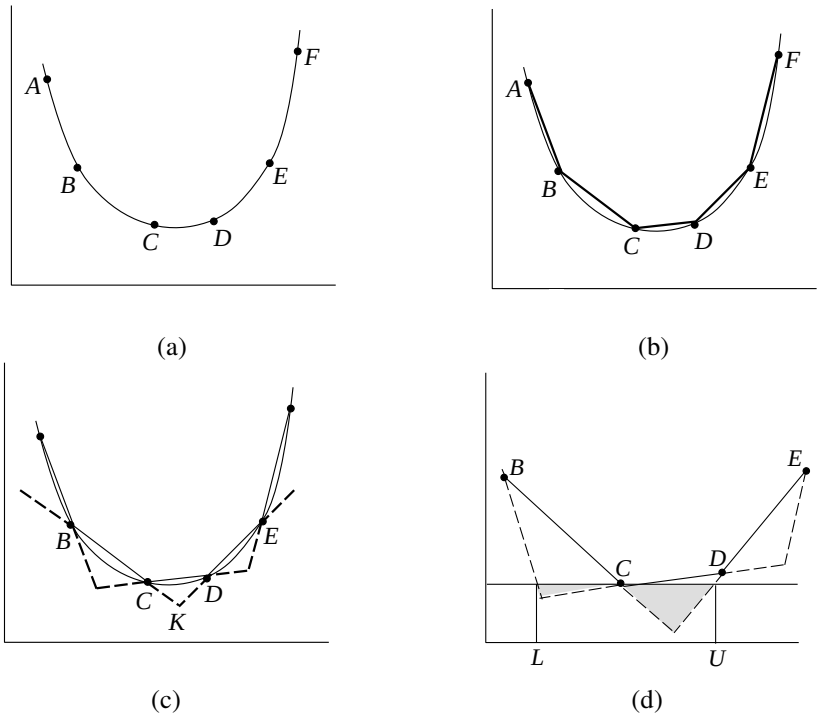

Figure 2.11. (a) Example of a convex function h with six function evaluations. (b) A piecewise-linear upper bound based on the convexity property. (c) A piecewise-linear lower bound based on the convexity property. (d) (Zoomed in compared to (a)–(c)) The optimum lies somewhere in the gray areas. The interval of uncertainty obtained using the convexity property is given by $[L,U]$, whereas the golden section method gives the interval between B and D.

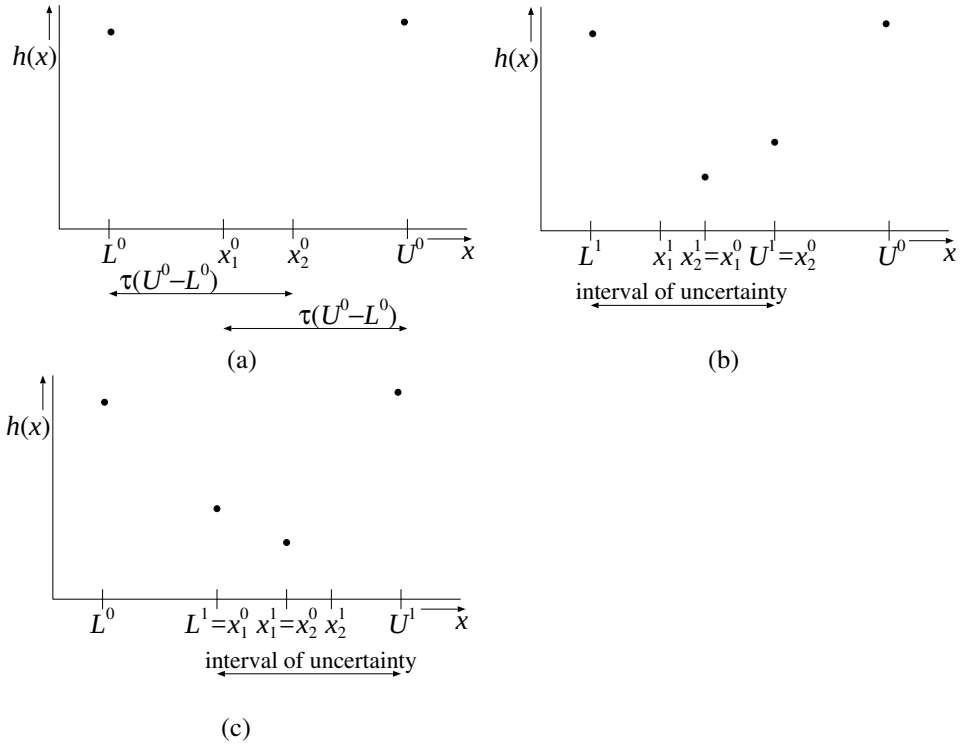

Figure 2.12. Starting the golden section method. (a) First two points, x_1^0 and x_2^0, are chosen in the interior of the starting interval of uncertainty, $[L^0, U^0]$. Next, these are evaluated, and depending on which has the minimum, the interval of uncertainty is adjusted; see (b) and (c).

such a way that the interval of uncertainty can be decreased by a constant factor $\kappa = (\sqrt{5}-1)/2$ in each iteration. Let $[L^0, U^0]$ be the initial interval of uncertainty. Then golden section chooses the following two interior points x_1^0 and x_2^0 for evaluation: $x_1^0 = U^0 - \kappa(U^0 - L^0)$ and $x_2^0 = L^0 + \kappa(U^0 - L^0)$. Now, suppose x_1^0 has the lowest function evaluation. Then the new interval of uncertainty $[L^1, U^1]$ is equal to $[L^0, x_2^0]$. Furthermore, the new interior points to be evaluated are $x_1^1 = U^1 - \kappa(U^1 - L^1)$ and $x_2^1 = L^1 + \kappa(U^1 - L^1)$. As x_2^1 is the same point as x_1^0, only x_1^1 has to be evaluated for the next step. Similarly, if x_2^0 has the lowest function evaluation, then $L^1 = x_1^0$ and $x_1^1 = x_2^0$. Figure 2.12 shows an example.

The improved golden section method now works as follows. Let $[L, U]$ be the interval of uncertainty with two interior points x_1 and x_2 such that $x_2 - L = U - x_1 = \kappa(U - L)$. We assume that $h(x_1) \leq h(x_2)$; if $h(x_2) < h(x_1)$ a strategy analogous to what we describe here can be followed. The new interval of uncertainty using the

golden section method is now given by $[L, x_2]$. Using the convexity property we can obtain a smaller interval of uncertainty $[L', U']$ for which $L' \geq L$ and $U' \leq x_2$ holds.

Golden section would choose a new point $x_3 = x_2 - \kappa(x_2 - L)$ so that $x_2 - x_3 = x_1 - L$. However, if we replace $[L, x_2]$ by $[L', U']$, and then choose $x_3 = U' - \kappa(U' - L')$, $U' - x_3$ is generally not equal to $x_1 - L'$. This means the golden section property does not hold for $[L', U']$ with interior points x_1 and x_3. Thus, we cannot guarantee that with new point x_3 the interval is again reduced by at least a factor κ. Therefore, we stretch the interval $[L', U']$ to a new interval $[\tilde{L}, \tilde{U}]$ such that the golden section property can be maintained for the new point to evaluate. We distinguish four possibilities for this:

(a) $x_1 \leq U' - \kappa(U' - L')$,

(b) $U' - \kappa(U' - L') < x_1 < (U' + L')/2$,

(c) $(U' + L')/2 \leq x_1 < L' + \kappa(U' - L')$,

(d) $x_1 \geq L' + \kappa(U' - L')$.

The corresponding stretched intervals are the following:

(a) $\tilde{L} = U' - (U' - x_1)/\kappa$, $\tilde{U} = U'$,

(b) $\tilde{L} = L'$, $\tilde{U} = (x_1 - \kappa L')/(1 - \kappa)$,

(c) $\tilde{L} = (x_1 - \kappa U')/(1 - \kappa)$, $\tilde{U} = U'$,

(d) $\tilde{L} = L'$, $\tilde{U} = L' + (x_1 - L')/\kappa$.

Figure 2.13 shows an example of these four possibilities. Den Boef & Den Hertog [2004] show that the obtained stretched interval is not larger than the interval of uncertainty according to the regular golden section method.

This leads to the following strategy for choosing a new point for evaluation.

Improved golden section strategy, $h(x_1) \leq h(x_2)$. Determine the stretched interval $[\tilde{L}, \tilde{U}]$ as described above. Choose the new point x_3 for function evaluation as follows for the four previously distinguished possibilities.

(a) $x_1 \leq U' - \kappa(U' - L')$ $x_3 = \tilde{L} + \kappa(\tilde{U} - \tilde{L}) = U' - \kappa(U' - x_1)$.

(b) $U' - \kappa(U' - L') < x_1 < \frac{1}{2}(L' + U')$ $x_3 = \tilde{L} + \kappa(\tilde{U} - \tilde{L}) = \frac{1}{\kappa}x_1 - \kappa L'$.

(c) $\frac{1}{2}(L' + U') \leq x_1 < L' + \kappa(U' - L')$ $x_3 = \tilde{U} - \kappa(\tilde{U} - \tilde{L}) = \frac{1}{\kappa}x_1 - \kappa U'$.

(d) $x_1 \geq L' + \kappa(U' - L')$ $x_3 = \tilde{U} - \kappa(\tilde{U} - \tilde{L}) = L' + \kappa(x_1 - L')$.

In case $h(x_2) < h(x_1)$ we have the following strategy.

Improved golden section strategy, $h(x_2) < h(x_1)$. Determine the stretched interval $[\tilde{L}, \tilde{U}]$. Choose the new point x_3 for function evaluation as follows.

(a) $x_2 \geq L' + \kappa(U' - L')$ $x_3 = L' + \kappa(x_2 - L')$.

(b) $\frac{1}{2}(L' + U') \leq x_2 < L' + \kappa(U' - L')$ $x_3 = \frac{1}{\kappa}x_2 - \kappa U'$.

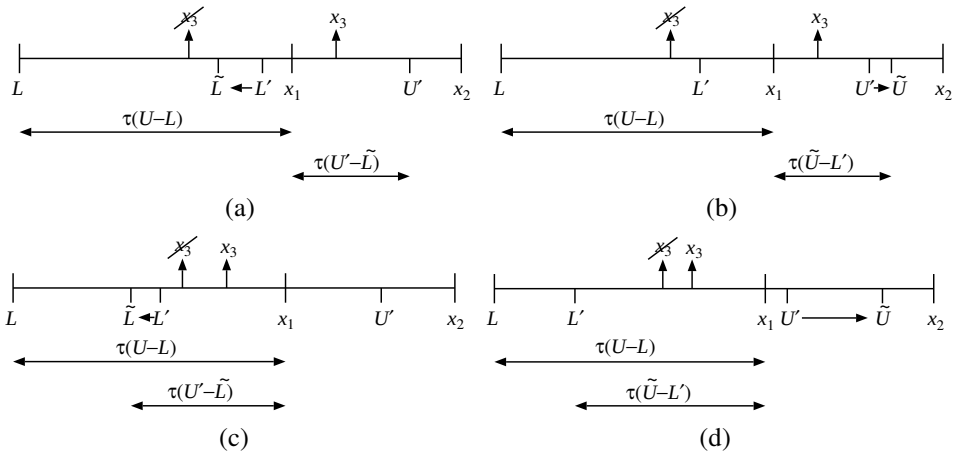

Figure 2.13. Stretching the interval of uncertainty obtained with the convexity property such that the golden section property is maintained for the new function evaluation. The four figures correspond to the four different possibilities.

(c) $U' - \kappa(U' - L') < x_2 < \frac{1}{2}(L' + U')$ $\quad x_3 = \frac{1}{\kappa}x_2 - \kappa L'.$

(d) $x_2 \leq U' - \kappa(U' - L')$ $\quad x_3 = U' - \kappa(U' - x_2).$

When both x_1 and x_2 are not interior points of the new interval of uncertainty given by $[L', U']$, two new points can be chosen inside $[L', U']$ that comply with the golden section property. Den Boef & Den Hertog prove that the improved golden section strategy performs at least as good as the normal golden section method, i.e., the improved golden section method requires at most the same number of function evaluations as the normal golden section method to reduce the interval of uncertainty to a given size. Furthermore, they show that the improved golden section method ensures that the new point chosen for function evaluation lies in the interval of uncertainty. Den Boef & Den Hertog also compare the actual performance of the improved golden section method to the performance of the normal golden section method; see also Figure 2.14. Using 16 different video streams and for each video stream a varying set of cost coefficients, both methods were tested on a total of 283 instances. The observed relative decrease in number of function evaluations required when improved golden section is used instead of normal golden section ranges from approximately 10% to as high as 80%.

Triangle section method.

The second method that can be used to find the minimum of a univariate convex black-box function is the triangle section method. In contrast to the improved golden section method its objective is to give a guaranteed reduction of the *range*

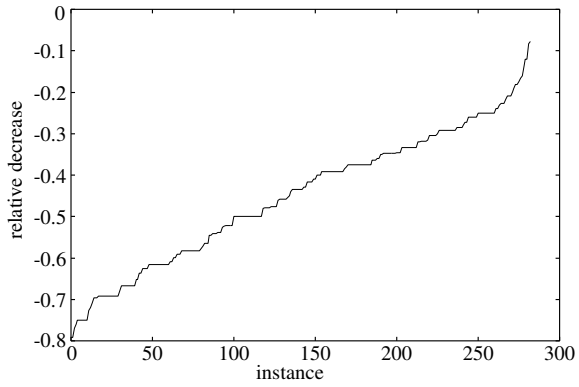

Figure 2.14. In this graph the number of function evaluations that the improved golden section method and the normal golden section method require are compared. Both methods were stopped when the interval of uncertainty was less than 1. The graph gives for 283 instances composed out of 16 different video traces, each with a number of different cost coefficients, the relative decrease in the number of function evaluations required by improved golden section compared to normal golden section, i.e., for each instance $(igs - gs)/gs$ is given, with *igs* and *gs* the number of function evaluations required for improved golden section and normal golden section, respectively.

of uncertainty, i.e., the interval of possible values of the minimum function value. Figure 2.15 depicts the area in which the optimum can be, together with the points corresponding to the function evaluations and the interval of uncertainty.

Let M be the point with the lowest function evaluation so far, $f(M)$. Then $L' \leq M \leq U'$. Now we define Δh_1^k as the height of the triangle in the area of uncertainty between L' and M after k function evaluations, and Δh_2^k as the height of the triangle between M and U'. The size of the range of uncertainty is now given by the maximum height of the area of uncertainty, i.e., $\max\{\Delta h_1^k, \Delta h_2^k\}$. The range of uncertainty itself is given by $[h(M) - \max\{\Delta h_1^k, \Delta h_2^k\}, h(M)]$. The triangle section method now chooses for function evaluation the point x that lies in the middle of the area with the largest height, i.e.,

$$x = \begin{cases} \frac{1}{2}(L' + M) & \text{if} \quad \Delta h_1^k \geq \Delta h_2^k \,, \\ \frac{1}{2}(M + U') & \text{if} \quad \Delta h_1^k < \Delta h_2^k \,. \end{cases} \tag{2.16}$$

Den Boef & Den Hertog [2004] prove that using the triangle section method the size of the range of uncertainty at least halves after every two new function evaluations.

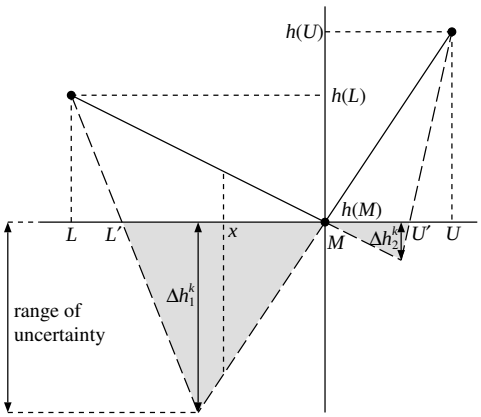

Figure 2.15. The areas of uncertainty. The three points with their function evaluations are given by L, M, and U. The interval of uncertainty in the function domain begins at L' and ends at U'. The height of the two areas after k function evaluations is given by Δh_1^k and Δh_2^k. The new point for function evaluation using the triangle section method is x.

Finite termination.

A final characteristic that we exploit is the fact that for the presented problem the function h is piecewise linear. This property can be used as follows to terminate the improved golden section method or the triangle section method in a finite number of steps with the exact minimum as result. For a piecewise-linear function the slope of a line segment will be identified as soon as three function evaluations are made of points on the segment. Furthermore, the optimum lies at the intersection of two segments. When these two line segments have been identified, the exact minimum can be obtained by determining the intersection of the two line segments.

We use this fact as follows. When a line segment containing the point with the lowest function evaluation so far has been identified, i.e., when three adjacent points lie on one line segment where the lowest function evaluation is either the first or the last point, then we evaluate the point x_k with the lowest lower bound value, i.e., $x_k = \arg\min h^{\mathrm{l}}(x)$, with h^{l} the lower bound on the function values; see Figure 2.16 for an example. Both the improved golden section method and the triangle section method can be adjusted to incorporate this strategy. For the improved golden section method it is then required to check before each new function evaluation whether the point with the lowest function evaluation lies on one line segment with either the two points immediately before or the two points immediately after itself. For the triangle section method either $\Delta h_1^k = 0$ or $\Delta h_2^k = 0$ holds in this case. Although $\Delta h_1^k = 0$ or $\Delta h_2^k = 0$ also holds when there are two points with the low-

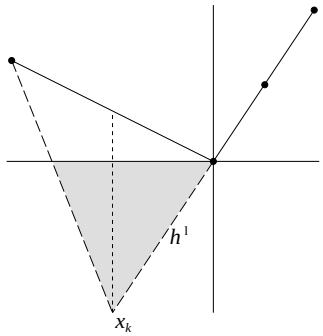

Figure 2.16. A line segment containing the point with the lowest function evaluation so far has been identified by three adjacent function evaluations. The next point x_k that is chosen for function evaluation is the point with the lowest lower bound value.

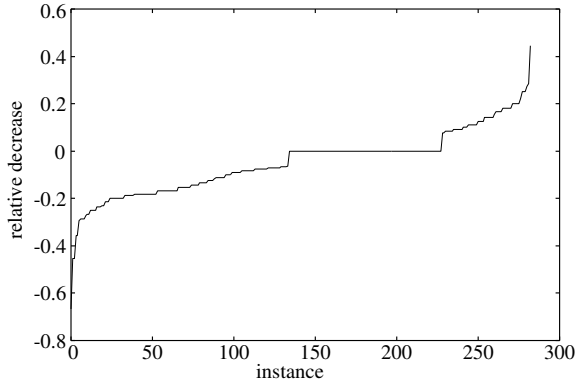

Figure 2.17. In this graph the number of function evaluations that the improved golden section method and the triangle section method require are compared. Both use the procedure that uses piecewise linearity to determine the exact optimum. The graph gives for 283 instances composed out of 16 different video traces, each with a number of different cost coefficients, the relative decrease in the number of function evaluations required by triangle section compared to improved golden section, i.e., for each instance $(ts - igs)/igs$ is given, with ts and igs the number of function evaluations required for triangle section and improved golden section, respectively.

est function evaluation, we use it to determine when to deviate from (2.16) and to evaluate $x_k = \arg\min h^1(x)$ instead. If $h(x_k) = h^1(x_k)$, then x_k is the point with the minimum and we are done. Otherwise, we continue with new function evaluations, also using the latest function evaluation $h(x_k)$.

Den Boef & Den Hertog [2004] compare the improved golden section method to the triangle section method, both using the above-described procedure for finding the exact minimum of a piecewise-linear convex function; see also Figure 2.17. Using 16 different video streams and for each video stream a varying set of cost coefficients, both methods were tested on a total of 283 instances. It follows that for approximately 50% of the instances the triangle section method needed fewer function evaluations than the improved golden section method, and that for another 30% of the instances, both methods required the same number of function evaluations. As the triangle section method for most instances performed better than the improved golden section method, we use it in the experimental results of Section 2.3.

The time complexity of the presented algorithms is $\mathcal{O}(T)$ for each call to Algorithm 1. Searching the optimum costs $\mathcal{O}(\log B/\varepsilon)$ calls for the improved golden section method and $\mathcal{O}(\log(c_b B + c_s M_s + c_r M_r/\varepsilon)$ calls for the triangle section method, where ε denotes the minimum size of the interval or range in which the optimum can be, when the algorithm stops. As the final resulting bandwidth and buffer shares should be rounded upwards to integer values for the final solution of MSSP, we can choose ε equal to 1 for the improved golden section method, leading to a time complexity $\mathcal{O}(T \log B)$. For the triangle section method, we can choose the size of ε to be linearly dependent on c_b, c_s, and c_r, giving a time complexity $\mathcal{O}(T \log(B + M_s + M_r))$.

2.3 Results

In the previous sections we described a method to solve MSSP. It determines whether a feasible solution exists, consisting of bandwidth and buffer shares and transmission schedules, for a given set of streams. When it is used for admission control of one or more streams, the run time of the method is an important parameter. Since the method is exact, i.e., it finds a solution if and only if one exists, effectiveness of the method is not an issue to be tested. We have performed experiments to study the run time and the behavior of the method, e.g., how often each sub-problem is solved. We also considered three strategies for determining an order in which the sub-problems of the different streams are solved. Finally, we present some results concerning the utilization of the resulting bandwidth and buffer shares. We first describe the setting and data used in Section 2.3.1. Then, in Section 2.3.2, we present and discuss results.

2.3.1 Experiment setting

For the experiments we considered a bus with six nodes and buffers connected to it. For the streams we have taken some MPEG-2-encoded movies, documentaries

	type	*T*	*send.*	*rec.*	*avg. frame*	*largest frame*
1	fantasy	256,599	1	2	22,859.6	138,476
2	comedy	161,082	3	4	23,359.7	126,579
3	documentary	76,750	4	1	32,613.6	143,482
4	scifi	196,160	5	6	23,359.2	120,377
5	pop concert	99,785	2	3	27,352.6	111,372
6	thriller	141,800	4	5	24,851.0	129,617
7	comedy	74,931	6	1	32,752.3	154,283
8	action	151,132	6	2	31,114.1	131,713
9	documentary	36,875	1	3	26,829.5	114,601
10	action	184,371	5	3	22,112.5	111,101
11	pop concert	166,765	2	4	24,631.5	143,871
12	comedy	164,199	4	6	21,318.4	122,383
13	comedy	38,215	1	5	33,946.5	145,693
14	action	165,141	3	6	26,418.6	139,284
15	documentary	75,100	5	2	24,717.6	140,321

Table 2.1. The 15 streams used in the experiment. The length of each stream is given, as well as its sending (*send.*) and receiving (*rec.*) node in the experiment setting. Furthermore, the average frame size (bytes) and largest frame size (bytes) of each stream are given.

	buffer sizes (M)				
	512,000	1,024,000	4,096,000	8,192,000	16,384,000
delay 10, $B = 125,000$	1	-	-	-	-
delay 10, $B = 500,000$	2	3	4	-	-
delay 100, $B = 125,000$	-	-	5	-	-
delay 100, $B = 500,000$	-	-	6	7	8

Table 2.2. Combinations of values of the delay (in time units), the bandwidth (in bytes/time unit), and the buffer sizes (in bytes) for which experiments have been performed, leading to a total of eight settings, denoted by the numbers '1'–'8'.

and concerts, and derived the traces with the frame sizes. Table 2.1 shows for each trace used for the experiments, its length, its sending node, and its receiving node, and its average and largest frame size. The length of a time unit is given by the inter-frame time, i.e., 1/25 s. Furthermore, the supply and demand schemes correspond to the frame sizes of each trace. The delay of a stream, i.e., the time between the start of the supply scheme and the start of the demand scheme, was set at a fixed value for each experiment. For an experiment with x streams, we used the first x streams shown in Table 2.1, i.e., if $x = 10$, then streams 1–10 are used.

Several different settings have been used for the experiments as follows. We took different values for the bus and buffer capacities and for the stream delays, given in Table 2.2. Furthermore, we increased the number of streams on the network starting with one stream, until no feasible solution existed for the chosen experiment setting.

Finally, the Dantzig-Wolfe decomposition allows several strategies for deter-

mining an order in which sub-problems for different streams are solved in each iteration, i.e., for each new set of cost coefficients given by the master problem. We have considered the following three in particular.

Strategy 1. We start with the latest stream for which a sub-problem was solved in the previous iteration. If no solution exists for this stream that improves the master problem, we continue with the next stream, wrapping around to the first stream again after the last stream.

Strategy 2. The sub-problems for all streams are solved, and all solutions that improve the master problem are returned.

Strategy 3. We start with the least recently considered stream. If no solution exists for this stream that improves the master problem, we continue with the next stream, wrapping around to the first stream again after the last stream.

Using strategy 1, the algorithm first searches exhaustively for a good solution for one stream before continuing with the next stream. However, this may lead to an unnecessarily high number of sub-problems solved and thus to an unnecessarily long run time of the algorithm, as it tries to improve the solution for a stream to a very detailed level that may not be necessary to find a feasible solution for the overall problem. Using strategy 2, detailed improvements for the solution of a stream are only searched when no feasible solution for the overall problem has been found yet. However, the optimal solution for a stream may be affected by a solution that is used for another stream as they both share the bandwidth and possibly one or two buffers. Therefore, solutions generated for a stream using this strategy may become obsolete immediately during the optimization of the master problem when a generated solution for another stream is used. Strategy 3 takes into account that an other stream's solution may affect the optimal solution of a stream, as it obtains a solution for all streams as soon as possible.

The algorithm terminates when no stream gives an improving solution anymore. For strategies 1 and 3 this means that in the last iteration the algorithm solves a sub-problem for each stream.

2.3.2 Experimental results

The experiments have been executed on a PC equipped with an AMD XP 1800+ processor and 512 MB RAM, running Windows XP. We first discuss some feasibility results and the corresponding run times of the solution method. Figure 2.18 shows the run time in seconds and the total number of sub-problem instances that were solved for setting 3 which has delay 10, bandwidth $500,000$, and buffer sizes $1,024,000$, as a function of the number of streams. The largest number of streams for which results are shown in the figure, i.e., 11 streams, is the first number of streams for which no feasible solution exists for this setting.

Table 2.3 shows for all settings the maximum number of streams for which a

	max. #	*strategy 1 run time*		*strategy 2 run time*		*strategy 3 run time*	
setting	*streams*	*avg. feas.*	*1st inf.*	*avg. feas.*	*1st inf.*	*avg. feas.*	*1st inf.*
1	3	0.875	3.672	0.943	3.813	0.958	3.469
2	4	0.926	1.672	0.934	1.719	0.758	1.937
3	10	1.703	7.079	1.955	6.079	1.731	6.484
4	13	2.581	10.516	2.378	12.156	2.064	9.594
5	3	0.839	6.141	0.859	3.407	0.917	4.172
6	4	0.961	2.281	1.039	3.047	0.801	3.344
7	12	3.116	9.532	2.685	6.422	2.624	6.812
8	14	3.547	13.500	2.482	12.015	2.637	13.687

Table 2.3. For all eight settings this table gives the maximum number of streams (*max.*) for which a feasible solution exists. Furthermore, it gives for each strategy the run time to determine a feasible solution averaged over all numbers of streams for which a feasible solution exists (*avg. feas.*). It also gives the run time for the first number of streams which is infeasible (*1st inf.*), i.e., the maximum number of streams plus one.

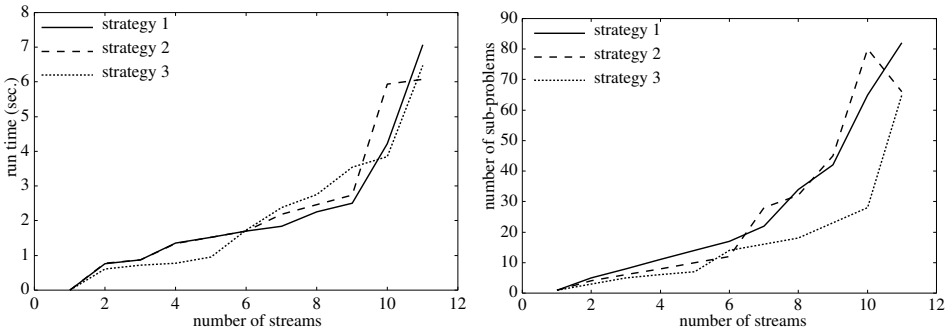

Figure 2.18. Results for setting 3, which has delay 10, bandwidth 500,000, and buffer sizes 1,024,000. The left graph shows the run times in seconds for the three strategies, and the right graph the number of sub-problems that have been solved. The results are shown up to the first number of streams for which no feasible solution exists, in this case eleven streams.

feasible solution exists. Furthermore, it shows for each strategy the average run time to compute a feasible solution, where the average is taken over all numbers of streams for which a feasible solution exists for the setting. It also shows for each strategy the run time for the lowest number of streams for which a feasible solution does not exist.

As we can see in Figure 2.18, the increase in run time is more than linear in the number of streams for setting 3, which we also observed for the other settings. This is mainly caused by the fact that the number of sub-problems solved also increases more than linearly in the number of streams, i.e., more instances per stream are solved. Furthermore, Figure 2.18 shows that the strategy with the shortest run time

	strategy 1		strategy 2		strategy 3	
sub-problem	# times	run time	# times	run time	# times	run time
$c_b, c_r, c_s \leq 0$	5	0	4	0	1	0
$c_b > 0$	4	0.063	3	0.157	4	0.094
$c_r > 0$	0	0	0	0	0	0
$c_s > 0$	0	0	0	0	0	0
$c_r, c_s > 0$	0	0	0	0	0	0
$c_b, c_r > 0$	5	0.282	6	0.093	8	0.312
$c_b, c_s > 0$	6	0.688	17	0.641	8	0.344
$c_b, c_r, c_s > 0$	25	5.108	14	2.516	18	3.422
total	45	6.141	44	3.407	39	4.172

Table 2.4. For setting 5, which has delay 100, bandwidth 125,000, and buffer sizes 4,096,000, and with streams 1–4, this table gives the number of times each case of the sub-problems has been solved, and the total run time for solving each case.

depends on the number of streams involved, e.g., in this figure for some numbers of streams strategy 3 has the shortest run time, while for some other numbers of streams strategy 1 has the shortest run time. Likewise, Table 2.3 shows that the strategy with the lowest average run time, or the shortest run time for the first infeasible number of streams, depends on the setting. None of the three strategies consistently outperformed one of the other strategies for all settings. Therefore, none of them is the best regarding the run times. However, the differences between the strategies are very small so it does not really matter which strategy is used.

We now continue with results concerning the behavior of the solution method. Table 2.4 shows for one setting how many times a call has been made to each sub-problem and the time spent solving each sub-problem. Table 2.5 shows the same results but now accumulated over all settings and numbers of streams. For all settings, the time used for optimizing the master problem is almost zero.

Table 2.4 shows that for setting 5 with four streams, none of the buffer minimizing sub-problems needed to be solved, i.e., the sub-problems with $c_b \leq 0$, and $c_s > 0$ or $c_r > 0$ or both. As for setting 5 the buffer sizes are not a bottleneck but the bandwidth is, only the sub-problems with $c_b > 0$ have been solved to find a feasible solution. When all settings are considered, the buffer minimizing sub-problems have been solved several times as is shown in Table 2.5. However, it is remarkable that for strategy 3 the sub-problems with either only $c_r > 0$ or only $c_s > 0$ are barely solved, only four times in total.

Table 2.5 also shows that for all strategies the case with $c_b, c_r, c_s > 0$ required a large fraction of the total run time. Furthermore, overall strategy 3 has solved the fewest sub-problems and has the lowest total run time, while strategy 1 has solved the most sub-problems and has the highest total run time. However, where the differences in number of sub-problems solved are large, the differences in run

	strategy 1		*strategy 2*		*strategy 3*	
sub-problem	*# times*	*run time*	*# times*	*run time*	*# times*	*run time*
$c_b, c_r, c_s \leq 0$	702	0	474	0	81	0
$c_b > 0$	134	2.796	156	5.203	178	5.814
$c_r > 0$	44	0.434	35	0.489	4	0.047
$c_s > 0$	42	0.205	24	0.246	0	0
$c_r, c_s > 0$	77	1.000	36	0.593	58	1.597
$c_b, c_r > 0$	134	15.373	94	8.468	116	10.556
$c_b, c_s > 0$	198	18.557	112	9.639	73	7.923
$c_b, c_r, c_s > 0$	916	166.158	787	154.666	707	147.923
total	2,247	204.523	1,718	179.304	1,217	173.860

Table 2.5. The total number of instances of each sub-problem that has been solved for each strategy and the total run time involved, accumulated over all settings and all numbers of streams that were considered.

		Total utilization		*Min. utilization*		*Max. utilization*	
setting	*streams*	U_B	U_M	$U_B^{\min}$	$U_M^{\min}$	$U_B^{\max}$	$U_M^{\max}$
1	3	62.12	44.43	57.17	40.54	68.10	53.78
2	4	55.79	40.95	45.29	35.68	71.33	55.80
3	10	58.81	47.28	43.10	36.55	77.69	58.24
4	13	68.13	37.55	58.08	30.26	84.42	53.85
5	3	63.80	49.87	57.15	39.67	73.66	60.73
6	4	59.80	52.10	47.14	39.30	78.60	68.80
7	12	62.71	62.46	48.15	51.24	82.12	79.33
8	14	74.88	42.34	65.54	34.33	87.76	63.49

Table 2.6. The utilization of the reserved bandwidth (bus) and buffer shares in percentages. For each setting (first column) we considered the results for the maximum number of admitted streams (second column), all using strategy 3 for considering the sub-problems. The third and fourth column give the total utilization over all streams for the reserved bandwidth and buffer shares, U_B and U_M, respectively. The fifth and sixth column give the minimum utilization, $U_B^{\min}$ and $U_M^{\min}$, as achieved by a single stream in the setting. The seventh and eighth column give the maximum utilization, $U_B^{\max}$ and $U_M^{\max}$, as achieved by a single stream in the setting.

time are relatively small, i.e., strategy 3 has solved over 40 % fewer sub-problems than strategy 1, but the total run time of strategy 3 is less than 20 % below the total run time of strategy 1. This is caused by the fact that the difference in number of sub-problems solved is mostly a result of the difference in the number of times the sub-problem $c_b, c_r, c_s \leq 0$ is solved. The difference in the number of times the sub-problem $c_b, c_r, c_s > 0$ is solved is much lower, while this sub-problem takes at least 80 % of the total run time for each strategy.

Finally, in Table 2.6 we consider the utilization of the obtained bandwidth and buffer shares. We used the results for the maximum number of streams for each setting with strategy 3 for selecting sub-problems. The utilization percentages are determined as follows. The total utilization of the bandwidth is defined as the

percentage of the total reserved bandwidth that is actually required for transmission by all streams, i.e., the total bandwidth utilization U_B is given by

$$U_B = \frac{\sum_{d\in\mathcal{D}}\sum_{t\in\mathcal{T}} P_d(t)}{\sum_{d\in\mathcal{D}}(T_d+\delta)b_d}\,, \tag{2.17}$$

where T_d is the length of a stream as given in Table 2.1 and δ the delay. The total utilization of the buffer shares is defined as the percentage of the buffer shares that is actually used for buffering data by all streams, i.e., the total buffer utilization U_M is given by

$$U_M = \frac{\sum_{d\in\mathcal{D}}\sum_{t\in\mathcal{T}} P_d(t)(\delta+1)}{\sum_{d\in\mathcal{D}}(T_d+\delta)(m_{s_d,d}+m_{r_d,d})}\,. \tag{2.18}$$

The total data supply of a stream is multiplied by $\delta+1$ since all data remain in the network for δ time units and during the time unit in which data is transmitted it needs to be buffered in both the sending and the receiving buffer. Furthermore, we use the fact that in our experiments the demand of data equals the supply of data with a difference of δ time units, i.e., $C_d(t) = P_d(t-\delta)$.

The utilization of the bandwidth for a single stream is defined as the percentage of the reserved bandwidth share that is required for transmission, i.e., for each $d \in \mathcal{D}$, the bandwidth utilization $U_B(d)$ is given by

$$U_B(d) = \frac{\sum_{t\in\mathcal{T}} P_d(t)}{(T_d+\delta)b_d}\,. \tag{2.19}$$

The minimum and maximum utilization of the bandwidth $U_B^{\min}$ and $U_B^{\max}$ give the minimum and maximum of the bandwidth utilization, respectively, over all streams, i.e.,

$$U_B^{\min} = \min_{d\in\mathcal{D}} U_B(d)\,, \tag{2.20}$$

and

$$U_B^{\max} = \max_{d\in\mathcal{D}} U_B(d)\,. \tag{2.21}$$

Similarly, the utilization of the buffers for a single stream is defined as the percentage of the reserved buffer shares that is actually used for buffering data, i.e., for each $d \in \mathcal{D}$, the buffer utilization $U_M(d)$ is given by

$$U_M(d) = \frac{\sum_{t\in\mathcal{T}} P_d(t)(\delta+1)}{(T_d+\delta)(m_{s_d,d}+m_{r_d,d})}\,. \tag{2.22}$$

The minimum and maximum utilization of the buffers $U_M^{\min}$ and $U_M^{\max}$ give the minimum and maximum of the buffer utilization, respectively, over all streams, i.e.,

$$U_M^{\min} = \min_{d\in\mathcal{D}} U_M(d)\,, \tag{2.23}$$

and

$$U_M^{\max} = \max_{d \in \mathcal{D}} U_M(d) \; . \tag{2.24}$$

The total bandwidth utilization varies between 55% and 75% for the different settings, thus giving a relatively high amount of unused bandwidth. The minimum and maximum bandwidth utilization show that for the different streams the bandwidth utilization also varies heavily for each setting, with some streams having a utilization below 50% and other streams having a utilization above 80%. Higher bandwidth utilizations might be obtained when bandwidth reservations may be altered during the run time of a stream.

If we compare the different settings, we notice that settings 4 and 8 have the highest total bandwidth utilization percentages. This can be explained by the fact that both settings also have for their given values of B and the delay the largest available buffers compared to other settings; see Table 2.2. With more buffer space available, transmission can be made smoother for more streams, thus leading to lower bandwidth reservations and higher bandwidth utilizations. At the same time it would also lead to higher buffer reservations and thus lower buffer utilizations. However, if we compare setting 2 to setting 3 and setting 6 to setting 7 we notice that the bandwidth utilization as well as the buffer utilization increase. An explanation for this observation lies in the fact that the number of streams used for the results of settings 3 and 7 are more than twice the number of streams used for the results of settings 2 and 6. Since every video stream is different, with some behaving more efficiently regarding bandwidth and buffer utilization than others, an increase in the number of streams can lead to a higher total bandwidth utilization as well as a higher total buffer utilization.

3

Leaky-bucket-controlled streams

In this chapter we consider the problem in which leaky-bucket-controlled streams need to be given shares of the bandwidth and buffers. As mentioned in Chapter 1, a leaky-bucket-controlled stream does not have a supply and demand scheme given as can be the case for a fully-specified stream in Chapter 2, but only an upper bound on the actual supply. Furthermore, unlike a fully-specified stream with controllable supply or demand, the supply and demand of a leaky-bucket-controlled stream cannot be chosen in accordance with its upper bound; the actual realization of the supply is only known when the data is supplied, but we know that it satisfies the given upper bound.

We first describe a leaky-bucket controller and its data output in Section 3.1. Then, we describe the problem for leaky-bucket-controlled streams in Section 3.2 and we show how it can be solved using the solution method for fully-specified streams. In Section 3.3 we show how the sub-problems can be efficiently solved for leaky-bucket-controlled streams. Finally, in Section 3.4 we present experimental results.

3.1 Leaky-bucket controllers

While normally the supply and demand schemes are known for pre-recorded streams, they are generally not known for live video streams. In order to give

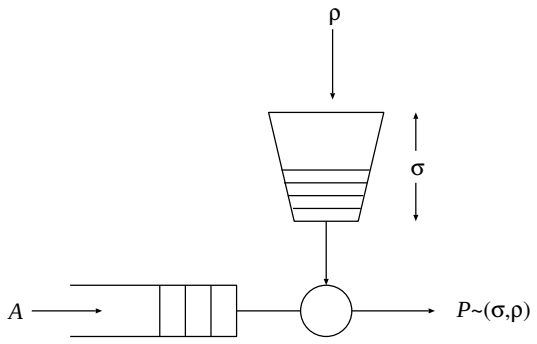

Figure 3.1. A (σ,ρ)-leaky bucket controller.

guaranteed resource reservations to live video streams we need to use traffic characterizations of the streams. These characterizations can be deterministic such as the (σ,ρ)-model [Cruz, 1991a] and the D-BIND model [Knightly & Zhang, 1995], or probabilistic such as the model by Krunz & Tripathi [1997]. As in an IHDN typically only a few streams, e.g. 5–10 streams, use the network simultaneously, we may not hope that peaks in the amount of supplied data are cancelled out over the total set of streams. Therefore, a deterministic approach of admission test for live video streams is required.

Deterministic traffic characterizations give a worst-case bound on the amount of data that is supplied during any time window. More specifically, let ξ be the size of a time window, and let $P(t)$ denote the cumulative supply of data up to time unit t. Then a traffic characterization described by a function f bounds the supply of data by

$$P(t+\xi) - P(t) \leq f(\xi)\ , \tag{3.1}$$

for all t. When f is a piecewise-linear, concave function, the traffic characterization corresponds to a stream controlled by one or more leaky buckets [Turner, 1986].

Figure 3.1 gives an example of a (σ,ρ)-leaky-bucket controller. It consists of a bucket of size σ in which tokens are generated at rate ρ. If this bucket is full, then newly generated tokens are lost. Data arrives at the controller according to an arrival process A. Each data packet requires a token from the bucket before it may pass the controller. If a token is not available for a packet, it waits in a buffer at the controller until it can get a token from the bucket. We assume that this buffer is large enough to store all data packets that need to wait for a token. The resulting data output P of the controller is characterized by (σ,ρ), also denoted as $P \sim (\sigma,\rho)$, i.e., $f(t) = \sigma + \rho t$. Here, ρ can be seen as the maximum sustainable rate of data, while σ gives the maximum burst size of data. The function $f(t) = \sigma + \rho t$ gives an upper bound on the amount of data that can leave the leaky-bucket controller

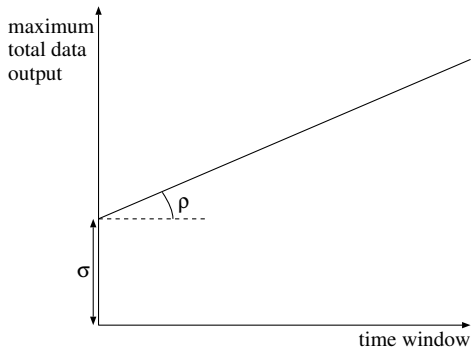

Figure 3.2. The maximum data output of a (σ,ρ)-leaky-bucket controller for all time windows larger than 0.

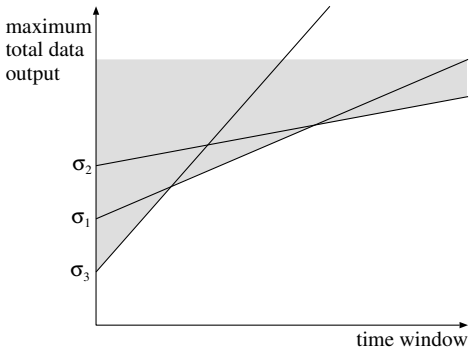

Figure 3.3. The maximum data output of a sequence of three leaky-bucket controllers, each with their own characteristics. The white area gives the possible data output for each time window.

during a time window of size t; see Figure 3.2.

Besides a single leaky-bucket controller, multiple leaky-bucket controllers placed in a sequence can also be used to control the data. Each leaky-bucket controller limits the amount of data that can pass during a certain time window. For a given time window, the controller with the lowest limit determines the amount of data that can result as output. However, the controller with the lowest limit is generally not always the same for all possible time windows. A controller with a small token buffer σ but a high token rate ρ will be more limiting for small time windows, while a controller with a large token buffer and a low token rate will be more limiting for large time windows. The upper bound on the resulting data output during a time window t is thus given by $f(t) = \min_{i\in\{1,\ldots,k\}}\{\sigma_i + \rho_i t\}$ for a sequence of k leaky-bucket controllers; see Figure 3.3 for an example. The resulting upper bound is piecewise linear and concave.

3.2 Problem

In this section we describe the bandwidth and buffer allocation problem with streams controlled by one or more leaky buckets. We start with the assumptions and notation specific for the leaky-bucket-controlled streams in Section 3.2.1. Then, we give a formal description for the problem concerning leaky-bucket-controlled streams in Section 3.2.2. In Section 3.2.3 we show that the problem concerning leaky-bucket-controlled streams can be reduced to the problem concerning fully-specified stream.

3.2.1 Model assumptions and notation

The assumptions we make for leaky-bucket-controlled streams are generally the same as the assumptions for fully-specified streams; see Section 2.1.1. The only difference lies in the supply and demand of data. We assume that for a leaky-bucket-controlled stream, the data that is supplied to the network at the sending node, is shaped by one or more leaky buckets. The supply can then be bounded by a piecewise-linear, concave function. Next, we assume that the demand of data of a leaky-bucket-controlled stream is a displaced copy of the supply function, i.e., the amount of data that is supplied during a time unit at the sending node, is the exact amount of data that is demanded at the receiving node after a given delay.

The notation we use for a leaky-bucket-controlled stream is similar to the notation used for fully-specified streams, introduced in Section 2.1.2. For ease of use we again denote the set of streams by $\mathcal{D}$. Whenever a stream $d \in \mathcal{D}$ is considered in this thesis, it should be clear from the context whether it is a fully-specified stream, a leaky-bucket-controlled stream, or whether the exact type of stream is not relevant. The new notation which we introduce here concerns the upper bound on data supply. For each leaky-bucket-controlled stream $d \in \mathcal{D}$ the supply of data during a time window of t time units, $t = 1, \ldots, T$, is bounded by $f_d(t)$ with f_d positive, piecewise-linear, and concave on $(0, T]$, and $f_d(0) = 0$. The delay is given by δ_d for each stream $d \in \mathcal{D}$. As buffer underflow and overflow depend on the actual supply and demand, we use for the leaky-bucket-controlled streams $d \in \mathcal{D}$, analogously to fully-specified streams, $p_d(t)$ as the actual supply scheme and $c_d(t)$ as the actual demand scheme for all $t \in \mathcal{T}$. Since demand equals the supply with delay δ_d, it follows that for all $t \in \mathcal{T}$,

$$c_d(t) = p_d(t - \delta_d) \; . \tag{3.2}$$

The decision variables for each leaky-bucket-controlled stream $d \in \mathcal{D}$ are similar to the decision variables of fully specified streams, i.e., the reserved bus bandwidth b_d and the reserved buffer sizes $m_{s_d,d}$ and $m_{r_d,d}$ at the sending node and at the receiving node, respectively. As the actual supply and demand are not known beforehand, we cannot give a transmission schedule $x_d(t), t \in \mathcal{T}$. However, we can give a strategy that describes $x_d(t), t \in \mathcal{T}$ for which we will use the same symbol. Thus the transmission schedule of a leaky-bucket-controlled stream is given implicitly by $x_d(t), t \in \mathcal{T}$, which we therefore call the transmission strategy. An example of a transmission strategy is given by the greedy strategy $x(t) = \min\{b, P(t) - X(t-1), m_r + C(t-1) - X(t-1)\}$, which tries to transmit data as soon as possible.

In this chapter, $P_d(t)$, $C_d(t)$, and $X_d(t)$ are also defined as the cumulative supply scheme, cumulative demand scheme, and cumulative transmission strategy, respectively. To avoid tedious formulations concerning function values at the bound-

aries of their domains, we assume that for all functions h on $\mathcal{T}$ used in this chapter, $h(t) = 0$ if $t \leq 0$, and $h(t) = h(T)$ if $t \geq T$.

3.2.2 Multiple Leaky-Bucket Streams Smoothing Problem

We can now define the following problem for a set of leaky-bucket-controlled streams.

Definition 3.1. Multiple Leaky-Bucket Streams Smoothing Problem (MLBSSP). Given are a set $\mathcal{N}$ of nodes, with for each node $n \in \mathcal{N}$ a buffer capacity M_n, a set $\mathcal{T}$ of time units, and a bandwidth B. Furthermore, a set $\mathcal{D}$ of streams is given with for each stream $d \in \mathcal{D}$ a sending node $s_d \in \mathcal{N}$ with a concave, piecewise-linear function $f_d(t)$ that bounds the supply of data during time windows of size $t = 1, 2, \ldots, T$, and a receiving node $r_d \in \mathcal{N}$ with a delay δ_d that indicates the number of time units between supply of data and demand of the same data.

Determine for all streams $d \in \mathcal{D}$ values for the bandwidth share $b_d \geq 0$ and buffer shares $m_{s_d,d}, m_{r_d,d} \geq 0$, and a transmission strategy $x_d(t)$ for all $t \in \mathcal{T}$, such that the bandwidth B and buffer capacities M_n will not be exceeded and buffer shares will neither overflow nor underflow. □

Any solution to MLBSSP must satisfy the following constraints for all actual supply schemes $p_d(t)$ that satisfy (3.1), i.e., that are bounded by $f_d(t)$ for stream d, and for all actual demand schemes $c(t)$ that satisfy (3.2), i.e., that are equal to the supply shifted by δ_d. These constraints are similar to (2.1)–(2.7) for MSSP.

The total bandwidth and total buffer sizes reserved may not exceed the bandwidth and buffer capacities, i.e.,

$$\sum_{d \in \mathcal{D}} b_d \leq B\,, \tag{3.3}$$

and,

$$\sum_{d \in \mathcal{D} \mid s_d = n} m_{n,d} + \sum_{d \in \mathcal{D} \mid r_d = n} m_{n,d} \leq M_n\,, \qquad \forall\, n \in \mathcal{N}\,. \tag{3.4}$$

The amount of data transmitted for stream d during time unit t may not exceed the reserved bandwidth share of stream d, i.e.,

$$x_d(t) \leq b_d\,, \qquad \forall\, d \in \mathcal{D}\,,\ t \in \mathcal{T}\,. \tag{3.5}$$

Furthermore, the reserved buffer shares at the sending and receiving node of each stream may not underflow nor overflow, i.e., we must have

$$P_d(t) - X_d(t) \geq 0\,, \qquad \forall\, d \in \mathcal{D}\,,\ t \in \mathcal{T}\,, \tag{3.6}$$

$$P_d(t) - X_d(t-1) \leq m_{s_d,d}\,, \qquad \forall\, d \in \mathcal{D}\,,\ t \in \mathcal{T}\,, \tag{3.7}$$

$$X_d(t) - C_d(t) \geq 0\,, \qquad \forall\, d \in \mathcal{D}\,,\ t \in \mathcal{T}\,, \tag{3.8}$$

$$X_d(t) - C_d(t-1) \leq m_{r_d,d}\,, \quad \forall\, d \in \mathcal{D}\,,\ t \in \mathcal{T}\,. \tag{3.9}$$

In the next section we show that instead of considering all such supply and demand schemes, we may use the functions f_d as supply schemes when determining an optimal solution.

3.2.3 Problem reduction

In this section we show that MLBSSP reduces to MSSP. This means that we can use the same solution method for MLBSSP as for MSSP. To show that MLBSSP reduces to MSSP, we show that we may use $f_d(t)$ as actual cumulative supply scheme $P_d(t)$ and $f_d(t-\delta_d)$ as actual cumulative demand scheme $C_d(t)$.

First, we consider the constraints involving the actual supply and demand schemes for the leaky-bucket-controlled streams, i.e., constraints (3.6)–(3.9) for all $d \in \mathcal{D}$. For ease of notation we omit the subscript d in this section as each of these constraints involve only one stream. We call a solution b, m_s, and m_r for a single leaky-bucket-controlled stream *feasible w.r.t. (3.5)–(3.9)*, if for all p and c that satisfy (3.1) and (3.2), there exists a transmission strategy for which (3.5)–(3.9) are satisfied. Instead of considering all p that satisfy (3.1) and c that satisfy (3.2), we now show that it is sufficient to consider only a worst-case supply which is given by f directly, i.e., let for all $t \in \mathcal{T}$,

$$P(t) = f(t)\,, \tag{3.10}$$
$$C(t) = f(t-\delta)\,. \tag{3.11}$$

It can be easily verified that this function P satisfies (3.1) by the concavity of f.

The feasible solutions of one stream are then described by the following constraints.

$$x(t) \leq b\,, \qquad \forall\, t \in \mathcal{T}\,, \tag{3.12}$$
$$f(t) - X(t) \geq 0\,, \qquad \forall\, t \in \mathcal{T}\,, \tag{3.13}$$
$$f(t) - X(t-1) \leq m_s\,, \qquad \forall\, t \in \mathcal{T}\,, \tag{3.14}$$
$$X(t) - f(t-\delta) \geq 0\,, \qquad \forall\, t \in \mathcal{T}\,, \tag{3.15}$$
$$X(t) - f(t-\delta-1) \leq m_r\,, \qquad \forall\, t \in \mathcal{T}\,. \tag{3.16}$$

Note that (3.12) is the same as (3.5). We call a solution b, m_s, and m_r *feasible w.r.t. (3.12)–(3.16)*, if there exists a transmission strategy for which (3.12)–(3.16) are satisfied. Next, we show that a solution b, m_s, and m_r is feasible w.r.t. (3.12)–(3.16), if and only if it is feasible w.r.t. (3.5)–(3.9).

Theorem 3.1. *A solution b, m_s, and m_r is feasible w.r.t. (3.12)–(3.16), if and only if it is feasible w.r.t. (3.5)–(3.9).*

Proof. "⇐" Let b, m_s, and m_r be a feasible solution w.r.t. (3.5)–(3.9). As $P = f$

is a possible actual supply function, it follows immediately that b, m_s, and m_r is a feasible solution w.r.t. (3.12)–(3.16).

"$\Rightarrow$" Let b, m_s, and m_r be a feasible solution w.r.t. (3.12)–(3.16) with a transmission strategy given by $x^*(t)$. Now consider the greedy transmission strategy $x(t) = \min\{b, P(t) - X(t-1), m_r + C(t-1) - X(t-1)\}$ for any actual supply P satisfying (3.1) and for any actual demand C satisfying (3.2). It follows immediately from the definition of $x(t)$ that (3.5), (3.6), and (3.9) are satisfied. So we only need to show that (3.7) and (3.8) are also satisfied.

Let $t \in \mathcal{T}$. We now choose

$$\tilde{k} = \max\{k \in \{1,\ldots,t-1\} \mid \left.\begin{array}{lcll} x(k) & = & P(k) - X(k-1) & \vee \\ x(k) & = & m_r + C(k-1) - X(k-1) & \end{array}\right\}, \quad (3.17)$$

so for all k with $\tilde{k} < k < t$, $x(k) = b$ holds. We first show that (3.7) is satisfied, i.e., $P(t) - X(t-1) \leq m_s$. We distinguish three cases, viz. (i) $\tilde{k} = -\infty$, i.e., $\tilde{k}$ is undefined, (ii) $x(\tilde{k}) = P(\tilde{k}) - X(\tilde{k}-1)$, and (iii) $x(\tilde{k}) = m_r + C(\tilde{k}-1) - X(\tilde{k}-1)$.

(i) If $\tilde{k} = -\infty$, then $X(t-1) = (t-1)b$. We derive

$$\begin{array}{rcl} P(t) - X(t-1) & = & P(t) - (t-1)b \\ \{\textit{ use (3.1)) and (3.12) for } x^*(t)\} & \leq & f(t) - X^*(t-1) \\ \{\textit{ use (3.14)}\} & \leq & m_s\,. \end{array}$$

(ii) If $x(\tilde{k}) = P(\tilde{k}) - X(\tilde{k}-1)$, then $X(\tilde{k}) = P(\tilde{k})$. We derive

$$\begin{array}{rcl} P(t) - X(t-1) & = & P(t) - X(\tilde{k}) - \sum_{k=\tilde{k}+1}^{t-1} x(k) \\ \{\, x(k) = b \textit{ for } k > \tilde{k}\} & = & P(t) - P(\tilde{k}) - (t-1-\tilde{k})b \\ \{\textit{ use (3.1)}\} & \leq & f(t-\tilde{k}) - (t-1-\tilde{k})b \\ \{\textit{ (3.12) for } x^*(t)\} & \leq & f(t-\tilde{k}) - X^*(t-\tilde{k}-1) \\ \{\textit{ use (3.14)}\} & \leq & m_s\,. \end{array}$$

(iii) If $x(\tilde{k}) = m_r + C(\tilde{k}-1) - X(\tilde{k}-1)$, then $X(\tilde{k}) = m_r + C(\tilde{k}-1)$. We derive

$$\begin{array}{rcl} P(t) - X(t-1) & = & P(t) - X(\tilde{k}) - \sum_{k=\tilde{k}+1}^{t-1} x(k) \\ \{\, x(k) = b \textit{ for } k > \tilde{k}\} & = & P(t) - m_r - C(\tilde{k}-1) - (t-1-\tilde{k})b \\ \{\textit{ use (3.2)}\} & = & P(t) - m_r - P(\tilde{k}-\delta-1) - (t-1-\tilde{k})b \end{array}$$

$$\begin{array}{rcl}
\{\textit{ use (3.1)}\} & = & f(t-(\tilde{k}-\delta-1))-m_r-(t-1-\tilde{k})b. \\
\left\{\begin{array}{l}\textit{(3.16) for } X^*(\delta+1)\\ \textit{and } f(0)=0\end{array}\right\} & \leq & f(t-\tilde{k}+\delta+1)-X^*(\delta+1)-(t-1-\tilde{k})b \\
\{\textit{ (3.12) for } x^*(t)\} & \leq & f(t-\tilde{k}+\delta+1)-X^*(\delta+1)-\sum_{k=\delta+2}^{t-\tilde{k}+\delta}x^*(k) \\
 & = & f(t-\tilde{k}+\delta+1)-X^*(t-\tilde{k}+\delta) \\
\{\textit{ use (3.14)}\} & \leq & m_s\,.
\end{array}$$

To show that (3.8) is satisfied we make a similar derivation, where instead of $\tilde{k}$, we use $\hat{k}$, which also includes time unit t, i.e.,

$$\hat{k}=\max\{k\in\{1,\ldots,t\}\;\left|\;\begin{array}{rcll}x(k) & = & P(k)-X(k-1) & \vee\\ x(k) & = & m_r+C(k-1)-X(k-1) & \end{array}\right\}\,, \qquad (3.18)$$

so for all k with $\hat{k}<k\leq t$, $x(k)=b$ holds. Again, we distinguish three cases, viz. (i) $\hat{k}=-\infty$, (ii) $x(\hat{k})=P(\hat{k})-X(\hat{k}-1)$, and (iii) $x(\hat{k})=m_r+C(\hat{k}-1)-X(\hat{k}-1)$.

(i) If $\hat{k}=-\infty$, then $X(t)=tb$. We derive

$$\begin{array}{rcl}
X(t)-C(t) & = & tb-P(t-\delta) \\
\{\textit{ use (3.1) and (3.12) for } x^*(t)\} & \geq & X^*(t)-f(t-\delta) \\
\{\textit{ use (3.15)}\} & \geq & 0\,.
\end{array}$$

(ii) If $x(\hat{k})=P(\hat{k})-X(\hat{k}-1)$, then $X(\hat{k})=P(\hat{k})$. We derive

$$\begin{array}{rcl}
X(t)-C(t) & = & X(\hat{k})+\sum_{k=\hat{k}+1}^{t}x(k)-P(t-\delta) \\
 & = & P(\hat{k})+(t-\hat{k})b-P(t-\delta)\,.
\end{array}$$

If $\hat{k}\geq t-\delta$, then it follows immediately that $X(t)-C(t)\geq 0$. If $\hat{k}<t-\delta$, we derive

$$\begin{array}{rcl}
X(t)-C(t) & = & (t-\hat{k})b-\sum_{k=\hat{k}+1}^{t-\delta}p(k) \\
\{\textit{ use (3.1)}\} & \geq & (t-\hat{k})b-f(t-\delta-\hat{k}) \\
\{\textit{ (3.12) for } x^*(t)\} & \geq & X^*(t-\hat{k})-f(t-\delta-\hat{k}) \\
\{\textit{ use (3.15)}\} & \geq & 0\,.
\end{array}$$

(iii) If $x(\hat{k})=m_r+C(\hat{k}-1)-X(\hat{k}-1)$, then $X(\hat{k})=m_r+C(\hat{k}-1)$. We derive

$$
\begin{aligned}
X(t)-C(t) &= X(\hat{k})+\sum_{k=\hat{k}+1}^{t} x(k)-C(t) \\
&= m_r+C(\hat{k}-1)+(t-\hat{k})b-C(t) \\
\{\textit{ use (3.2)}\} \quad &= m_r+(t-\hat{k})b+P(\hat{k}-\delta-1)-P(t-\delta) \\
&= m_r+(t-\hat{k})b-\sum_{k=\hat{k}-\delta}^{t-\delta} p(k) \\
\{\textit{ use (3.1)}\} \quad &\geq m_r+(t-\hat{k})b-f(t-\hat{k}+1) \\
\left\{ \begin{array}{l} \textit{(3.16) for } X^*(\delta+1) \\ \textit{and } f(0)=0 \end{array} \right\} \quad &\geq X^*(\delta+1)+(t-\hat{k})b-f(t-\hat{k}+1) \\
\{\textit{ (3.12) for } x^*(t)\} \quad &\geq X^*(\delta+1)+\sum_{k=\delta+2}^{t-\hat{k}+\delta+1} x^*(k)-f(t-\hat{k}+1) \\
&= X^*(t-\hat{k}+1+\delta)-f(t-\hat{k}+1) \\
\{\textit{ use (3.15)}\} \quad &\geq 0\,.
\end{aligned}
$$

□

Both sets of constraints thus describe the same solutions b, m_s, and m_r. Furthermore, the greedy transmission strategy given by $x(t) = \min\{b, P(t) - X(t-1), m_r + C(t-1) - X(t-1)\}$ is feasible. Therefore, we may use f as the actual supply and we can use (3.12)–(3.16) instead of (3.5)–(3.9) for the leaky-bucket-controlled streams. As a corollary of this theorem we have that MLBSSP reduces to MSSP.

Corollary 3.1. *MLBSSP reduces to MSSP.*

Proof. Consider an instance of MLBSSP. Take the sets $\mathcal{T}$, $\mathcal{D}$, and $\mathcal{N}$ as direct input for MSSP as well as the capacities B and M_n for all $n \in \mathcal{N}$. Now take as supply scheme for each $d \in \mathcal{D}$, $p_d(t) = f_d(t) - f_d(t-1)$, and as demand scheme for each $d \in \mathcal{D}$, $c_d(t) = f_d(t-\delta_d) - f_d(t-\delta_d-1)$. As any solution b_d, $m_{s_d,d}$, and $m_{r_d,d}$ to this instance of MSSP satisfies (2.1)–(2.7) for the greedy transmission strategy, it follows that it also satisfies (3.3), (3.4), and (3.12)–(3.16). Furthermore, from Theorem 3.1 is follows that any feasible solution w.r.t. (3.12)–(3.16) is also feasible w.r.t. (3.5)–(3.9). □

3.3 Single-stream methods

In this section we show how to efficiently solve the LP sub-problem 2.15 on page 19 for leaky-bucket-controlled streams. First, we derive in Section 3.3.1 four new stream constraints that we use to determine the solution of all cases of the sub-problems for leaky-bucket-controlled streams. In Section 3.3.2 we consider

the cases for which only either the bus share or the sending buffer share or the receiving buffer share needs to be minimized. In Section 3.3.3 we consider the case for which both the sending buffer share and the receiving buffer share need to be minimized. In Section 3.3.4 we show how the cases can be solved for which an optimal trade-off has to be made between the bus share and one or two buffer shares. A feasible transmission strategy is for all cases given by the greedy transmission strategy.

The methods that we present in this section all use the piecewise linearity of the function f. For methods solving these sub-problems in which f only needs to be concave we refer to Den Boef et al. [2004].

3.3.1 Necessary and sufficient constraints

In the previous section we have shown that the solution approach for MSSP can also be used to obtain an optimal solution for MLBSSP. However, the cumulative supply and demand schemes that are used in the instance of MSSP constructed from a given instance of MLBSSP to solve the latter, have some specific properties, namely they are piecewise-linear and concave. These properties can be exploited to solve the sub-problems even more efficiently. In this section we derive four new constraints for sub-problems in which the cumulative supply scheme is given by a concave function f, and the cumulative demand scheme by the same function f shifted in time by a delay δ, i.e., by $f(t-\delta)$. We also show that these new constraints are sufficient. In Sections 3.3.2–3.3.4 we show how these constraints can be used to solve the sub-problems for a leaky-bucket-controlled stream, i.e., for a piecewise-linear, concave function f.

Constraints (3.12)–(3.16) include transmission strategy $x(t)$, with which no costs are associated in the objective function of sub-problem (2.15). Therefore, we first derive four necessary constraints on b, m_s, and m_r, that do not involve the transmission strategy. Then we show that they are sufficient, i.e., any solution that satisfies these constraints, is also feasible w.r.t. (3.12)–(3.16), so that we do not need to use (3.12)–(3.16).

We derive four necessary constraints using the fact that the allocated bandwidth share b should be large enough to avoid buffer underflow at the receiving buffer and buffer overflow at the sending buffer. To avoid buffer underflow at the receiving buffer, the bandwidth share should be large enough such that for each time unit the cumulative amount of data demanded could have been transmitted, i.e., such that,

$$f(t) \leq (t+\delta)b\,, \qquad \forall\, t \in \mathcal{T}\,. \tag{3.19}$$

Figure 3.4 gives a graphical representation of this constraint.

Now to avoid buffer overflow at the sending buffer, the bandwidth share should be large enough such that for each time unit the difference between the cumulative

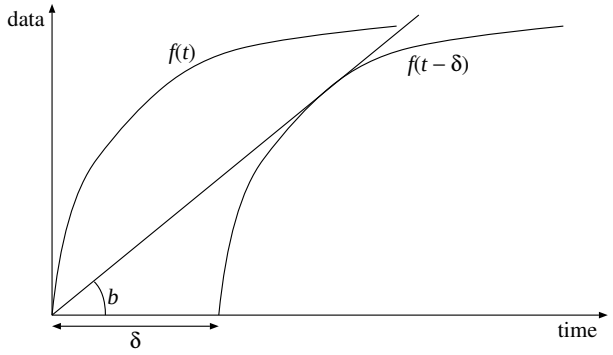

Figure 3.4. The bandwidth share b should be large enough such that all data can be transmitted in time before it is demanded. The minimum value of b is thus given by the slope of the line starting in $(0,0)$ that touches $f(t-\delta)$.

amount of data supplied and the cumulative amount of data that maximally could have been transmitted, is not larger than the sending buffer share, i.e., such that,

$$f(t) - (t-1)b \leq m_s\,, \qquad \forall\, t \in \mathcal{T}\,. \tag{3.20}$$

See also Figure 3.5.

Constraints (3.19) and (3.20) are derived with the assumption that data can be transmitted using the full bandwidth share. However, all data that is transmitted needs to be buffered at the receiving side. When the buffer share at the receiving side is completely filled, no more data can be transmitted until data is demanded from the receiving buffer. A larger bandwidth share may then be needed to transmit data at a later time such that buffer underflow at the receiving side and buffer overflow at the sending side are avoided.

There are $\delta+1$ time units in which data can be transmitted before data is demanded from the receiving buffer. The amount of data that can be transmitted during these $\delta+1$ time units is thus bounded by the receiving buffer share m_r. The bandwidth share b should be large enough such that after the first $\delta+1$ time units enough data can be transmitted to avoid buffer underflow at the receiving side. Constraint (3.19) states that for time units $t' \geq \delta+1$, to avoid buffer underflow, we need $f(t'-\delta) \leq t'b$. As during the first $\delta+1$ time units only m_r data can be transmitted, we now have for $t' \geq \delta+1$, that $f(t'-\delta) - m_r \leq (t'-(\delta+1))b$. If we take $t = t'-\delta$, we get,

$$f(t) - m_r \leq (t-1)b\,, \qquad \forall\, t \in \mathcal{T}\,. \tag{3.21}$$

Figure 3.6 shows an example of this.

Furthermore, the bandwidth share should be large enough such that after the

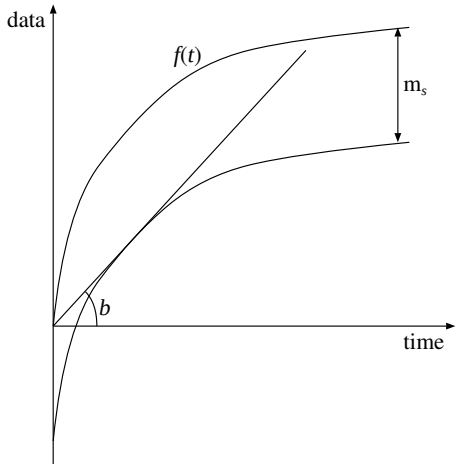

Figure 3.5. The bandwidth share should be large enough such that all data can be transmitted in time before then sending buffer share is full. The lower bound in this figure concerning the sending buffer is obtained by shifting the supply function f downwards with the size of the sending buffer share and by shifting it one time unit leftwards. This latter shift is required as data is supplied at the begin of a time unit and is buffered completely before it is transmitted.

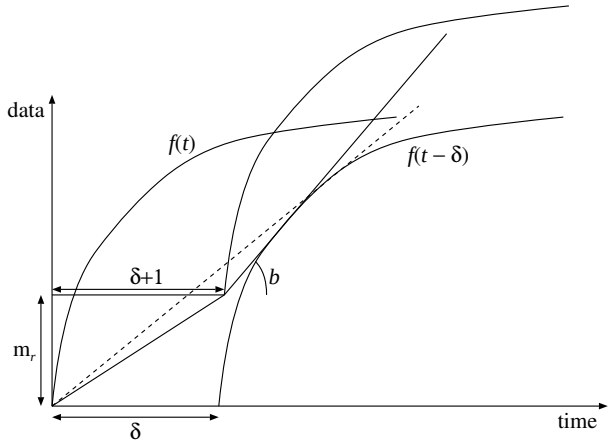

Figure 3.6. The receiving buffer share can limit the amount of data that can be transmitted during a period of time. As data is supplied at the begin of a time unit and demanded at the end of a time unit, there are $\delta + 1$ time units to transmit data before the demand starts. This means that the upper bound in this figure concerning the receiving buffer share is obtained by shifting the demand function upwards with the size of the receiving buffer share and by shifting it one time unit to the right.

first $\delta+1$ time units enough data can be transmitted to avoid buffer overflow at the sending side. Constraint (3.20) states that for time units $t > \delta+1$, we need $f(t) - m_s \leq (t-1)b$. As during the first $\delta+1$ time units only m_r data can be transmitted, we now have for $t > \delta+1$, $f(t) - m_s - m_r \leq (t-1-(\delta+1))b$. It now follows that,

$$f(t) - m_r - (t-\delta-2)b \leq m_s\,, \qquad \forall\, t > \delta+1\,. \tag{3.22}$$

Notice that (3.21) and (3.22) follow from (3.19) and (3.20), respectively, by decreasing the amount of data to be transmitted, given by $f(t)$, with m_r, and by decreasing the number of time units during which data can be transmitted, given directly in front of b, with $\delta+1$.

We continue with showing that (3.19)–(3.22) are not only necessary but also sufficient for any solution that is feasible w.r.t. (3.12)–(3.16).

Theorem 3.2. *Any feasible solution b, m_s, and m_r w.r.t. (3.19)–(3.22) is also a feasible solution w.r.t. (3.12)–(3.16).*

Proof. Let b, m_s, and m_r satisfy (3.19)–(3.22). Again, we take the greedy transmission strategy $x(t) = \min\{b, f(t)-X(t-1), m_r + f(t-\delta-1) - X(t-1)\}$. It follows immediately from the transmission strategy that (3.12), (3.13), and (3.16) are satisfied. Let $t \in \mathcal{T}$. We choose $\tilde{k}$ analogous to (3.17) with $P(t) = f(t)$, i.e., $\tilde{k} = \max\{k \in \{1,\ldots,t-1\} \,|\, x(k) = f(k) - X(k-1) \vee x(k) = m_r + f(k-\delta-1) - X(k-1)\}$. Thus, for all k with $\tilde{k} < k < t$, $x(k) = b$ holds. First, we consider (3.14). We again distinguish three cases, viz. (i) $\tilde{k} = -\infty$, (ii) $x(\tilde{k}) = f(\tilde{k}) - X(\tilde{k}-1)$, and (iii) $x(\tilde{k}) = m_r + f(\tilde{k}-\delta-1) - X(\tilde{k}-1)$.

(i) If $\tilde{k} = -\infty$, then $X(t-1) = (t-1)b$. We derive

$$\begin{aligned} f(t) - X(t-1) &= f(t) - (t-1)b \\ \{\ \textit{use (3.20)}\} &\leq m_s\,. \end{aligned}$$

(ii) If $x(\tilde{k}) = f(\tilde{k}) - X(\tilde{k}-1)$, then $X(\tilde{k}) = f(\tilde{k})$. We derive

$$\begin{aligned} f(t) - X(t-1) &= f(t) - X(\tilde{k}) - \sum_{k=\tilde{k}+1}^{t-1} x(k) \\ &= f(t) - f(\tilde{k}) - (t-\tilde{k}-1)b \\ \{\ \textit{concavity of } f\} &\leq f(t-\tilde{k}) - (t-\tilde{k}-1)b \\ \{\ \textit{use (3.20)}\} &\leq m_s\,. \end{aligned}$$

(iii) If $x(\tilde{k}) = m_r + f(\tilde{k}-\delta-1) - X(\tilde{k}-1)$, then $X(\tilde{k}) = m_r + f(\tilde{k}-\delta-1)$. We

derive

$$\begin{aligned} f(t)-X(t-1) &= f(t)-X(\tilde{k})-\sum_{k=\tilde{k}+1}^{t-1} x(k) \\ \{\, x(k)=b \text{ for } k>\tilde{k}\,\} &= f(t)-m_r-f(\tilde{k}-\delta-1)-(t-\tilde{k}-1)b \\ \{\, \textit{concavity of } f\,\} &\leq f(t-\tilde{k}+\delta+1)-m_r-(t-\tilde{k}-1)b \\ \{\, \textit{use (3.22)}\,\} &\leq m_s\,. \end{aligned}$$

To show that (3.15) is satisfied, we make a similar derivation, where again we use $\hat{k}$ analogous to (3.18) with $P(t)=f(t)$, i.e., $\hat{k}=\max\{k\in\{1,\ldots,t\}\,|\,x(k)=f(k)-X(k-1)\vee x(k)=m_r+f(k-\delta-1)-X(k-1)\}$. Again, we distinguish three cases, viz. (i) $\hat{k}=-\infty$, (ii) $x(\hat{k})=f(\hat{k})-X(\hat{k}-1)$, and (iii) $x(\hat{k})=m_r+f(\hat{k}-\delta-1)-X(\hat{k}-1)$.

(i) If $\tilde{k}=-\infty$, then $X(t)=tb$. We derive

$$\begin{aligned} X(t)-f(t-\delta) &= tb-f(t-\delta) \\ \{\, \textit{use (3.19)}\,\} &\geq 0\,. \end{aligned}$$

(ii) If $x(\tilde{k})=f(\tilde{k})-X(\tilde{k}-1)$, then $X(\tilde{k})=f(\tilde{k})$. We derive

$$\begin{aligned} X(t)-f(t-\delta) &= X(\tilde{k})+\sum_{k=\tilde{k}+1}^{t} x(k)-f(t-\delta) \\ &= f(\tilde{k})+(t-\tilde{k})b-f(t-\delta)\,. \end{aligned}$$

If $\tilde{k}\geq t-\delta$, it immediately follows that $X(t)-f(t-\delta)\geq 0$. If $\tilde{k}<t-\delta$, we derive using the concavity of f

$$\begin{aligned} X(t)-f(t-\delta) &\geq (t-\tilde{k})b-f(t-\tilde{k}-\delta) \\ \{\, \textit{use (3.19)}\,\} &\geq 0\,. \end{aligned}$$

(iii) If $x(\tilde{k})=m_r+f(\tilde{k}-\delta-1)-X(\tilde{k}-1)$, then $X(\tilde{k})=m_r+f(\tilde{k}-\delta-1)$. We derive

$$\begin{aligned} X(t)-f(t-\delta) &= X(\tilde{k})+\sum_{k=\tilde{k}+1}^{t} x(k)-f(t-\delta) \\ &= m_r+f(\tilde{k}-\delta-1)+(t-\tilde{k})b-f(t-\delta) \\ \{\, \textit{concavity of } f\,\} &\geq m_r+(t-\tilde{k})b-f(t-\tilde{k}+1) \\ \{\, \textit{use (3.21)}\,\} &\geq 0\,. \end{aligned}$$

□

By rewriting (3.19)–(3.22) we can obtain the following constraints on b given

values of m_s and m_r.

$$\begin{aligned} b &\geq f(t-\delta)/t\,, && \forall\, t \in \mathcal{T}\,, && (3.23)\\ b &\geq (f(t+1)-m_s)/t\,, && \forall\, t \in \mathcal{T}\,, && (3.24)\\ b &\geq (f(t+1)-m_r)/t\,, && \forall\, t \in \mathcal{T}\,, && (3.25)\\ b &\geq (f(t+\delta+2)-m_r-m_s)/t\,, && \forall\, t \in \mathcal{T}\,. && (3.26) \end{aligned}$$

Notice that only one of (3.24) and (3.25) need to be considered, depending on which buffer is the largest.

Likewise, we can obtain the following constraints on m_s given values of b and m_r, and on m_r given values of b and m_s.

$$\begin{aligned} m_s &\geq f(t)-(t-1)b\,, && \forall\, t \in \mathcal{T}\,, && (3.27)\\ m_s &\geq f(t+\delta+1)-m_r-(t-1)b\,, && \forall\, t \in \mathcal{T}\,, && (3.28)\\ m_r &\geq f(t)-(t-1)b\,, && \forall\, t \in \mathcal{T}\,, && (3.29)\\ m_r &\geq f(t+\delta+1)-m_s-(t-1)b\,, && \forall\, t \in \mathcal{T}\,. && (3.30) \end{aligned}$$

Constraints (3.28) and (3.30) are equivalent, however, this gives a clear presentation of the constraints on m_s and m_r separately. This constraint can also be rewritten into the following constraint on the total buffer space required given a value of b.

$$m_s+m_r \geq f(t+\delta+1)-(t-1)b\,, \qquad \forall\, t \in \mathcal{T}\,. \qquad (3.31)$$

In the following sections we show how to use (3.23)–(3.31) to solve the subproblems for a leaky-bucket-controlled stream. When a stream is controlled by k leaky buckets with parameters (σ_i, ρ_i) for $i = 1, \ldots, k$, the corresponding function f is piecewise-linear and concave, and it is given by $f(t) = \min_i\{\sigma_i + \rho_i t\}$. W.l.o.g. we assume that $\sigma_i \leq \sigma_{i+1}$ and $\rho_i \geq \rho_{i+1}$ for $i = 1, \ldots, k-1$.

3.3.2 Single resource minimization

When only one resource needs to be minimized, i.e., either the bandwidth share or the sending or receiving buffer share, we can determine its optimal value directly from the appropriate constraints, given by (3.23)–(3.30). An optimal value for the bandwidth share can be obtained by determining the minimum value that satisfies (3.23)–(3.26) for all time units $t \in \mathcal{T}$. The right-hand-sides of these constraints all have the form $f(t+y)/t$, where the value of y is independent of t when determining the maximum. When f is a piecewise-linear, concave function, the maximum is attained at one of the bending points of f. So, instead of determining the values of the right-hand-side for all $t \in \mathcal{T}$, only the bending points of f need to be considered.

To obtain an optimal value for either the sending or the receiving buffer share, we need to determine the minimum value that satisfies (3.27) and (3.28), or (3.29) and (3.30), respectively, for all time units $t \in \mathcal{T}$. The right-hand-sides of these

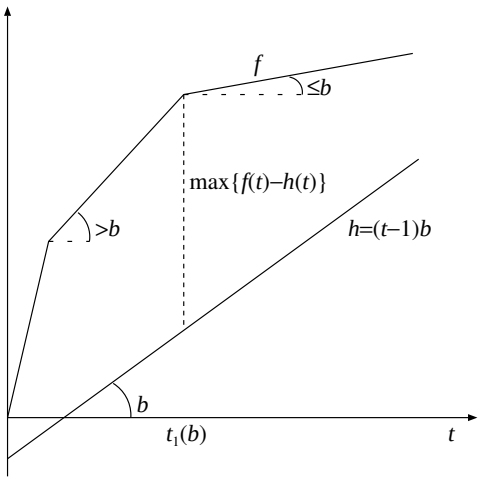

Figure 3.7. The maximum difference between a piecewise-linear, concave function f and a linear function h is attained at the bending point of f where the gradient of f changes from being larger than the gradient of h to being smaller.

constraints all have the form $f(t) - bt - y$, where the value of y is independent of t when determining the maximum. When f is a piecewise-linear, concave function, the maximum is again attained at a bending point. Obviously, it is attained at the bending point between ρ_i and ρ_{i+1} with $\rho_i > b$ and $\rho_{i+1} \leq b$; see also Figure 3.7.

3.3.3 Two-buffer minimization

When both the sending and the receiving buffer share of a single stream need to be minimized, we take the following approach. The buffer shares m_s and m_r have to satisfy (3.27)–(3.31). For a given value of b, let $m_1(b)$ be the maximum value of the right-hand sides of (3.27) and (3.29), and let $m_2(b)$ be the maximum value of the right-hand side of (3.31), i.e.,

$$\begin{aligned} m_1(b) &= \max\{ f(t) - (t-1)b \mid t \geq 1\} , \\ m_2(b) &= \max\{ f(t+\delta+1) - (t-1)b \mid t \geq 1\} . \end{aligned}$$

If $2m_1(b) \geq m_2(b)$ holds, then $m_s = m_1(b)$ and $m_r = m_1(b)$ are the optimal values as then (3.27) and (3.29) are tight and (3.31) has slack. Otherwise, their sum $m_s + m_r = 2m_1 < m_2$ does not satisfy (3.31). Hence, at least one of them must be increased: first the buffer with the lowest cost coefficient and then, if necessary, the buffer with the highest cost coefficient. Formally, if $c_s \leq c_r$, then optimal values are given by $m_s = \min\{m_2(b) - m_1(b), M_s\}$ and $m_r = m_2(b) - m_s$. Notice that $m_r \geq m_1(b)$. If $c_s > c_r$, then optimal values are given by $m_r = \min\{m_2(b) - m_1(b), M_r\}$ and $m_s = m_2(b) - m_r$. With the above approach we can solve the sub-problem

case with $c_s, c_r > 0$ and $c_b \leq 0$, taking the bandwidth share b equal to B. Notice that this approach is similar to Algorithm 1 for fully-specified streams, which first minimizes the expensive buffer and then the cheap buffer.

3.3.4 Bandwidth-buffer trade-off

When the cost coefficient of the bandwidth and of at least one buffer is positive, a trade-off has to be made between bandwidth and buffer space. First, we show how the trade-off can be performed between the bandwidth and one buffer. Then, we consider the trade-off between the bandwidth and both buffers. For all trade-offs we initially start with a solution with minimum bandwidth share b and corresponding values of m_s and m_r, by first minimizing b for $m_s = M_s$ and $m_r = M_r$ and then minimizing m_s and m_r for the obtained value of b. Starting from the initial solution, we increase b and determine the effect on m_s and m_r.

Trade-off between b and m_s, and between b and m_r.

For the trade-off between b and m_s, the buffer share m_s needs to satisfy (3.27) and (3.28) with $m_r = M_r$, for a given value of b. Furthermore, for the trade-off between b and m_r, the buffer share m_r needs to satisfy (3.29) and (3.30) with $m_s = M_s$, for a given value of b. We describe the trade-off between b and m_s; the trade-off between b and m_r can be performed analogously.

First, we define the following lower bounds on the buffer share m_s as functions of b, which correspond to the right-hand-sides of each of the mentioned constraints.

$$\begin{aligned} m_1(b) &= \max\{ f(t) - (t-1)b \mid t \geq 1 \}, \\ m_2^{\mathrm{s}}(b) &= \max\{ f(t+\delta+1) - M_r - (t-1)b \mid t \geq 1 \}, \end{aligned}$$

where the 's' in $m_2^{\mathrm{s}}(b)$ indicates that it is a lower bound specifically for use in the trade-off between the bandwidth and the sending buffer. Notice that m_1 is the same lower bound as defined for the joint minimization of two buffers. As f is a piecewise-linear function, each of these lower bound functions forms a piecewise-linear, convex curve in the (b, m_s)-plane. The trade-off curve between b and m_s is given by the maximum of these lower bound curves; see Figure 3.8. Instead of constructing each of the lower bound curves before determining their maximum, we construct them iteratively while increasing b. Furthermore, we determine the trade-off curve efficiently by using the expressions of the lower bounds as we show hereafter.

First, we denote the values of t at which the above-defined lower bounds attain

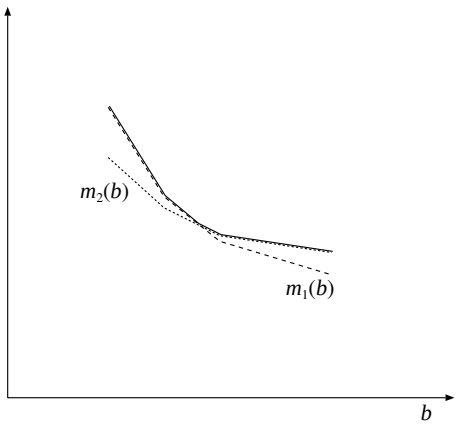

Figure 3.8. The trade-off curve between b and m_s is given by the maximum of the lower bounds m_1 and m_2^s.

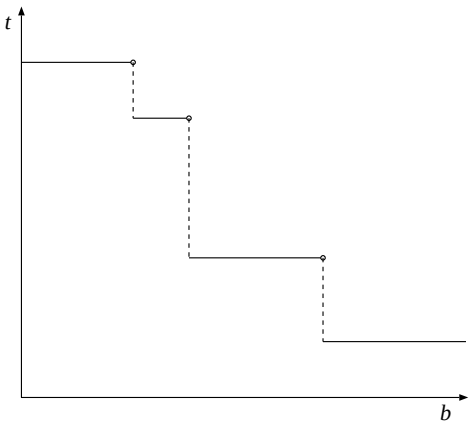

Figure 3.9. Example of a piecewise-constant function such as $t_1(b)$, that jumps at specific values of b.

their maximum by $t_1(b)$ and $t_2^s(b)$, respectively. So we have for a given b,

$$\begin{aligned} m_1(b) &= f(t_1(b)) - (t_1(b)-1)b\,, \\ m_2^s(b) &= f(t_2^s(b)+\delta+1) - M_r - (t_2^s(b)-1)b\,, \end{aligned}$$

The function f is piecewise linear and concave, thus the maximum values are attained at the bending point of f between ρ_i and ρ_{i+1} with $\rho_i > b$ and $\rho_{i+1} \leq b$. When b is increased and becomes equal to ρ_i, the corresponding lower bound values shift along the line segment to the bending point at the beginning of the segment. Therefore, $t_1(b)$ and $t_2^s(b)$ are piecewise-constant functions of b, that jump in value when b equals one of the ρ_i; see Figure 3.9 for an example. If for a certain lower bound the value of t corresponding to the bending point at which the maximum is attained, is less than 1 and, therefore, is infeasible, then this lower bound attains its value at $t = 1$ and remains constant for any further increase of b.

Next, we denote the gradients of the lower bounds m_1 and m_2^s by Δ_1 and Δ_2^s, respectively. It follows from above definitions that $\Delta_1(b) = -(t_1(b)-1)$ and $\Delta_2^s(b) = -(t_2^s(b)-1)$. Notice that $\Delta_1(b)$ and $\Delta_2^s(b)$ are also piecewise-constant functions. If b is now increased by an amount ε for which the gradient of a lower bound remains constant, then the value of this lower bound changes by ε times the gradient.

Now we describe how to determine the maximum of the lower bounds efficiently. As both m_1 and m_2^s involve function f, their values are for a given b at-

tained at the same bending point of f. More precisely, if $t_2^s(b) > 1$, then we have $t_1(b) = t_2^s(b) + \delta + 1$. So, $\Delta_2^s(b) > \Delta_1(b)$ holds, i.e., the decrease of $m_1(b)$ is larger than the decrease of $m_2^s(b)$ when b is increased. Thus, if for a given value of b we have $m_2^s(b) \geq m_1(b)$, this also holds for larger values of b. We have $m_2^s(b) = m_1(b)$ if $f(t_2^s(b) + \delta + 1) - M_r - (t_2^s(b) - 1)b = f(t_1(b)) - (t_1(b) - 1)b$ holds, which is equivalent to $b = M_r/(\delta + 1)$ if $t_2^s(M_r/(\delta + 1)) > 1$. It now follows that if $t_2^s(M_r/(\delta+1)) > 1$, $m_2^s(b) \geq m_1(b)$ holds if and only if $b \geq M_r/(\delta+1)$. If $t_2^s(b) = 1$ for some b with $b < M_r/(\delta + 1)$, then $m_2^s(b) = m_1(b)$ when $f(\delta + 2) - M_r = f(t_1(b)) - (t_1(b) - 1)b$ holds. As $m_2^s(b)$ remains equal to $f(\delta + 2) - M_r$ when $t_2^s(b) = 1$, this ends the trade-off curve.

We now have shown how to determine $\max\{m_1(b), m_2^s(b)\}$ for all b, which gives the trade-off curve. The optimal solution is found at one of the bending points of the trade-off curve including the begin and end point. When the trade-off curve is constructed iteratively starting with a solution with minimum bandwidth, the optimal solution is obtained as soon as increasing b by an amount ε would not lead to a reduction in costs, i.e., when $c_b\varepsilon + c_s\Delta_{m_s}(b)\varepsilon < 0$ or $\Delta_{m_s}(b) < -\frac{c_b}{c_s}$ holds, with $\Delta_{m_s}(b)$ the gradient of the trade-off curve, i.e., $\Delta_{m_s}(b) = \Delta_1(b)$ or $\Delta_{m_s}(b) = \Delta_2^s(b)$ when $\max\{m_1(b), m_2^s(b)\}$ is given by $m_1(b)$ or $m_2^s(b)$, respectively.

Trade-off between b, m_s, and m_r.

We assume that $c_s \leq c_r$. If $c_s > c_r$ the optimal solution can be obtained in an analogous manner. The trade-off is again performed by first determining the solution with minimum bandwidth and corresponding optimal buffer sizes as described for the previous trade-off, and then increasing the bandwidth, while determining the effect on the buffer sizes.

For the trade-off between b, m_s, and m_r we use lower bound curves on m_s, m_r, and the total buffer size $m_s + m_r$. The lower bound curves on m_s and m_r are both given by the lower bound $m_1(b)$ as defined for the trade-off between the bandwidth and one buffer. Lower bound $m_2^s(b)$ as defined for that trade-off is excluded from these lower bound curves as it involves one of the buffers. Instead we use $m_2(b)$ as defined for the joint minimization of both buffers, i.e.,

$$m_2(b) = \max\{ f(t + \delta + 1) - (t - 1)b \mid t \geq 1\} .$$

Here, $m_2(b)$ gives a lower bound on the total buffer size $m_s + m_r$ as given in (3.31). Furthermore, we again define $t_2(b)$ as the function that returns for each b the time unit in which the maximum of $m_2(b)$ is attained, i.e.,

$$m_2(b) = f(t_2(b) + \delta + 1) - (t_2(b) - 1)b .$$

Finally, we define $\Delta_2(b)$ again as the gradient of $m_2(b)$. It follows that $\Delta_2(b) = -(t_2(b) - 1)$.

So, we have $m_1(b)$ as a lower bound curve on m_s and m_r, and $m_2(b)$ as a lower

bound curve on $m_s + m_r$. These lower bound curves can be constructed iteratively as described for the trade-off between the bandwidth and one buffer. During the trade-off we distinguish two cases, viz. $2m_1(b) \geq m_2(b)$ and $2m_1(b) < m_2(b)$.

As we have shown for the trade-off with one buffer, $\Delta_2(b) < \Delta_1(b)$ for the relevant values of b and, therefore, if $2m_1(b) < m_2(b)$ holds for a certain value of b, it also holds for any larger value of b. We have $2m_1(b) = m_2(b)$ if $2(f(t_1(b)) - (t_1(b) - 1)b = f(t_2(b) + \delta + 1) - (t_2(b) - 1)b$ holds. Opposite to $m_1(b) = m_2(b)$ for the trade-off with one buffer, this does not lead to an expression for b independent of $t_1(b)$. Hence, for each value change of $t_1(b)$ and $t_2(b)$, i.e., when $b = \rho_i$ with ρ_i the slope of a line segment of f, we have to determine the new value of b for which $2m_1(b) = m_2(b)$ holds, until we have $2m_1(b) < m_2(b)$.

If $2m_1(b) \geq m_2(b)$ holds, the required values for the buffer shares separately are enough to meet the required value of the total buffer size. In this case, for a given b, $m_s = m_r = m_1(b)$. If now b is increased by an amount ε, the solution costs change by $(c_b + (c_s + c_r)\Delta_1(b))\varepsilon$. Thus, if $\Delta_1(b) < -\frac{c_b}{c_s + c_r}$, then a better solution is obtained by increasing b.

If $2m_1(b) < m_2(b)$ holds, the lower bound on the total buffer size determines the total buffer allocation, as described in Section 3.3.3. For $c_s \leq c_r$ we get for a given b that $m_s = \min\{m_2(b) - m_1(b), M_s\}$ and $m_r = m_2(b) - m_s$. If now b is increased by an amount ε, the solution costs change by $(c_b + c_s(\Delta_2(b) - \Delta_1(b)) + c_r\Delta_1(b))\varepsilon$ if $m_2(b) - m_1(b) \leq M_s$, and by $(c_b + c_r\Delta_2(b))\varepsilon$ if $m_2(b) - m_1(b) > M_s$. From the definitions it follows that $m_2(b) - m_1(b) = M_s$ if $f(t_2(b) + \delta + 1) - (t_2(b) - 1)b - f(t_1(b)) + (t_1(b) - 1)b = M_s$, which is equivalent to $b = M_s/(\delta + 1)$ if $t_2(M_s/(\delta + 1)) > 1$.

Finally, we remark that the time complexity of the algorithms for all cases depends only linearly on the number of bending points of the function f. As the number of bending points is generally very small, the algorithms are very efficient when dealing with leaky-bucket-controlled streams.

3.4 Results

In the previous sections we have described a method to solve MLBSSP. As we already mentioned for fully-specified streams, the run time is an important parameter when the method is used for admission control. Since the algorithms for the sub-problems only linearly depend on the number of bending points of f, the expected run time to solve an instance of MLBSSP is very small. The experiments that we have performed, confirmed this expectation. Even when using streams which are described by a maximum number of leaky buckets, see the next section for a description, instances of 10–15 streams are solved within a fraction of a second. Therefore, we exclude results concerning the run time in this section.

Instead, we mainly focus on results concerning the utilization of the bandwidth and buffer shares, and compare them to the utilization results for fully-specified streams which were given in Section 2.3.2. We begin with describing the setting and data used in Section 3.4.1. Next, we discuss the results in Section 3.4.2.

3.4.1 Experiment setting

For the experiments we used the same setting as for fully-specified streams. Thus, we considered a bus with six nodes and buffers connected to it. Furthermore, we considered eight different settings concerning the values of the bandwidth, buffer sizes, and stream delays, as given in Table 2.2 on page 50. We also used the same streams in the same sequence as given in Table 2.1 on page 50; see also Table 3.1. Finally, to determine the order in which we consider the streams for solving the sub-problems we use strategy 3 as described in Section 2.3.1. We next describe how we obtain leaky-bucket descriptions for the used streams.

First, we determine the empirical envelope $E(t)$ of a stream, which gives for all t the maximum amount of data that is supplied for a stream during t consecutive time windows. Thus, $E(t)$ is given by

$$E(t) = \max\{P(\xi + t) - P(\xi) \mid \xi \geq 0\} \; . \tag{3.32}$$

Note that $E(t)$ already satisfies (3.1). However, $E(t)$ is generally not a concave function and, thus, does not give a leaky-bucket description. Therefore, we next determine the concave hull of $E(t)$, given by $\hat{E}(t)$. Notice that $\hat{E}(t)$ is the tightest piecewise-linear, concave function that bounds the supply of a stream. Therefore, we use $\hat{E}(t)$ as the supply-bounding function that corresponds to the maximum number of leaky buckets of a stream. Table 3.1 gives all the used streams and the number of leaky buckets in their maximum-leaky-bucket descriptions. As $\hat{E}(t)$ is the convex hull of the empirical envelope of a stream, the upper bounds at the bending points of $\hat{E}(t)$ are tight. Therefore, we suspect that using the maximum-leaky-bucket description will give the same results as using the exact supply and demand scheme. In Section 3.4.2 we will see if the experimental results confirm this supposition.

We have now obtained for each stream a leaky-bucket description consisting of the maximum number of leaky buckets that is required to describe a stream. However, this number of leaky buckets is much higher than the amount of leaky buckets that realistically can be expected in a network. The number of leaky buckets that is available in an actual network to control streams, may be limited to only two or three. Therefore, besides the maximum-leaky-bucket description of a stream, we also consider descriptions consisting of a lower number of leaky buckets.

These descriptions can be obtained in a numerous amount of methods. However, not every description is suitable for testing our solution method. Especially

	type	*T*	*send.*	*rec.*	*avg. frame*	*largest frame*	*max. # lb*
1	fantasy	256,599	1	2	22,859.6	138,476	60
2	comedy	161,082	3	4	23,359.7	126,579	26
3	documentary	76,750	4	1	32,613.6	143,482	42
4	scifi	196,160	5	6	23,359.2	120,377	50
5	pop concert	99,785	2	3	27,352.6	111,372	54
6	thriller	141,800	4	5	24,851.0	129,617	48
7	comedy	74,931	6	1	32,752.3	154,283	44
8	action	151,132	6	2	31,114.1	131,713	61
9	documentary	36,875	1	3	26,829.5	114,601	41
10	action	184,371	5	3	22,112.5	111,101	68
11	pop concert	166,765	2	4	24,631.5	143,871	52
12	comedy	164,199	4	6	21,318.4	122,383	52
13	comedy	38,215	1	5	33,946.5	145,693	33
14	action	165,141	3	6	26,418.6	139,284	63
15	documentary	75,100	5	2	24,717.6	140,321	34

Table 3.1. The 15 streams used in the experiment, now also with the number of leaky buckets in their maximum-leaky-bucket description.

Algorithm 5 Obtaining k leaky-bucket parameters from the concave hull $\hat{E}(t)$ which corresponds to m leaky-bucket parameters with $m > k$.

Input: A set of m leaky-bucket parameters $\{(\hat{\sigma}_j, \hat{\rho}_j) \mid j = 1, \ldots, m\}$, that define $\hat{E}(t)$, a number of leaky buckets k, a cost function $C(f(t), \hat{E}(t))$, and a sensitivity parameter ε.
Output: A set of k leaky-bucket parameters (σ_i, ρ_i) that define the supply-bounding function $f(t)$.
Initialize (σ_i, ρ_i), $i = 1, \ldots, k$.
For $i = 1$ **to** k
  $\sigma_i = \hat{\sigma}_{\lfloor \frac{im}{k} \rfloor}$;
  $\rho_i = \hat{\rho}_{\lfloor \frac{im}{k} \rfloor}$;
Endfor
Greedy modifications
Do
  Cost $= C(f(t), \hat{E}(t))$;
  For $i = k$ **down to** 1
    Select (σ_i, ρ_i) to minimize $C(f(t), \hat{E}(t))$;
  Endfor
While (Cost $- C(f(t), \hat{E}(t)) > \varepsilon$)
$f(t) = \min_{1 \le i \le k} \{\sigma_i + \rho_i t\}$

descriptions which have a relatively large value of σ_1, i.e., the smallest and thus first σ_i, are not suitable, as they would lead to extremely high bandwidth shares. For example, if we consider (3.23) for $t = \delta + 1$, the bandwidth share b needs to satisfy $b \geq f(1)/(\delta+1)$. The higher σ_1 is, the larger the value of $f(1)$ will be, and thus the larger the bandwidth share b will be. Note that as generally $\sigma_1 + \rho_1 t$ gives the value of f not only for $t = 1$, but also for larger values of t, a larger value of σ_1 cannot be offset by an equally lower value of ρ_1.

Thus, a suitable description contains a relatively low value of σ_1. To obtain these descriptions we use two different methods. The first method involves Algorithm 5, originally described by Wrege & Liebeherr [1996]. It takes as input the maximum-leaky-bucket function $\hat{E}(t)$, which consists of n leaky buckets, and the desired number of leaky-bucket parameters, m. To determine these parameters it also requires a cost function which assigns costs to a set of leaky-bucket parameters. Wrege & Liebeherr propose the following cost function to be used with the algorithm.

$$C(f(t),\hat{E}(t)) = \int_0^T \frac{(f(t)-\hat{E}(t))}{\hat{E}(t)} \mathrm{d}t \; . \tag{3.33}$$

This cost function minimizes the total relative deviation of the resulting leaky-bucket description from the maximum-leaky-bucket description. Other cost functions that we have considered, minimize the maximum difference between $f(t)$ and $\hat{E}(t)$, i.e., $C(f(t),\hat{E}(t)) = \max_{0\leq t\leq T}(f(t)-\hat{E}(t))$, or minimize the total area bounded by $f(t)$, i.e., $C(f(t),\hat{E}(t)) = \int_0^T f(t)\mathrm{d}t$. However, both cost functions result in only large values of σ_1, especially for a small number of leaky buckets, and therefore, these latter two cost functions are not suitable to obtain good leaky-bucket descriptions of streams to test our method. The cost function given by (3.33) also results for some streams in a value of σ_1 that is still too large when a small number of leaky buckets is considered. Therefore, we use this method only to obtain leaky-bucket descriptions of at least five leaky buckets.

For leaky-bucket descriptions consisting of fewer than five leaky buckets we instead use the following method. We consider the leaky-bucket parameters of the maximum-leaky-bucket description $\hat{E}(t)$ that give the lowest upper bound on supply for time windows up to $10,000$ time units. From these parameters we uniformly select five as follows. Let m' denote the number of leaky-bucket parameters that we consider, and let these parameters be ordered decreasingly on ρ, i.e., $\rho_1 > \rho_2 > \ldots > \rho_{m'}$. The five selected parameters then are given by the indices

$$\lceil i(m'-5)/6 \rceil + i \; , \tag{3.34}$$

where $i = 1,\ldots,5$. For example, $m' = 20$ gives 4, 7, 11, 14, and 18. The unselected parameters then form six subsequent groups of approximately equal size. If fewer

Setting	*Streams*	U_B	U_M	$U_B^{\min}$	$U_M^{\min}$	$U_B^{\max}$	$U_M^{\max}$
1	3	62.50	44.45	57.17	40.54	70.38	53.66
2	4	59.68	41.04	47.16	35.68	71.33	53.67
3	10	64.06	46.53	50.71	35.68	83.35	56.02
4	13	67.78	38.98	57.17	32.46	84.42	55.33
5	3	63.80	48.81	57.91	37.05	70.63	68.26
6	4	61.17	51.93	47.14	39.30	78.60	68.26
7	12	66.10	63.82	54.49	54.71	83.07	79.28
8	14	74.54	43.48	62.77	33.28	84.98	73.48

Table 3.2. The utilization of the reserved bandwidth and buffer shares in percentages for the leaky-bucket descriptions using the maximum number of leaky buckets for each stream. For each setting (first column) we considered the results for the maximum number of admitted streams (second column). The third and fourth column give the total utilization over all streams for the reserved bandwidth and buffer shares, respectively. The fifth and sixth column give the minimum utilization as achieved by a single stream in the setting. The seventh and eighth column give the maximum utilization as achieved by a single stream in the setting.

than five leaky-bucket parameters are needed, then we stop with selecting parameters as soon as the required number of parameters has been obtained. For this, we consider the values of i in the order 1, 3, 5, 2, 4, e.g., if three parameters are required, then we use (3.34) with $i = 1,3,5$; for $m' = 20$ this gives 4, 11, and 18. This strategy for obtaining leaky-bucket parameters is mainly based on intuition. It has the advantage that σ_1 of the obtained description will be relatively small. Furthermore, when the number of leaky buckets is increased from one to three, this strategy will give a better description for larger windows, and when the number of leaky buckets is increased even further to five, it will refine the leaky-bucket description.

Besides the maximum-leaky-bucket description we use in total twelve different leaky-bucket descriptions, namely descriptions consisting of 1, 2, 3, 4, and 5 leaky-bucket parameters using (3.34), and descriptions consisting of 5, 6, 7, 8, 9, 10, and 20 leaky-bucket parameters using (3.33). We remark that as we assumed that $f(0) = 0$, each of the obtained leaky-bucket descriptions start at $t = 1$. Thus, the first bending point of each description occurs at $t = 1$; for the description consisting of one leaky bucket it is the only bending point.

3.4.2 Experimental results

Table 3.2 gives results concerning the maximum number of admitted streams for each setting and the utilization of the bandwidth and buffer shares, using the maximum number of leaky buckets for all streams. The total, minimum, and maximum utilization of the bandwidth and buffer shares over all streams are again given by (2.17)–(2.22) on page 55, in which the actual realizations of $P(t)$ and $C(t)$ are used.

	Setting 1			*Setting 2*			*Setting 3*			*Setting 4*	
	#	U_B	U_M	#	U_B	U_M	#	U_B	U_M	#	U_B	U_M
F.S.	3	62.12	44.43	4	55.79	40.95	10	58.81	47.28	13	68.13	37.55
Max. LB	3	62.50	44.45	4	59.68	41.04	10	64.06	46.53	13	67.78	38.98
ULB 1	2	51.53	35.52	2	51.53	29.52	10	64.10	46.49	12	62.75	42.33
ULB 3	2	55.01	29.52	2	55.15	27.58	10	64.10	46.38	12	64.01	40.60
ULB 5	3	62.31	44.44	4	64.26	41.40	10	66.86	46.83	13	67.39	33.71
CLB 5	3	62.53	44.46	4	64.68	41.66	10	68.56	46.10	11	61.98	33.52
CLB 10	3	62.55	43.93	4	64.72	41.65	10	68.19	47.15	13	67.76	39.82
	Setting 5			*Setting 6*			*Setting 7*			*Setting 8*	
	#	U_B	U_M	#	U_B	U_M	#	U_B	U_M	#	U_B	U_M
F.S.	3	63.80	49.87	4	59.80	52.10	12	62.71	62.46	14	74.88	42.34
Max. LB	3	63.80	48.81	4	61.17	51.93	12	66.10	63.82	14	74.54	43.48
ULB 1	2	51.52	35.75	4	57.66	42.82	11	62.81	59.23	12	62.72	49.07
ULB 3	3	62.94	55.91	4	66.27	45.49	11	63.61	59.30	14	73.88	41.18
ULB 5	3	63.09	50.36	4	66.00	51.67	12	67.04	62.63	14	74.74	40.65
CLB 5	3	63.38	48.86	4	65.61	51.09	11	68.58	59.82	13	68.71	55.39
CLB 10	3	63.40	48.86	4	66.20	51.84	12	63.36	62.34	14	74.30	43.22

Table 3.3. The number of admitted streams (#) and the total utilization of the reserved bandwidth and buffer shares in percentages for the fully-specified streams (F.S.), the maximum-leaky-bucket description (Max. LB), the leaky-bucket descriptions using (3.34) consisting of 1, 3, and 5 leaky buckets (ULB 1, ULB 3, ULB 5), and the leaky-bucket descriptions using (3.33) consisting of 5 and 10 leaky buckets (CLB 5, CLB 10), using settings 1–8. For each setting (top row) we present the results for the maximum number of admitted streams (first column/setting), and the total utilization over all streams for the reserved bandwidth and buffer shares, respectively (second and third column/setting).

The second column of Table 3.2 gives the maximum number of streams that could be admitted for each setting. As we suspected, for all settings, this maximum is equal to the maximum number of admitted streams for fully-specified streams. Furthermore, if we compare these utilization results to the results for fully-specified streams given in Table 2.6 on page 54, we notice that they are more or less the same. For settings 2 and 7 even a total bandwidth utilization of almost 4% higher is obtained while also having a slightly higher total buffer utilization. An explanation for this lies in the fact that the sub-problems are not solved in an identical manner. The sub-problems for which a trade-off between bandwidth and buffer size has to be made, can have a whole line segment of optimal solutions. With different approaches, different solutions can be found. As the final solution for a stream is a convex combination of the found solutions, this may result in differences between the solutions.

Table 3.3 compares, using all eight settings, the maximum number of admitted streams and total utilization of the bandwidth and buffers for the fully-specified streams, the maximum-leaky-bucket description, and several other leaky-bucket descriptions. If we first compare the number of streams admitted, we notice that

for settings 3 and 6 the leaky-bucket description consisting of just one leaky bucket (ULB 1) admits the same number of streams as is maximally possible according to the results of the fully specified streams. For the other settings at most 2 streams fewer are admitted using the ULB-1-description. When the number of leaky buckets in the description increases, we see that for these settings also the number of streams admitted increases up to the maximum possible. However, where the description consisting of five leaky buckets using (3.34) (ULB 5) admits for all settings the maximum number of streams possible, the description using (3.33) with five leaky buckets (CLB 5) admits fewer streams for settings 4, 7, and 8. This shows that not only the number of leaky buckets used is important, but also the parameters of the leaky buckets.

A closer look at the leaky-bucket description CLB 5 for each stream revealed that for six of the streams the description for the first $10,000$ time units based on CLB 5 consisted of only two leaky-bucket parameters, while for the other nine streams it consisted of just three leaky-bucket parameters. As the leaky-bucket parameters of ULB 5 are taken from the parameters of the maximum-leaky-bucket description for the first $10,000$ time units, we conclude that for these experiments a good leaky-bucket description for the short or medium term of a stream is far more important than a good leaky-bucket description for the long term. In other words, it is better to have a good approximation of the amount of data that needs to be transmitted in a short period of time than to have a good approximation of the amount of data that needs to be transmitted in total.

Next, if we compare the results concerning the bandwidth and buffer utilization we notice that especially when the number of streams admitted is equal to the maximum possible, the utilization is also comparable to the utilization obtained for fully-specified and maximum-leaky-bucket streams, and in some cases it is even better. An explanation for the latter again lies in the fact that a lot of different solutions exist as MSSP and MLBSSP are feasibility problems. Using different descriptions and different approaches can lead to different solutions where some may give far better utilization results. For example, using fully-specified streams, results may be found where for one stream bandwidth was minimized leading to relatively very high buffer shares, while for another stream the buffer shares were minimized leading to a relatively very high bandwidth share. These streams' bandwidth and buffer shares combined can then lead to a relatively low utilization, while average bandwidth and buffer shares are found using leaky-bucket descriptions leading to a relatively high utilization.

Table 3.4 compares the fully-specified description to all leaky-bucket descriptions by minimization of the bandwidth or a single buffer for all streams. Here, we see that when the bandwidth is minimized, on average the difference between the optimal solutions and the solutions for the leaky-bucket descriptions is relatively

	Min. B with $M = 512,000$, $\delta = 10$			*Min. B with* $M = 4,096,000$, $\delta = 100$			*Min. M_r with* $B = 125,000$, $\delta = 100$, $M_s = 16,384,000$		
	Avg.	*Max. % diff.*	*Min. % diff.*	*Avg.*	*Max. % diff.*	*Min. % diff.*	*Avg.*	*Max. % diff.*	*Min. % diff.*
F.S.	37502			34345			131544		
Max. LB	37502			34345			131544		
ULB 1	40387	52.86	0.329	40355	57.93	4.890	170302	73.34	0
ULB 2	39340	32.69	0.011	35661	13.33	0.229	170302	73.34	0
ULB 3	39340	32.69	0.011	35040	7.43	0.152	170302	73.34	0
ULB 4	37938	9.83	0.006	34889	7.43	0.152	170302	73.34	0
ULB 5	37938	9.83	0.006	34678	3.05	0.006	170302	73.34	0
CLB 5	44939	196.17	0.174	35577	15.02	0.025	311482	808.61	12.505
CLB 6	42204	196.17	0.068	35247	15.02	0.026	268329	808.60	8.432
CLB 7	37759	2.28	0.028	34873	7.00	0.009	196852	183.80	8.432
CLB 8	37911	7.68	0.027	34697	5.93	0.008	185372	105.28	8.432
CLB 9	37723	2.28	0.011	34693	5.93	0.009	168745	75.20	0
CLB 10	37624	1.30	0.011	34428	1.79	0	163424	55.10	0
CLB 20	37521	0.27	0	34349	0.09	0	131544	0	0

Table 3.4. Comparison of the fully-specified description to all leaky-bucket descriptions using bandwidth and buffer minimization. The first column gives the description, with fully specified (F.S.), maximum-leaky-bucket (Max. LB), leaky-bucket descriptions using (3.34) consisting of 1–5 leaky buckets (ULB 1–5), and leaky-bucket descriptions using (3.33) consisting of 5–10 and 20 leaky buckets (CLB 5–10, 20). The top row gives the three different settings (minimizing bandwidth or one buffer given the values of the other resources and delay). The first column for each setting gives the average value obtained using each description over all 15 streams given in Table 2.1 on page 50. The second column gives for each description the maximum difference in percentages compared to the F.S. and Max. LB result, that was observed for a single stream. The third column gives the minimum difference in percentages, that was observed for a single stream.

small, especially for the ULB descriptions with two or more leaky buckets and the CLB descriptions using seven or more leaky buckets. However, for some streams the minimum bandwidth can be 50% (ULB 1) to almost 200% (CLB 5 & 6) larger than the optimum while for other streams the difference with the optimum is less than 1%. The differences are larger when a buffer is minimized, especially for the CLB 5 and CLB 6 descriptions, for which one stream needs an 800% higher buffer than the optimal size. Overall, we see again that the results for the leaky-bucket descriptions using (3.34) (ULB) are generally better than the results for the leaky-bucket descriptions using (3.33) (CLB).

Finally, in line with our supposition of Section 3.4.1, Table 3.4 also shows that again the results for the maximum-leaky-bucket descriptions are equal to the results for the fully-specified streams. However, it is not trivial to show that a solution to an instance of MSSP is also a solution to a corresponding instance of MLBSSP with maximum-leaky-bucket descriptions. Therefore, we conclude this chapter with the following conjecture.

Conjecture 3.1. *Let $\mathcal{I}$ be an instance of MSSP where the demand for each stream is a shifted copy of the supply, i.e., for each $d \in \mathcal{D}$, we have $C_d(t) = P_d(t - \delta_d)$ for a certain $\delta_d \geq 0$. Now, let $\mathcal{I}'$ be the corresponding instance of MLBSSP. So, $\mathcal{N}$, $\mathcal{T}$, $\mathcal{D}$, B, M_n for all $n \in \mathcal{N}$, and s_d, r_d, and δ_d for all $d \in \mathcal{D}$ are equal for $\mathcal{I}$ and $\mathcal{I}'$. Furthermore, for each stream $d \in \mathcal{D}$, the supply-bounding function $f_d(t)$ is given by the maximum-leaky-bucket description of the supply $P_d(t)$, i.e., $f_d(t) = \hat{E}_d(t)$.*

If a solution consisting of bandwidth shares b_d and buffer shares $m_{s_d,d}$ and $m_{r_d,d}$ for each stream $d \in \mathcal{D}$ is a feasible solution to $\mathcal{I}$, then it is a feasible solution to $\mathcal{I}'$. □

Notice that any feasible solution to $\mathcal{I}'$ is also a feasible solution to $\mathcal{I}$, as for each stream $d \in \mathcal{D}$, the supply $P_d(t)$ satisfies (3.1) for function $f_d(t)$ and the demand $C_d(t)$ satisfies (3.2).

4

On-line problem

In this chapter we consider on-line settings of MSSP and MLBSSP. In an on-line setting of a problem, generally at any time there is only partial knowledge of a problem instance, unlike an off-line setting, in which everything is known in advance. For one problem different on-line settings can exist, for which different parts of the problem are unknown and given on-line. For scheduling problems, Sgall [1998] gives several paradigms for classification of on-line problems. The on-line settings of MSSP and MLBSSP that we consider, are based on the on-line paradigm 'jobs arrive over time'. More appropriately, we call it here *streams arrive over time*. In this setting there is no knowledge about future streams, i.e., it is unknown if and when a new stream starts. However, the properties of a stream are known as soon as it starts. The solution of the on-line setting can be used to decide if a new stream can be admitted service given a set of already admitted streams.

For a scheduling problem, an on-line setting is called *clairvoyant* if the running time of a job is known as soon as it arrives, while it is called *non-clairvoyant* if the running time remains unknown until it has finished. Thus, the on-line setting of MSSP is clairvoyant, as the exact supply and demand schemes are known when a stream starts. However, in MLBSSP only an upper bound function f on the supply and demand of a stream during any time window is given. Thus the on-line setting of MLBSSP is clairvoyant w.r.t. the upper bound f, but non-clairvoyant w.r.t. the exact supply and demand schemes. In this respect, MLBSSP itself can be seen as

an on-line setting of MSSP that is non-clairvoyant. It corresponds to the on-line paradigm 'unknown running times' for scheduling problems.

In Section 4.1 we describe how the LP-method for the off-line setting can be used for the on-line setting of MSSP and MLBSSP. In Section 4.2 we show how to adapt the off-line solution methods for the single-stream problems to obtain single-stream solutions for the on-line setting. Finally, in Section 4.3 we present experimental results.

4.1 LP model

In the off-line settings of MSSP and MLBSSP all streams and their description are given in advance and the streams are assumed to run simultaneously. The on-line settings of these problems follow the paradigm 'streams arrive over time', as described before. We introduce some notation for these on-line settings in Section 4.1.1. Then we describe the problem and solution approach in Section 4.1.2. Finally, in Section 4.1.3 we give some objectives that can be used with the LP solution method.

4.1.1 Notation

For consistency we again define the set of time units by $\mathcal{T} = \{1, 2, \ldots, T\}$; however, T is unknown until the last stream has stopped. Each fully-specified or leaky-bucket-controlled stream $d \in \mathcal{D}$ has a start or release time denoted by $\tau_d \in \mathcal{T}$. The release time τ_d of a stream d is unknown at any time unit $t < \tau_d$. At τ_d, the sending and receiving nodes s_d and r_d are given. Furthermore, at τ_d a supply scheme $p_d(t)$ and demand scheme $c_d(t)$ are given for a fully-specified stream, and a supply-bounding function $f_d(t)$ and delay δ_d are given for a leaky-bucket-controlled stream. The end time of a stream, i.e., the time unit in which the final demand of a stream takes place, is denoted by e_d. For a fully-specified stream $e_d \in \mathcal{T}$ is implicitly given by its supply and demand schemes; for a leaky-bucket-controlled stream $e_d \in \mathcal{T}$ is unknown at any time unit $t < e_d$.

4.1.2 On-line problem and solution approach

When a new stream d' starts and requests admission to the network, a solution b, m_s, and m_r has to be determined for it. The solution method has to take into account the streams that have already been admitted to the network, i.e., all streams $d \in \mathcal{D}$ for which $\tau_d < \tau_{d'} \leq e_d$. In the remainder of this chapter we will refer to the already admitted streams as *current streams*. As the characteristics concerning supply and demand of the streams using the network at $\tau_{d'}$ are known, a slightly adjusted form of the solution approach for the off-line setting can be used at τ_d to determine a new solution.

We now consider two variants of the on-line problems. In the first variant the shares of the current streams on the network cannot be changed. Then we have to determine bandwidth and buffer shares only for the new stream, which can be done by generating columns in the master LP tableau for the new stream, i.e., by solving the corresponding single-stream sub-problem. For each current stream d the master LP tableau contains one column for a variable $\lambda_{d,q}$ that corresponds to the current bandwidth and buffer shares of stream d. As no new columns are generated for the current streams, these initial variables $\lambda_{d,q}$ will still have value 1 after optimization of the master LP. Thus, the solutions of the current streams remain preserved. However, this may result in an infeasible solution, while a feasible solution may exist if changing the shares of the current streams is allowed. Especially for fully-specified streams, the required bandwidth and buffer shares may be decreased when the peak of such a stream has already passed the network.

Hence, the second variant that we consider, allows to change the reserved bandwidth and buffer shares of the current streams. However, when new bandwidth and buffer shares are determined for these streams, their transmission states must be taken into account as these pose extra constraints to the sending and receiving buffer shares. For ease of notation we use τ in the remainder of this chapter to denote the start time of a new stream. So $X_d(\tau-1)$, $P_d(\tau-1)$, and $C_d(\tau-1)$ denote the total amount of data transmitted, the total amount of data supplied, and the total amount of data demanded, respectively, for a stream d when a new stream starts. Then,

$$P_d(\tau) - X_d(\tau-1) \leq m_{s_d,d} \;, \qquad \forall\, d \in \mathcal{D} \;,\; \tau_d < \tau \leq e_d \;, \tag{4.1}$$

and

$$X_d(\tau) - C_d(\tau-1) \leq m_{r_d,d} \;, \qquad \forall\, d \in \mathcal{D} \;,\; \tau_d < \tau \leq e_d \;, \tag{4.2}$$

must hold for the new buffer reservations $m_{s_d,d}$ and $m_{r_d,d}$ for the current streams. As these constraints concern only one stream, they are added as extra constraints to the single-stream problems.

When a new stream d' starts, the off-line method can be used as follows. Analogously to the first variant, we begin with a master LP tableau that contains the current reserved bandwidth and buffer shares, i.e., for each current stream d we have one variable $\lambda_{d,q}$ that corresponds to a column with the current bandwidth and buffer shares of stream d. Notice that bandwidth and buffer shares of previously generated columns may not satisfy (4.1) and (4.2), only the current bandwidth and buffer reservations of a current stream are definitely feasible. For the new stream, we generate an initial column by solving its sub-problem. This column can be added to the master LP tableau, which subsequently can be optimized. Notice that for the new stream only one variable is added to the bandwidth and relevant buffer constraints of the master LP. Furthermore, one constraint is added for the solutions

of the new stream. If a feasible solution cannot be found for the new stream by only solving its own sub-problem, we also determine new solutions for the sub-problems of the current streams. In Section 4.2 we show how the solution methods for the sub-problems can be adjusted to incorporate (4.1) and (4.2) for the current streams.

4.1.3 LP objectives

In the on-line setting of MSSP and MLBSSP we do not know in advance if new streams will arrive in the future and what their characteristics are. However, we do know that new streams can arrive in the future, which we would like to give guaranteed shares of the bandwidth and buffers. However, if we do not take into account that new streams might arrive, we may end up with a solution that gives e.g. the total bandwidth capacity to current streams while these streams also could have been serviced with much lower bandwidth reservations. Thus, when we determine a new solution we need to keep in mind that in the future we might want to allocate some of the resources to another stream. If a resource is currently already used by relatively many streams, then a possible way of taking future streams into account is minimizing the total reservation of the heavily-used resource. For this we can add an appropriate objective function to the LP model.

The LP approach described in Chapters 2 and 3 searches for a feasible solution for a given set of streams, and thus initially does not contain an objective function. By adding slack and penalty variables to some constraints we obtained an objective for the feasibility problem such that determining a feasible start solution for the LP was trivial. We will now show how objectives that minimize the total reserved bandwidth or buffer size at a certain node can be used with the presented LP models. Using these objectives in an on-line setting can help to avoid bottlenecks.

We first show how the total bandwidth reservation can be minimized. Obviously, the objective should include $\sum_{d\in\mathcal{D}} b_d$. Furthermore, the buffer capacities should not be exceeded. Hence, the penalty variables for the buffers, i.e., $\sum_{n\in\mathcal{N}} p_n$, should also be included in the objective. Notice that the penalty variable for the bandwidth, p, does not need to be included in the objective as it is already minimized by minimizing the sum of the bandwidth reservations. When we now want to find a feasible solution, we need to be sure that none of the buffer capacities is exceeded, i.e., that $\sum_{n\in\mathcal{N}} p_n = 0$ if a feasible solution exists. For this, we add a coefficient larger than one to each p_n. If a penalty variable is positive, i.e., $p_n > 0$, while a feasible solution exists, then we can decrease the buffer share of a stream at node n by an amount Δ, which will increase the stream's bandwidth share by at most Δ, and thus will lower the objective value. So, if a feasible solution exists, we will get $p_n = 0$ for all n, and next to that, $\sum_{d\in\mathcal{D}} b_d$ will be minimized. So, to

minimize the total reserved bandwidth, we need to minimize

$$\sum_{d\in\mathcal{D}} b_d + 2\sum_{n\in\mathcal{N}} p_n \,,$$

instead of $p+\sum_{n\in\mathcal{N}} p_n$. Notice that a feasible solution does not exist if either $\sum_{d\in\mathcal{D}} b_d > B$ or $\sum_{n\in\mathcal{N}} p_n > 0$ holds in the optimal solution.

Next, we show how to minimize the total reserved buffer share at a single node. Let $\tilde{n}$ be the node at which the total reserved buffer share needs to be minimized. Then the objective should include $\sum_{d\in\mathcal{D}\mid s_d=\tilde{n}} m_{\tilde{n},d} + \sum_{d\in\mathcal{D}\mid r_d=\tilde{n}} m_{\tilde{n},d}$. Furthermore, the available bandwidth should not be exceeded, so the bandwidth penalty variable p needs to be included in the objective. Finally, the reservations of the other buffers may not be exceeded, thus we also need to include $\sum_{n\in\mathcal{N}} p_n$. Notice that in the latter term we may exclude $p_{\tilde{n}}$. If a feasible solution exists, we need to be sure that $p=0$ and $p_n=0$ for all $n\in\mathcal{N}/\{\tilde{n}\}$ in the final solution. As the other buffer share of a stream d will not be increased by more than Δ when $m_{\tilde{n},d}$ is decreased by Δ, it is sufficient to add a coefficient larger than one to each p_n for all $n\neq\tilde{n}$. Furthermore, in the worst-case situation an increase of the bandwidth share for a certain stream by Δ cannot lead to a decrease of one of its buffer shares by more than $T\Delta$, with T the length of the time horizon. For a leaky-bucket-controlled stream T is unknown. However, in that case we can take for T the time unit at the last bending point of the supply-bounding function f. To minimize the total reserved buffer size at a single node, we thus need to minimize

$$\sum_{d\in\mathcal{D}\mid s_d=\tilde{n}} m_{\tilde{n},d} + \sum_{d\in\mathcal{D}\mid r_d=\tilde{n}} m_{\tilde{n},d} + (T+1)p + 2\sum_{n\in\mathcal{N}/\{\tilde{n}\}} p_n \,,$$

instead of $p+\sum_{n\in\mathcal{N}} p_n$. This ensures that first $p=0$, and $p_n=0$ for $n\neq\tilde{n}$, before the buffer reservation at $\tilde{n}$ is minimized.

The above objectives can be used with the Dantzig-Wolfe decomposition as described in Section 2.1.5. The bandwidth share b_d or buffer shares $m_{n,d}$ in the objectives in above equations of the new master LP then need to be replaced by their expressions in the sum of the generated solutions, i.e., if we want to minimize the total reserved bandwidth, we get

$$\min \sum_{d\in\mathcal{D}} \sum_{q=1}^{k_d} \lambda_{d,q} b_d^q + 2\sum_{n\in\mathcal{N}} p_n$$

as objective, and if we want to minimize the total reserved buffer size at a given node $\tilde{n}$, we get

$$\min \sum_{d\in\mathcal{D}\mid s_d=\tilde{n}} \sum_{q=1}^{k_d} \lambda_{d,q} m_{\tilde{n},d}^q + \sum_{d\in\mathcal{D}\mid r_d=\tilde{n}} \sum_{q=1}^{k_d} \lambda_{d,q} m_{\tilde{n},d}^q + (T+1)p + 2\sum_{n\in\mathcal{N}/\{\tilde{n}\}} p_n$$

as objective.

Finally, we consider the minimization of the maximum relative resource reservation as objective. More specifically, we want to minimize

$$\max\{\frac{\sum_d b_d}{B}, \max_{n\in\mathcal{N}} \frac{\sum_{d\in\mathcal{D} \mid s_d=n} m_{n,d} + \sum_{d\in\mathcal{D} \mid r_d=n} m_{n,d}}{M_n}\} .$$

With this objective we try to balance the load of the resource reservations over the bandwidth and buffers with their total capacities in mind. To incorporate this objective in the LP, we add a variable z which represents the maximum relative resource reservation. Thus z must be at least as large as each relative resource reservation, i.e.,

$$z \geq \frac{\sum_d b_d}{B} , \tag{4.3}$$

and,

$$z \geq \frac{\sum_{d\in\mathcal{D} \mid s_d=n} m_{n,d} + \sum_{d\in\mathcal{D} \mid r_d=n} m_{n,d}}{M_n} , \qquad \forall\, n \in \mathcal{N} . \tag{4.4}$$

To minimize the maximum relative resource reservation we now have the objective

$$\min z .$$

Notice that for a feasible solution to MSSP or MLBSSP exists if and only if $z \leq 1$ in the optimal solution. Thus for this objective we can leave out (2.1) and (2.2) for MSSP, and (3.3) and (3.4) for MLBSSP. Instead, we include (4.3) and (4.4). We can now add slack variables s and s_n to these constraints and rewrite them into

$$\frac{\sum_d b_d}{B} - z + s = 0 , \tag{4.5}$$

and,

$$\frac{\sum_{d\in\mathcal{D} \mid s_d=n} m_{n,d} + \sum_{d\in\mathcal{D} \mid r_d=n} m_{n,d}}{M_n} - z + s_n = 0 , \qquad \forall\, n \in \mathcal{N} . \tag{4.6}$$

An example of the corresponding master LP tableau is given in Figure 4.1.

4.2 Single-stream methods revisited

In this section we show how the single-stream methods for fully-specified streams and for leaky-bucket-controlled streams as described in Chapters 2 and 3 can be adjusted to handle a stream that already has been admitted to the network and for which the bandwidth and buffer shares may be altered. Such a stream already has transmitted data, which occupies the buffers. This has to be taken into account when determining new bandwidth and buffer shares for this stream. Notice that for a new stream, the methods as described for the off-line setting can be used, since it does not have any data in the buffers. In Section 4.2.1 we show how to adapt the single-stream methods for fully-specified streams to the on-line setting.

s	s_1	s_2	s_3	s_4	z	$\lambda_{1,1}$	$\lambda_{2,1}$	$\lambda_{3,1}$
					1			
0	1				-1	b_1^1/B	b_2^1/B	b_3^1/B
0		1			-1	$m_{1,1}^1/M_1$	$m_{1,2}^1/M_1$	
0			1		-1	$m_{2,1}^1/M_2$		$m_{2,3}^1/M_2$
0				1	-1		$m_{3,2}^1/M_3$	
0					-1			$m_{4,3}^1/M_4$
1						1		
1							1	
1								1

Figure 4.1. Starting LP tableau of the master LP for the example of Figure 1.2, when the maximum relative resource reservation is minimized. On top are the coefficients for each decision variable in the objective function, followed by the rows corresponding to the constraints concerning the total relative bandwidth reservation, the total relative buffer reservation at each node, and the sum of the values of the λ's.

In Section 4.2.2 we revisit the single-stream methods for leaky-bucket-controlled streams.

4.2.1 Fully-specified stream sub-problem

The single-stream problem for a current fully-specified stream in the on-line setting can be solved using the same methods as in the off-line setting. The extra constraints concerning the initial transmission state $X(\tau-1)$, which determines the amount of data in each of the stream's buffers, can be taken into account as follows.

Obviously, when all cost coefficients are non-positive, the optimal solution remains $b = B$, $m_s = M_s$, and $m_r = M_r$. When the bandwidth needs to be minimized, the MVBA-algorithm as described in Section 2.2.1 can be executed with one difference, viz. transmission starts at $X(\tau-1)$. When one of the buffers needs to be minimized, the algorithm as described in Section 2.2.2 can also be executed. If the resulting buffer size does not satisfy (4.1) for the sending buffer or (4.2) for the receiving buffer, then the minimum buffer size is given by the smallest value that satisfies the appropriate constraint.

In Section 2.2.3 we showed that a joint minimization of two buffers can be obtained by first minimizing the buffer with the highest cost coefficient and then the buffer with the lowest cost coefficient. For the on-line setting, this procedure can be used again taking (4.1) and (4.2) into account. For this, a single buffer is minimized as described above.

For the bandwidth-buffer trade-off algorithm described in Section 2.2.4 trans-

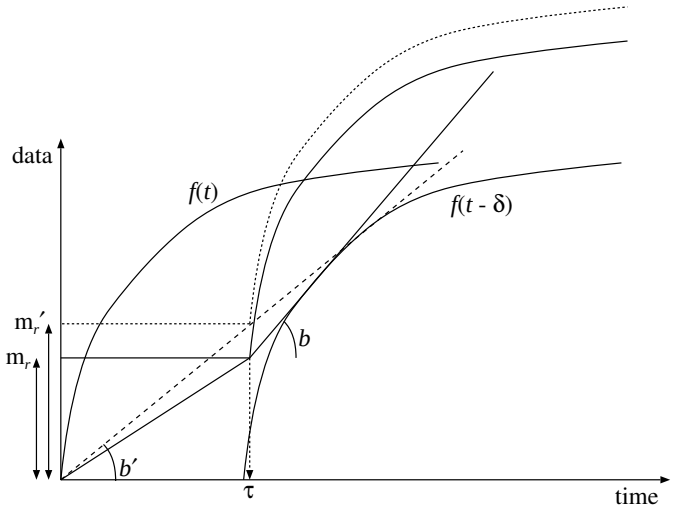

Figure 4.2. Suppose an existing stream supplies according to its upper bound f, and has a bandwidth share b and receiving buffer share m_r. Now a new stream starts at time τ, and the bandwidth and receiving buffer shares of the existing stream are changed to b' and m_r' which satisfies (3.19)–(3.22), (4.1), and (4.2). However, for the initial solution only m_r data could be transmitted up to time τ and the bandwidth share b was required to transmit everything on time after τ. With the new solution b' there will be data arriving too late at the receiving buffer.

mission again is fixed to start at $X(\tau-1)$. Constraints (4.1) and (4.2) form a lower bound on the sending and receiving buffer, respectively. After the initial solution with minimum buffer size has been determined, the trade-off algorithm can be executed as described. Finally, when all cost coefficients are positive, the off-line approach described in 2.2.5 can be used in combination with the joint buffer minimization for the on-line setting.

4.2.2 Leaky-bucket-controlled stream sub-problem

For a leaky-bucket-controlled stream it is not sufficient to use the methods for the off-line setting together with (4.1) and (4.2) as lower bounds on the buffer shares. As the reserved bandwidth share depends on the size of the buffer shares, this may lead to an infeasible solution if the solution is changed; see Figure 4.2 for an example. For a fully-specified stream this problem is circumvented by directly using the initial transmission state $X(\tau-1)$ in the single-stream methods. For a leaky-bucket-controlled stream $X(\tau-1)$ needs to be used together with an upper bound on the supply and demand of data that may occur immediately after a new stream starts.

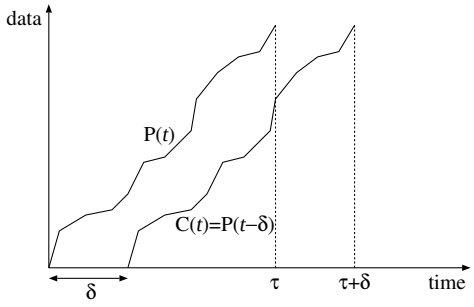

Figure 4.3. Since demand equals supply shifted by a delay δ, the demand up to time unit $\tau+\delta$ is known at time unit τ.

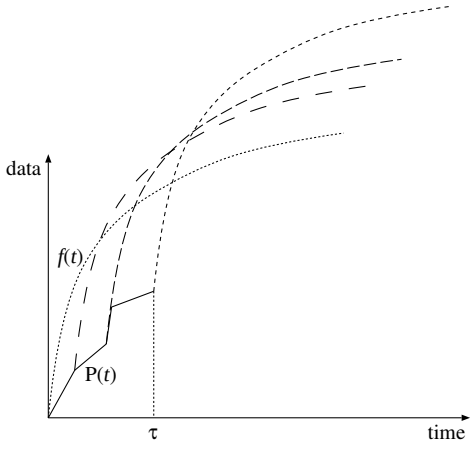

Figure 4.4. For each time unit it holds that the amount of data that can be supplied during the next t time units is bounded by $f(t)$.

Thus, we need to take into account all information we have about possible data supply and demand in the time units after a new stream starts. For a current stream, the supply that has taken place before a new stream starts, is already known. As the demand of a leaky-bucket-controlled stream is shifted with a delay δ compared to the supply, the demand of the first δ time units after $\tau-1$ is also known, see Figure 4.3. So, for a solution it must be possible to transmit data in time for the known demand $C(t)$, i.e., for $\tau \le t \le \tau+\delta-1$,

$$C(t) - X(\tau-1) \le (t-\tau+1)b \tag{4.7}$$

must hold. Furthermore, the buffer share m_r may limit the amount of data that can be transmitted during a time unit. If the receiving buffer share is full at the end of a time unit q, then $C(q-1)+m_r$ data has been transmitted in total and any amount of data above this amount that is required for later time units, needs to be transmitted after time unit q. So, for $\tau \le q \le \tau+\delta-1$ and $q \le t \le \tau+\delta-1$,

$$C(t) - C(q-1) - m_r \le (t-q)b \tag{4.8}$$

must hold. Notice that for $t = q$ this constraint just requires that the buffer share m_r is large enough to hold the demand for time unit q.

The supply at time units $t \ge \tau$ and the demand at time units $t \ge \tau+\delta$ is unknown. We derive an upper bound for this using the realized supply up to time unit τ. For each time unit it holds that the total supply and demand during the next t time units is bounded by $f(t)$; see Figure 4.4. So, if a stream starts at

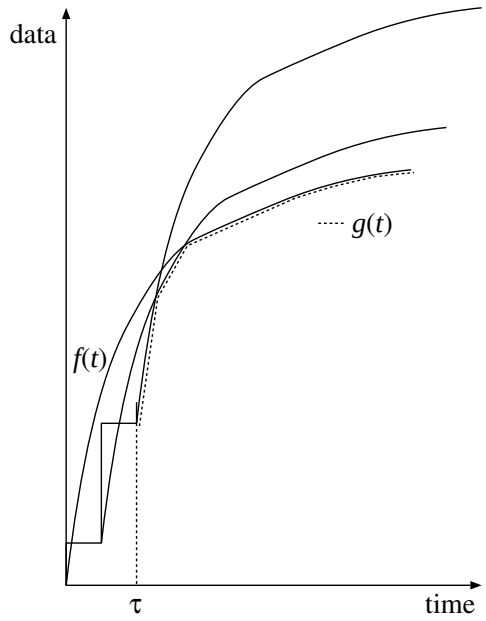

Figure 4.5. An upper bound on the data supply composed by taking the minimum of all upper bounds consisting of the function f starting at cumulative supply amounts.

time unit 1, the data supply at a time unit $t \geq \tau$ has τ upper bounds, viz. $f(t)$, $P(1) + f(t-1)$, $P(2) + f(t-2)$, ..., and $P(\tau-1) + f(t-\tau+1)$. If we take the minimum of these τ upper bounds for all $t \geq \tau$, we get a new upper bound g, i.e., $g(t) = \min\{P(t') + f(t-t') \mid t' = 0, \ldots, \tau-1\}$; see Figure 4.5. As g is the minimum of τ piecewise-linear, concave functions, g is also a piecewise-linear, concave function. Furthermore, since g is the minimum of τ shifted functions f, the gradient of a line segment of g must be equal to the gradient of a line segment of f. Therefore, the number of bending points of g is at most the number of bending points of f. Function g can be determined at once at time unit τ, which requires taking the minimum of τ functions, or it can be constructed on-line by adjusting it each time unit t' for the new upper bound $P(t') + f(t-t')$ for $t > t'$, which requires taking the minimum of two functions during each time unit.

Using upper bound g we can derive the following constraints for a solution of an existing stream, analogously to (3.19)–(3.22), (4.7), and (4.8). The bandwidth share should be large enough such that data can be transmitted before it is demanded and before the sending buffer share overflows. So, for $t \geq 1$

$$g(t+\tau-1) - X(\tau-1) \leq (t+\delta)b \tag{4.9}$$

and

$$g(t+\tau-1)-X(\tau-1)-(t-1)b \leq m_s \tag{4.10}$$

must hold, cf. (3.19) and (3.20). Furthermore, as the receiving buffer share can be full at the end of a time unit q, this may limit the amount of data transmitted up to and including q to $C(q-1)+m_r$. Then, $C(q-1)+m_r-X(\tau-1)$ data is transmitted in the first $q-(\tau-1)$ time units after the new stream starts. Thus, similar to the derivation of (3.21) and (3.22) from (3.19) and (3.20), respectively, we derive the following constraints from (4.9) and (4.10). If the receiving buffer share is full at the end of a time unit q, then, in order to transmit data before it is demanded, we have for $t \geq 1$,

$$g(t+\tau-1)-X(\tau-1)-(C(q-1)+m_r-X(\tau-1)) \leq (t+\delta-(q-(\tau-1)))b\,.$$

As q can be any time unit from τ to $\tau+\delta$, we get that for $\tau \leq q \leq \tau+\delta$ and $t \geq 1$,

$$g(t+\tau-1)-C(q-1)-m_r \leq (t+\delta-q+\tau-1)b \tag{4.11}$$

must hold. Note that this constraint is correctly defined for all $t \geq 1$.

Next, if the receiving buffer share is full at the end of a time unit q, then, in order to transmit data before the sending buffer overflows, we have for $t \geq 1+(q-(\tau-1))$,

$$g(t+\tau-1)-X(\tau-1)-(C(q-1)+m_r-X(\tau-1))-(t-1-(q-(\tau-1)))b \leq m_s\,.$$

As q again can be any time unit from τ to $\tau+\delta$, we get that for $\tau \leq q \leq \tau+\delta$ and $t \geq q-\tau+2$,

$$g(t+\tau-1)-C(q-1)-m_r-(t-q+\tau-2)b \leq m_s \tag{4.12}$$

must hold. Note that for this constraint we require $t \geq q-\tau+2$ instead of $t \geq 1$. Furthermore, notice that (4.11) and (4.12) follow from (4.9) and (4.10), respectively, by adjusting the amount of data in the constraints from $g(t+\tau-1)-X(\tau-1)$ to $g(t+\tau-1)-C(q-1)-m_r$ and subtracting $q-(\tau-1)$ from the number of time units during which data can be transmitted, which is given in the equations directly in front of b. Also notice that for $t=q+1-(\tau-1)$, (4.12) reduces to $g(q+1)-m_s \leq C(q-1)+m_r$, i.e., the upper bound induced by the receiving buffer share must lie above the lower bound induced by the sending buffer share.

The constraints (4.9)–(4.12) describe the worst-case situation for supply and demand of a current stream as observed at the beginning of time unit τ. However, generally the actual supply and demand of data will be less than the amount given by g, and the worst-case situation for the supply and demand as described by f may take place at the beginning of a time unit $t' > \tau$. Therefore, to take all possible realizations of the supply after time unit τ into account, a new solution for a current stream still has to satisfy (3.19)–(3.22).

Finally, we consider (4.1) and (4.2) for current leaky-bucket-controlled streams. A new solution should also satisfy these constraints. However, for (4.1) the value of $P(\tau)$ is unknown and only an upper bound value $g(\tau)$ is given. If we replace $P(\tau)$ by $g(\tau)$ in (4.1) we get (4.10) with $t = 1$. For (4.2) the value of $X(\tau)$ is unknown but we know that $X(\tau) \geq X(\tau-1)$ must hold. Combining this with (4.2) we have instead the following lower bound for the new receiving buffer share, viz.

$$m_r \geq X(\tau-1) - C(\tau-1)\,. \tag{4.13}$$

So, a solution for a current leaky-bucket-controlled stream has to satisfy (3.19)–(3.22) and (4.7)–(4.13). We next show that a solution that satisfies these constraints is also a solution to (3.5)–(3.9) with given values of $X(t)$, $P(t)$, and $C(t)$ for all $t \leq \tau-1$, and for all actual supply schemes $p(t)$ that satisfy (3.1) and all actual demand schemes $c(t)$ that satisfy (3.2). Notice that (3.2) fixes the values of $C(t)$ for $\tau \leq t \leq \tau+\delta-1$.

Theorem 4.1. *If a solution b, m_s, and m_r is feasible w.r.t. (3.19)–(3.22) and (4.7)–(4.13), it is also feasible w.r.t. (3.5)–(3.9) with given values of $X(t)$ and $P(t)$ for all $t \leq \tau-1$, and of $C(t)$ for all $t \leq \tau+\delta-1$.*

Proof. The proof of this theorem is similar to the proofs of Theorems 3.1 and 3.2 on pages 62 and 69, respectively. Therefore, when almost identical derivations are made, we refer to the proofs of these theorems and indicate any important differences.

Let b, m_s, and m_r be a feasible solution w.r.t. (3.19)–(3.22) and (4.7)–(4.13). First we show that this solution satisfies constraints similar to (3.12)–(3.16). We consider the following transformations $f = f'$ of the function f in (3.12)–(3.16), viz. $f'(t) = f(t-(\tau-1)) + X(\tau-1)$ and $f'(t) = g(t)$. So, we have for $f(t-(\tau-1)) + X(\tau-1)$ with $t \geq \tau$,

$$\begin{aligned}
x(t) &\leq b\,, && (4.14)\\
f(t-(\tau-1)) + X(\tau-1) - X(t) &\geq 0\,, && (4.15)\\
f(t-(\tau-1)) + X(\tau-1) - X(t-1) &\leq m_s\,, && (4.16)\\
X(t+\delta) - f(t-(\tau-1)) - X(\tau-1) &\geq 0\,, && (4.17)\\
X(t+\delta+1) - f(t-(\tau-1)) - X(\tau-1) &\leq m_r\,, && (4.18)
\end{aligned}$$

and for $g(t)$ with $t \geq \tau$,

$$\begin{aligned}
x(t) &\leq b\,, && (4.19)\\
g(t) - X(t) &\geq 0\,, && (4.20)\\
g(t) - X(t-1) &\leq m_s\,, && (4.21)\\
X(t+\delta) - g(t) &\geq 0\,, && (4.22)\\
X(t+\delta+1) - g(t) &\leq m_r\,. && (4.23)
\end{aligned}$$

Now we first show that any solution to (3.19)–(3.22) is also a solution to (4.14)–(4.18). Analogously to the proof of Theorem 3.2 we consider the greedy transmission strategy $x_f(t) = \min\{b, f(t-(\tau-1)) + X(\tau-1) - X_f(t-1), m_r + f(t-(\tau-1)-\delta-1) + X(\tau-1) - X_f(t-1)\}$. It then immediately follows that (4.14), (4.15), and (4.18) are satisfied. Let $t \geq \tau$. To show that (4.16) is satisfied we choose $\tilde{k} = \max\{k \in \{\tau,\ldots,t-1\} \,|\, x_f(k) = f(k-(\tau-1)) + X(\tau-1) - X_f(k-1) \vee x_f(k) = m_r + f(k-(\tau-1)-\delta-1) + X(\tau-1) - X_f(k-1)\}$. By considering three cases, namely $\tilde{k}$ is undefined, $x_f(\tilde{k}) = f(\tilde{k}-(\tau-1)) + X(\tau-1) - X_f(\tilde{k}-1)$, and $x_f(\tilde{k}) = m_r + f(\tilde{k}-(\tau-1)-\delta-1) + X(\tau-1) - X_f(\tilde{k}-1)$, we can make similar derivations as in the proof of Theorem 3.2. In these derivations we use the fact that we can write $X_f(t-1) = X_f(\tau-1) + \sum_{k=\tau}^{t-1} x_f(k)$ for $t \geq \tau$.

To show that (4.17) is satisfied we choose $\hat{k} = \max\{k \in \{\tau,\ldots,t\} \,|\, x_f(k) = f(k-(\tau-1)) + X(\tau-1) - X_f(k-1) \vee x_f(k) = m_r + f(k-(\tau-1)-\delta-1) + X(\tau-1) - X_f(k-1)\}$. Again, we consider three cases for which we make derivations analogous to the proof of Theorem 3.2.

Next, we show that any solution to (4.7)–(4.13) is also a solution to (4.19)–(4.23). We consider the greedy transmission strategy $x_g(t) = \min\{b, g(t) - X_g(t-1), m_r + g(t-\delta-1) - X_g(t-1)\}$. Again, it immediately follows that (4.19), (4.20), and (4.23) are satisfied. For (4.21) we choose $\tilde{k} = \max\{k \in \{\tau,\ldots,t-1\} \,|\, x_g(k) = g(k) - X_g(k-1) \vee x_g(k) = m_r + g(k-\delta-1) - X_g(k-1)\}$. The derivation for this constraint is similar to the proof of Theorem 3.2, but not trivial. Therefore, we show how it can be exactly made. We again consider three cases, viz. (i) $\tilde{k} = -\infty$, (ii) $x_g(\tilde{k}) = g(\tilde{k}) - X_g(\tilde{k}-1)$, and (iii) $x_g(\tilde{k}) = m_r + g(\tilde{k}-\delta-1) - X_g(\tilde{k}-1)$.

(i) If $\tilde{k} = -\infty$, then $X_g(t-1) = X(\tau-1) + (t-\tau)b$. We derive

$$\begin{aligned} g(t) - X_g(t-1) &= g(t) - X(\tau-1) - (t-\tau)b \\ \{\, t = t' + \tau - 1 \,\} &= g(t'+\tau-1) - X(\tau-1) - (t'+\tau-1-\tau)b \\ \{\, \textit{use (4.10)} \,\} &\leq m_s\,. \end{aligned}$$

(ii) If $x_g(\tilde{k}) = g(\tilde{k}) - X_g(\tilde{k}-1)$, then $X_g(\tilde{k}) = g(\tilde{k})$. We derive

$$\begin{aligned} g(t) - X_g(t-1) &= g(t) - X_g(\tilde{k}) - \sum_{k=\tilde{k}+1}^{t-1} x_g(k) \\ &= g(t) - g(\tilde{k}) + X(\tau-1) - X(\tau-1) - (t-\tilde{k}-1)b \\ \{\, X(\tau-1) \leq g(\tau-1) \,\} &\leq g(t) - g(\tilde{k}) + g(\tau-1) - X(\tau-1) - (t-\tilde{k}-1)b \\ \{\, \textit{concavity of } g \,\} &\leq g(t+\tau-1-\tilde{k}) - X(\tau-1) - (t-\tilde{k}-1)b \\ \{\, \textit{use (4.10)} \,\} &\leq m_s\,. \end{aligned}$$

(iii) If $x_g(\tilde{k}) = m_r + g(\tilde{k}-\delta-1) - X_g(\tilde{k}-1)$, then $X_g(\tilde{k}) = m_r + g(\tilde{k}-\delta-1)$. We derive

$$\begin{aligned}
& g(t) - X_g(t-1) \\
= \;& g(t) - X(\tilde{k}) - \sum_{k=\tilde{k}+1}^{t-1} x(k) \\
& \quad \{ x(k) = b \text{ for } k > \tilde{k} \} \\
= \;& g(t) - m_r - g(\tilde{k}-\delta-1) - (t-\tilde{k}-1)b \\
= \;& g(t) + C(\tau+\delta-1) - C(\tau+\delta-1) - m_r - g(\tilde{k}-\delta-1) - (t-\tilde{k}-1)b \\
& \quad \{C(\tau+\delta-1) = P(\tau-1) \leq g(\tau-1)\} \\
\leq \;& g(t) + g(\tau-1) - g(\tilde{k}-\delta-1) - C(\tau+\delta-1) - m_r - (t-\tilde{k}-1)b \\
& \quad \{ \text{ concavity of } g \} \\
\leq \;& g(t+\tau-1-\tilde{k}+\delta+1) - C(\tau+\delta-1) - m_r - (t-\tilde{k}-1)b \\
& \quad \{ t' = t-\tilde{k}+\delta+1 \} \\
= \;& g(t'+\tau-1) - C(\tau+\delta-1) - m_r - (t'+\tilde{k}-\delta-1-\tilde{k}-1)b \\
\leq \;& g(t'+\tau-1) - C(\tau+\delta-1) - m_r - (t'-\delta-2)b \\
& \quad \{ \text{(4.12) for } q = \tau+\delta \} \\
\leq \;& m_s \, .
\end{aligned}$$

For (4.22) we choose $\hat{k} = \max\{k \in \{\tau,\ldots,t\} \,|\, x_g(k) = g(k) - X_g(k-1) \vee x_g(k) = m_r + g(k-\delta-1) - X_g(k-1)\}$. Again, for we consider three cases and make derivations analogous to the proof of Theorem 3.2 and to the derivations above.

Now we need to show that if b, m_s, and m_r satisfy (4.14)–(4.23), then they are feasible w.r.t. (3.5)–(3.9) with given values of $X(t)$ and $P(t)$ for all $t \leq \tau-1$, and of $C(t)$ for all $t \leq \tau+\delta-1$. For this we take a similar approach as in the proof of Theorem 3.1 on page 62. We consider the greedy transmission strategy $x(t) = \min\{b, P(t) - X(t-1), m_r + C(t-1) - X(t-1)\}$ for any actual supply P satisfying (3.1) and for any actual demand C satisfying (3.2). It then immediately follows that (3.5), (3.6), and (3.9) are satisfied. For (3.7) and (3.8) we can make similar derivations as in the proof of Theorem 3.1.

Let $t \geq \tau$. We now choose $\tilde{k} = \max\{k \in \{\tau,\ldots,t-1\} \,|\, x(k) = P(k) - X(k-1) \vee x(k) = m_r + C(k-1) - X(k-1)\}$, so for all k with $\tilde{k} < k < t$, $x(k) = b$ holds. We first show that (3.7) is satisfied, i.e., $P(t) - X(t-1) \leq m_s$. For this we distinguish the same three possibilities as for Theorem 3.1. If $\tilde{k} = -\infty$, then $X(t-1) = X(\tau-1) + (t-\tau)b$, and the same derivation can be made using (4.19)–(4.23) involving function g. This is needed as $g(t)$ gives an upper bound on $P(t)$ but $f(t-(\tau-1)) +$

$X(\tau-1)$ does not. Analogously, if $x(\tilde{k}) = P(\tilde{k}) - X(\tilde{k}-1)$, or $x(\tilde{k}) = m_r + C(\tilde{k}-1) - X(\tilde{k}-1)$, the same derivations for these cases can be made as in the proof of Theorem 3.1, here using (4.14)–(4.18) involving function $f(t-(\tau-1))+X(\tau-1)$.

Similarly, for (3.8) we take $\hat{k} = \max\{k \in \{\tau,\ldots,t\} \mid x(k) = P(k) - X(k-1) \vee x(k) = m_r + C(k-1) - X(k-1)\}$, so for all k with $\hat{k} < k \leq t$, $x(k) = b$ holds. When $\hat{k} = -\infty$, and thus $X(t-1) = X(\tau-1) + (t-\tau)b$, we make the same derivation as in the proof of Theorem 3.1, but now using (4.19)–(4.23). For $x(\hat{k}) = P(\hat{k}) - X(\hat{k}-1)$ and $x(\hat{k}) = m_r + C(\hat{k}-1) - X(\hat{k}-1)$ the same derivations are made again using (4.14)–(4.18). □

Next, we distinguish the different possibilities of the single-stream problems and describe how they can be solved. For these solution methods we first rewrite (4.9)–(4.12) into constraints on a single resource given values of the other two resources. For the bandwidth share b this leads to the following constraints. From (4.9) it follows that for $\tilde{t} \geq \delta + 1$ with $\tilde{t} = t + \delta$,

$$b \geq (g(\tilde{t} + \tau - 1 - \delta) - X(\tau-1))/\tilde{t}\,. \tag{4.24}$$

From (4.10) it follows that for $\tilde{t} \geq 1$ with $\tilde{t} = t - 1$,

$$b \geq (g(\tilde{t} + \tau) - X(\tau-1) - m_s)/\tilde{t}\,. \tag{4.25}$$

Notice that $t = 1$ in (4.10) leads to a constraint that does not involve b; therefore, $\tilde{t} = 0$ is not included in this constraint. We transform (4.11) into two constraints on b depending on the value of q. For $\tau \leq q \leq \tau + \delta - 1$ and $\tilde{t} \geq \tau + \delta - q$ with $\tilde{t} = t + \delta - q + \tau - 1$, it follows that,

$$b \geq (g(\tilde{t} - \delta + q) - C(q-1) - m_r)/\tilde{t}\,, \tag{4.26}$$

and for $q = \tau + \delta$ and $\tilde{t} \geq 1$ with $\tilde{t} = t - 1$, it follows that,

$$b \geq (g(\tilde{t} + \tau) - C(\tau + \delta - 1) - m_r)/\tilde{t}\,. \tag{4.27}$$

Again, notice that $t = 1$ and $q = \tau + \delta$ in (4.11) leads to a constraint that does not involve b, thus excluding $\tilde{t} = 0$ from (4.27). From (4.12) it follows that for $\tau \leq q \leq \tau + \delta$ and $\tilde{t} \geq 1$ with $\tilde{t} = t - q + \tau - 2$,

$$b \geq (g(\tilde{t} + q + 1) - C(q-1) - m_s - m_r)/\tilde{t}\,. \tag{4.28}$$

For the sending buffer share m_s we can rewrite (4.9)–(4.12) into the following constraints. Constraint (4.10) already states that for $\tilde{t} \geq 1$ with $\tilde{t} = t$,

$$m_s \geq g(\tilde{t} + \tau - 1) - X(\tau-1) - (\tilde{t}-1)b\,. \tag{4.29}$$

From (4.12) it follows that for $\tau \leq q \leq \tau + \delta$ and $\tilde{t} \geq 1$ with $\tilde{t} = t - 1 - q + \tau$,

$$m_s \geq g(\tilde{t} + q) - C(q-1) - m_r - (\tilde{t}-1)b\,. \tag{4.30}$$

For the receiving buffer share m_r we obtain the following constraints. From

(4.11) it follows that for $\tau \leq q \leq \tau+\delta$ and $\tilde{t} \geq \tau+\delta-q+1$ with $\tilde{t} = t+\delta-q+\tau$,

$$m_r \geq g(\tilde{t}-\delta+q-1) - C(q-1) - (\tilde{t}-1)b\,. \tag{4.31}$$

From (4.12) it follows that for $\tau \leq q \leq \tau+\delta$ and $\tilde{t} \geq 1$ with $\tilde{t} = t-1-q+\tau$,

$$m_r \geq g(\tilde{t}+q) - C(q-1) - m_s - (\tilde{t}-1)b\,. \tag{4.32}$$

Finally, we can rewrite (4.12) into a constraint for the total buffer share m_s+m_r, which gives for $\tau \leq q \leq \tau+\delta$ and $\tilde{t} \geq 1$ with $\tilde{t} = t-1-q+\tau$,

$$m_s + m_r \geq g(\tilde{t}+q) - C(q-1) - (\tilde{t}-1)b\,. \tag{4.33}$$

In the rest of this chapter we will use variable t instead of $\tilde{t}$ when we refer to (4.24)–(4.33); $\tilde{t}$ is only used above to clarify the rewriting of (4.9)–(4.12).

Bandwidth minimization.

When the bandwidth needs to be minimized, the minimum value of b that satisfies (3.23)–(3.26), (4.7), (4.8), and (4.24)–(4.28), with $m_s = M_s$ and $m_r = M_r$ needs to be determined. For time units $t = \tau,\ldots,\tau+\delta-1$ the exact demand scheme $C(t)$ is known. If we take the worst-case supply $g(t)$ as actual supply scheme for these time units, we get an upper bound $U(t) = \min\{g(t), C(t-1)+M_r\}$ and a lower bound $L(t) = \{C(t), g(t+1)-M_s, X(\tau-1)\}$. The minimum required bandwidth during time units $t = \tau,\ldots,\tau+\delta-1$ can now be determined using the MVBA-algorithm as described in Section 2.2.1 with start point $X(\tau-1)$ and end point $L(\tau+\delta-1)$. This bandwidth share satisfies (4.7) and (4.8) as well as (4.25) for $1 \leq t \leq \delta$ and (4.28) for $\tau \leq q \leq \tau+\delta-2$ and $1 \leq t \leq \tau+\delta-q-1$.

For (3.23)–(3.26), (4.24), and (4.25) the minimum required value of b can be determined by taking the maximum over all time units t. As both f and g are piecewise-linear, concave functions in case of leaky-bucket-controlled streams, the minimum of a constraint is found at a bending point.

Constraints (4.26)–(4.28) involve time units $q = \tau,\ldots,\tau+\delta$. However, generally not all time units $\tau,\ldots,\tau+\delta$ need to be considered to determine the minimum value of b that satisfies these constraints. Instead we first consider the MVBA-schedule $X(t)$ with U and L as defined above, that starts at $X(\tau-1)$ and ends at $C(\tau+\delta-1)+M_r$. Notice that this schedule is equal to the MVBA-schedule that ends at $L(\tau+\delta-1)$ up to the last convex critical point of the latter schedule.

Let t' be the last concave critical point of the MVBA-schedule $X(t)$ ending in time unit $\tau+\delta$ at $C(\tau+\delta-1)+M_r$, i.e., $t' = \max\{t \mid \tau-1 \leq t \leq \tau+\delta-1 \wedge x(t) > x(t+1)\}$; notice that $X(t') = L(t')$. Then it is sufficient to consider only time units $q > t'$ which end at a convex m_r-critical point of the MVBA-schedule, i.e., $\{q \mid t' < q \leq \tau+\delta \wedge X(q) = C(q-1)+M_r \wedge (x(q) < x(q+1) \vee q = \tau+\delta)\}$. For example, in Figure 4.6 we only need to consider the time units q_1 and $\tau+\delta$. For each of these values of q, the minimum value of b that satisfies (4.26)–(4.28) can

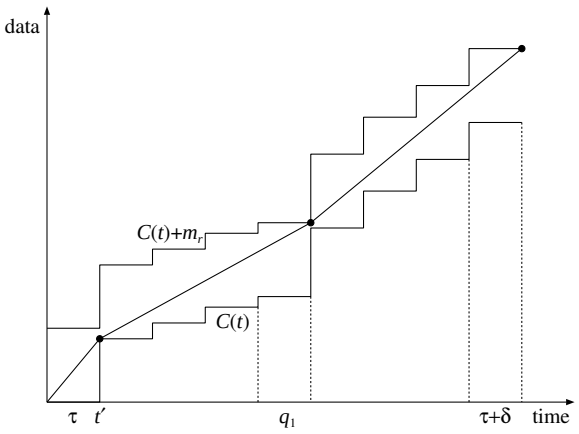

Figure 4.6. MVBA-schedule that starts at $X(\tau-1)$ and ends at $C(\tau+\delta-1)+m_r$. t' is the last concave critical point. Then only the time units after t' that end with a convex critical point need to be considered for (4.26)–(4.28), in this case $q=q_1$ and $q=\tau+\delta$.

be determined by considering the bending points of g. To see why other time units do not need to be considered as value of q in (4.26)–(4.28) we note the following.

First, we consider a convex critical point q' before the last concave critical point t'. Notice that then $\frac{C(\tau+\delta-1)+M_r-(C(q'-1)+M_r)}{\tau+\delta-q'} < \frac{L(t')-(C(q'-1)+M_r)}{t'-q'}$ holds. Now let transmission start at $C(q'-1)+M_r$ and let bandwidth b' be large enough to satisfy (4.26)–(4.28) for $q=q'$. If $b' < \frac{C(\tau+\delta-1)+M_r-(C(q'-1)+M_r)}{\tau+\delta-q'}$, i.e., the receiving buffer size is more constraining at q' than at $\tau+\delta$, then we also have $b' < \frac{L(t')-(C(q'-1)+M_r)}{t'-q'}$, which means that the bandwidth share resulting from either (4.8) with $q=q'$ and $t=t'$ or (4.28) with $q=q'$ and $t=t'-q'$, is larger than b'. If $b' \geq \frac{C(\tau+\delta-1)+M_r-(C(q'-1)+M_r)}{\tau+\delta-q'}$, then the bandwidth share resulting from (4.26)–(4.28) for $q=\tau+\delta$ is larger than b'. Thus, we do not need to consider q'.

Next, we consider a point $q' > t'$ that is not a convex critical point. Again, let transmission start at $C(q'-1)+M_r$ and let bandwidth b' be large enough to satisfy (4.26)–(4.28) for $q=q'$. Furthermore, let $\hat{q}$ be a succeeding convex critical point. Note that such a $\hat{q}$ always exists, since $\hat{q}=\tau+\delta$ is defined as a convex critical point. Now, let $\hat{b}$ be large enough to satisfy (4.26)–(4.28) for $q=\hat{q}$. If $b' < \hat{b}$ then constraints (4.26)–(4.28) are more constraining for $q=\hat{q}$ than for $q=q'$. Now suppose $b' \geq \hat{b}$. Note that then $b' \leq \frac{C(\hat{q}-1)+M_r-(C(q'-1)+M_r)}{\hat{q}-q'}$. We consider the preceding critical point of q', say $\hat{t}$. If $\hat{t}$ is a convex critical point, then $\frac{C(\hat{q}-1)+M_r-(C(q'-1)+M_r)}{\hat{q}-q'} < \frac{C(q'-1)+M_r-(C(\hat{t}-1)+M_r)}{q'-\hat{t}}$ holds. This means that (4.26)–

(4.28) for $q = \hat{t}$ leads to a higher bandwidth than b'. If $\hat{t}$ is a concave critical point, i.e., $\hat{t} = t'$, then the bandwidth required immediately before t', which results from (4.7), (4.8) with $t = t'$, or (4.28) with $t = t' - q$, is larger than $\frac{C(\hat{q}-1)+M_r-L(t')}{\hat{q}-t'}$, i.e., the gradient of the segment between $\hat{q}$ and t'. As $b' \leq \frac{C(\hat{q}-1)+M_r-L(t')}{\hat{q}-t'}$ must hold, this means we do not need to consider q'.

Now we have shown how for each constraint the minimum bandwidth b can be determined such that the constraint is satisfied. The minimum required bandwidth is naturally given by the maximum of these obtained bandwidth shares.

Buffer minimization.

First we consider the minimization of the sending buffer share. For this, we need to determine the minimum value of m_s that satisfies (3.27), (3.28), (4.29), and (4.30), with $b = B$ and $m_r = M_r$. The first three of these constraints only require determining the maximum over all time units t. With f and g piecewise-linear, concave functions, this can be found again at a bending point. Constraint (4.30) actually consists of $\delta + 1$ constraints for which the maximum over all time units t needs to be taken. However, as the maximum occurs at a bending point of g which for a given value of b will always be the same, the maximum for each of these $\delta + 1$ constraints can immediately be determined after the concerned bending point has been found. The resulting sending buffer share is found by taking the maximum over all obtained sending buffer share values.

Next we consider the minimization of the receiving buffer share. For this, we need to determine the minimum value of m_r that satisfies (3.29), (3.30), (4.8), (4.13), (4.31), and (4.32), with $b = B$ and $m_s = M_s$. For (3.29) and (3.30) we need to determine the maximum over all time units t, which can be found at a bending point of f. For (4.8) we can use backward RCBS as described in Section 2.2.2. For (4.13) the minimum value of m_r immediately follows. Constraints (4.31) and (4.32) again consist of $\delta + 1$ constraints, for which the maximum over all time units t can be easily found by determining the bending point of g where the maximum is attained for the given b. The resulting receiving buffer share is the maximum over all found minimum receiving buffer shares.

Two-buffer minimization.

When both buffers need to be minimized, we apply the same approach as for the off-line setting of MSSP and MLBSSP. First, we minimize the buffer with the highest cost coefficient, then we minimize the buffer with the lowest cost coefficient. For the minimization of each buffer we can use the procedures described above.

Bandwidth-buffer trade-off.

We start with discussing how to handle the trade-off between the bandwidth and the sending buffer share. Then we show how the trade-off between the bandwidth and the receiving buffer share can be performed, and between the bandwidth and both buffers. For all trade-offs we start with a solution with minimum bandwidth share b and corresponding values of m_s and m_r, which we obtain by first minimizing b for $m_s = M_s$ and $m_r = M_r$ and then minimizing m_s and m_r for the found value of b. Then we increase b and determine the effect on m_s and m_r.

Trade-off between b and m_s.

For a given value of b, m_s needs to satisfy (3.27), (3.28), (4.29), and (4.30) with $m_r = M_r$. For this, we define analogously to the trade-off described in Chapter 3 the following lower bounds on m_s as functions of b, which correspond to the right-hand-sides of each of the mentioned constraints.

$$\begin{aligned}
m_1(b) &= \max\{ f(t) - (t-1)b \mid t \geq 1\} , \\
m_2(b) &= \max\{ f(t+\delta+1) - M_r - (t-1)b \mid t \geq 1\} , \\
m_3(b) &= \max\{g(t+\tau-1) - X(\tau-1) - (t-1)b \mid t \geq 1\} , \\
m_4^q(b) &= \max\{g(t+q) - C(q-1) - M_r - (t-1)b \mid t \geq 1\} ,
\end{aligned}$$

with $q = \tau, \ldots, \tau+\delta$ for $m_4^q(b)$. Each of these $\delta+4$ lower bound functions forms a piecewise-linear, convex curve in the (b, m_s)-plane. The trade-off curve for b and m_s is given by the maximum of these lower bound curves; see Figure 4.7. Again, we construct the lower bound curves iteratively while increasing b. Furthermore, we determine the trade-off curve efficiently by using the expressions of the lower bounds as we show hereafter.

For all values of b the corresponding value of t at which the above-defined lower bounds attain their maximum, are given by $t_1(b)$, $t_2(b)$, $t_3(b)$, and $t_4^q(b)$, respectively. So we have for a given b,

$$\begin{aligned}
m_1(b) &= f(t_1(b)) - (t_1(b)-1)b , \\
m_2(b) &= f(t_2(b)+\delta+1) - M_r - (t_2(b)-1)b , \\
m_3(b) &= g(t_3(b)+\tau-1) - X(\tau-1) - (t_3(b)-1)b , \\
m_4^q(b) &= g(t_4^q(b)+q) - C(q-1) - M_r - (t_4^q(b)-1)b .
\end{aligned}$$

As the functions f and g are piecewise linear and concave, the maximum values for a given b are attained at the bending points of f and g that connect the line segment with the lowest gradient larger than b to the line segment with the highest gradient smaller than or equal to b. When b is increased and becomes equal to the gradient

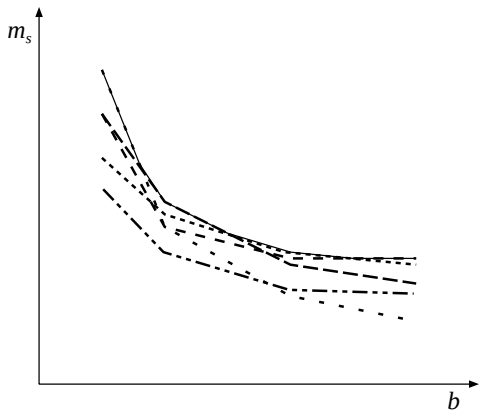

Figure 4.7. The solid trade-off curve between b and m_s is given by the maximum of the dashed lower bounds m_1, m_2, m_3, and m_4^q.

of the line segment preceding such a bending point, the corresponding lower bound values shift along the line segment to the bending point at the beginning of the segment. Therefore, $t_1(b)$, $t_2(b)$, $t_3(b)$, and $t_4^q(b)$ are piecewise-constant functions of b, that jump in value when b is equal to the gradient of a line segment of f and g. If for a certain lower bound the value of t corresponding to the bending point at which the maximum is attained, is less than 1 and, therefore, is infeasible, then this lower bound attains its value at $t = 1$ and remains constant for any further increase of b.

Next, we denote the gradients of the lower bounds m_1, m_2, m_3, and m_4^q by Δ_1, Δ_2, Δ_3, and Δ_4^q, respectively. It follows from the above definitions that $\Delta_1(b) = -(t_1(b)-1)$, $\Delta_2(b) = -(t_2(b)-1)$, $\Delta_3(b) = -(t_3(b)-1)$, and $\Delta_4^q(b) = -(t_4^q(b)-1)$. Notice that $\Delta_1(b)$, $\Delta_2(b)$, $\Delta_3(b)$, and $\Delta_4^q(b)$ are also piecewise-constant functions. If b is now increased by an amount ε for which the gradient of a lower bound remains constant, then the value of this lower bound changes by ε times the gradient.

We continue with describing how we can determine the maximum of the lower bounds efficiently, when the gradients of the lower bounds remain non-zero during the relevant increase of b. The maximum of the lower bounds m_1 and m_2 can be determined as described in Section 3.3.4. There, we show that if $t_2(M_r/(\delta+1)) > 1$, then $m_2(b) \geq m_1(b)$ holds if and only if $b \geq M_r/(\delta+1)$. If $t_2(b) = 1$ for a certain $b < M_r/(\delta+1)$, then lower bound $m_2(b)$ is constant for this and any larger values of b and given by $f(\delta+2) - M_r$.

Next, we consider lower bounds m_3 and m_4^q. Their values are attained at the same bending point of function g, if feasible, so $t_4^q(b) = t_3(b) + \tau - 1 - q$ holds for

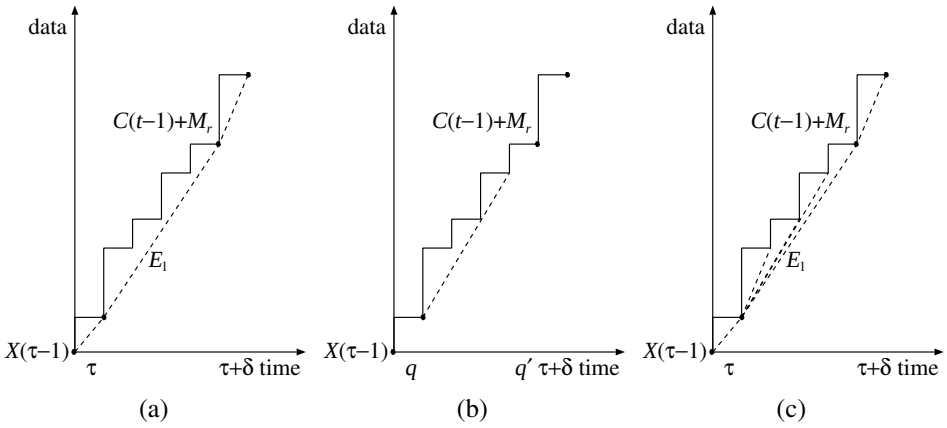

Figure 4.8. (a) Convex lower envelope E_l of $C(t-1)+M_r$, starting at $X(\tau-1)$, which is used to determine for which value of b which of $m_3(b)$ and $m_4^q(b)$ has the maximum value. (b) Convex lower envelope between q and $q'-1$. (c) All convex lower envelopes between τ and every other time unit.

$q=\tau,\ldots,\tau+\delta$. Furthermore, for $q<q'$, it follows that $t_4^q(b)+q=t_4^{q'}(b)+q'$. So, $\Delta_3(b)<\Delta_4^q(b)<\Delta_4^{q'}(b)$ holds. Thus, if for a given value of b we have $m_3(b)\leq m_4^q(b)$ for a certain q, then it also holds for larger values of b. Also, if for certain q and q' with $q<q'$ and for a given b we have $m_4^q(b)<m_4^{q'}(b)$, then the same holds for larger values of b.

For a given q and q' we have that $m_4^q(b)=m_4^{q'}(b)$ if $g(t_4^q(b)+q)-C(q-1)-M_r-(t_4^q(b)-1)b=g(t_4^{q'}(b)+q')-C(q'-1)-M_r-(t_4^{q'}(b)-1)b$ holds. With $q<q'$, we get

$$b=\frac{C(q'-1)-C(q-1)}{t_4^q(b)-t_4^{q'}(b)}=\frac{C(q'-1)-C(q-1)}{q'-q}\,. \tag{4.34}$$

So, the lower bound values of $m_4^q(b)$ and $m_4^{q'}(b)$ are equal when the bandwidth is equal to the average demand from time unit q until time unit q'. Similarly, we have $m_3(b)=m_4^q(b)$ if $g(t_3(b)+\tau-1)-X(\tau-1)-(t_3(b)-1)b=g(t_4^q(b)+q)-C(q-1)-M_r-(t_4^q(b)-1)b$ holds, which is equivalent to

$$b=\frac{C(q-1)+M_r-X(\tau-1)}{t_3(b)-t_4^q(b)}=\frac{C(q-1)+M_r-X(\tau-1)}{q-(\tau-1)}\,. \tag{4.35}$$

We now can determine the maximum of lower bounds $m_3(b)$ and $m_4^q(b)$ when b increases as follows. First, we take the upper bound on transmission $C(t-1)+M_r$ for the time units $t=\tau,\ldots,\tau+\delta$. Then, we determine the convex lower envelope of this upper bound, denoted by E_l, starting at the beginning of time unit τ at $X(\tau-1)$;

see Figure 4.8(a) for an example. The gradients of the line segments of E_1 give the values of b for which the maximum of $m_3(b)$ and $m_4^q(b)$ is attained at a different lower bound, i.e., at a different value of q or at a $m_4^q(b)$ instead of $m_3(b)$. Initially, for a small value of b the maximum of $m_3(b)$ and $m_4^q(b)$ will be given by $m_3(b)$. Now, let b' be the gradient of a line segment of E_1 and let q' be the last time unit of the line segment. If $b = b'$, then the maximum of the lower bounds $m_3(b)$ and $m_4^q(b)$ is attained at $m_4^{q'}(b)$. Note that $\Delta_4^{q'} < 0$ must hold, i.e., $t_4^{q'}(b) > 1$.

If $\Delta_4^{q'}(b)$ is equal to zero, then $m_4^{q'}(b)$ remains constant for any increase of b. Furthermore, for $q < q'$ for which $\Delta_4^q(b) < 0$, it generally does not hold that $t_4^q(b) - t_4^{q'}(b) = q' - q$ as we used in the derivations above. Instead we have $t_4^q(b) - t_4^{q'}(b) \leq q' - q$ as $t_4^{q'}(b) = 1$, which means the maximum over all q of lower bounds m_4^q may only move to q' at a higher value of b. Now suppose q' is the smallest value of q for which $t_4^q(b) - t_4^{q'}(b) < q' - q$. Then, for all time units y, with $q < y < q'$, we have $t_4^q(b) - t_4^y(b) = y - q$. The maximum of m_4^q over all q now can move to y instead of q'. To determine for which value of b we have $m_4^y(b) \geq m_4^q(b)$, we determine the convex lower envelope between time units q and $q' - 1$; see Figure 4.8(b). For all time units $y' \geq q'$, which have a constant lower bound $m_4^{y'}(b)$, we determine the maximum of all these constant lower bounds. The value of b for which $m_4^q(b)$ is then equal to this maximum, can straightforwardly be determined. Finally, instead of determining the convex lower envelope between time units q and $q' - 1$ when $\Delta_4^{q'}(b) = 0$, we can also determine in advance all convex lower envelopes between time unit τ and every other time unit up to $\tau + \delta$, when we determine E_1; see Figure 4.8(c).

We now have shown how to determine $\max\{m_1(b), m_2(b)\}$ and $\max\{m_3(b), m_4^q(b)\}$ for all b. The trade-off curve is given by the maximum of these two curves. The optimal solution is found at one of the bending points of the trade-off curve including the begin and end point. When the trade-off curve is constructed iteratively starting with a solution with minimum bandwidth, the optimal solution is obtained as soon as increasing b by an amount ε would not lead to a reduction in costs, i.e., when $c_b\varepsilon + c_s\Delta_{m_s}\varepsilon < 0$ or $\Delta_{m_s} < -\frac{c_b}{c_s}$ holds, with Δ_{m_s} the gradient of the trade-off curve.

Trade-off between b and m_r.

The trade-off between b and m_r can be performed in a similar way as the trade-off between b and m_s. For a given value of b, m_r needs to satisfy (3.29), (3.30), (4.8), (4.13), (4.31), and (4.32) with $m_s = M_s$. Constraint (4.13) just gives a constant lower bound for the receiving buffer share. Constraint (4.8) involves the known demand scheme during the first δ time units after τ and can be taken into account by using the trade-off algorithm for fully-specified streams described in Section 2.2.4.

With this algorithm a bandwidth-buffer trade-off curve can be constructed that satisfies (4.8).

For (3.29), (3.30), (4.31), and (4.32) we define analogously to the trade-off for m_s, the following lower bounds on m_r as a function of b,

$$\begin{aligned}
m_5(b) &= \max\{ f(t) - (t-1)b \mid t \geq 1\} , \\
m_6(b) &= \max\{ f(t+\delta+1) - M_s - (t-1)b \mid t \geq 1\} , \\
m_7^q(b) &= \max\{g(t-\delta+q-1) - C(q-1) - (t-1)b \mid t \geq \tau+\delta-q+1\} , \\
m_8^q(b) &= \max\{g(t+q) - C(q-1) - M_s - (t-1)b \mid t \geq 1\} ,
\end{aligned}$$

with $q = \tau, \ldots, \tau+\delta$ for $m_7^q(b)$ and $m_8^q(b)$. These lower bounds each form a piecewise-linear, convex curve in the (b, m_r)-plane. The trade-off curve for b and m_r is given by the maximum of these lower bound curves, the lower bound given by (4.13), and the trade-off curve based on (4.8).

Now we define $t_5(b)$, $t_6(b)$, $t_7^q(b)$, and $t_8^q(b)$ as the values of t where the lower bound values of $m_5(b)$, $m_6(b)$, $m_7^q(b)$, and $m_8^q(b)$ are attained, respectively. Then, for a given value of b, the gradients of $m_5(b)$, $m_6(b)$, $m_7^q(b)$, and $m_8^q(b)$ are given by $\Delta_5(b) = -(t_5(b)-1)$, $\Delta_6(b) = -(t_6(b)-1)$, $\Delta_7^q(b) = -(t_7^q(b)-1)$, and $\Delta_8^q(b) = -(t_8^q(b)-1)$, respectively. Analogously to the trade-off between b and m_s, the values of $t_5(b)$, $t_6(b)$, $t_7^q(b)$, and $t_8^q(b)$ decrease when, during the increase of b, the maximum of the corresponding lower bound equation is attained at a different bending point. When one of them is equal to its minimum value 1, the corresponding gradient equals zero and the corresponding lower bound value thus remains constant for any further increase of b.

Next, we discuss how to efficiently determine the maximum of lower bounds m_5, m_6, m_7^q, and m_8^q when their gradients remain non-zero during the relevant increase of b. First, we consider $m_5(b)$ and $m_6(b)$. As $m_5(b) = m_1(b)$ and $m_6(b) = m_2(b) + M_r - M_s$, we can derive in a similar manner that if $t_6(M_s/(\delta+1)) > 1$, then $m_5(b) \leq m_6(b)$ if and only if $b \geq M_s/(\delta+1)$ holds. Furthermore, if $t_6(b) = 1$ for a certain $b < M_s/(\delta+1)$, then lower bound $m_6(b)$ is constant for this and any larger values of b and given by $f(\delta+2) - M_s$.

Next, we consider m_7^q and m_8^q. We remark that for a given q, $t_8^q(b) = t_7^q(b) - \delta - 1$ holds, so for a given q, $\Delta_7^q(b) < \Delta_8^q(b)$. First, we determine the value of q for which $m_7^q(b)$ and $m_8^q(b)$ are maximal. Similar to lower bound $m_4^q(b)$ in the trade-off between b and m_s, we have for a given q and q' with $q < q'$, that $m_7^q(b) = m_7^{q'}(b)$ and $m_8^q(b) = m_8^{q'}(b)$ when $b = \frac{C(q'-1)-C(q-1)}{q'-q}$. Thus, the value of q for which $m_7^q(b)$ and $m_8^q(b)$ are maximal, can be determined with the convex lower envelope E_1 of $C(t)$ for the time units $\tau, \ldots, \tau+\delta$ starting at $C(\tau-1)$. Whenever b is equal to the

gradient of a line segment of E_1, the maxima over all q of $m_7^q(b)$ and $m_8^q(b)$ shift to the value of q corresponding to the time unit at the end of the line segment.

Next, we consider for which value of b we get $m_8^q(b) \geq m_7^q(b)$. As $\Delta_8^q(b) \geq \Delta_7^q(b)$ holds, the maximum of $m_7^q(b)$ and $m_8^q(b)$ is attained at $m_8^q(b)$ for any increase of b as soon as $m_8^q(b) \geq m_7^q(b)$ holds. Using the definitions we get that $m_8^q(b) \geq m_7^q(b)$ if and only if $b \geq M_s/(\delta+1)$.

The trade-off is now performed similarly to the trade-off between b and m_s. Starting with a solution with minimum bandwidth share $\tilde{b}$, b is increased iteratively until the gradient of m_r is too small for an increase of b to give a better solution.

Trade-off between b, m_s, and m_r.

For the trade-off between b, m_s, and m_r we use lower bound curves on m_s, m_r, and the total buffer size $m_s + m_r$. The lower bound curves on m_s and m_r are constructed similarly to the trade-off curves between b and m_s, and between b and m_r, described above. However, for these lower bound curves, we do not use (3.28), (3.30), (4.30), and (4.32), which correspond to lower bounds m_2, m_6, m_4^q, and m_8^q, respectively, as these constraints involve both buffers. Instead we use (3.31) and (4.33) as constraints on the total buffer size to construct a lower bound curve on $m_s + m_r$. For this, we define analogously to the above described trade-offs the lower bound functions

$$\begin{aligned} m_9(b) &= \max\{\, f(t+\delta+1) - (t-1)b \mid t \geq 1\} \,, \\ m_{10}^q(b) &= \max\{g(t+q) - C(q-1) - (t-1)b \mid t \geq 1\} \,, \end{aligned}$$

with $q = \tau, \ldots, \tau+\delta$ for $m_{10}^q(b)$. Note that lower bound $m_9(b)$ corresponds to the lower bounds $m_2(b)$ and $m_6(b)$ and that lower bound $m_{10}^q(b)$ corresponds to lower bounds $m_4^q(b)$ and $m_8^q(b)$. Furthermore, we define $t_9(b)$ and $t_{10}^q(b)$ as the values of t for which $m_9(b)$ and $m_{10}^q(b)$ attain their value, respectively. Then, the gradients of $m_9(b)$ and $m_{10}^q(b)$ are given by $\Delta_9(b) = -(t_9(b)-1)$ and $\Delta_{10}^q(b) = -(t_{10}^q(b)-1)$.

The lower bound curve on $m_s + m_r$ is given by the maximum of $m_9(b)$ and $m_{10}^q(b)$. To determine the maximum of $m_{10}^q(b)$ over all q, we use, analogously to the trade-off between b and m_r, the convex lower envelope E_1 of $C(t)$ for the time units $t = \tau, \ldots, \tau+\delta$ starting at $C(\tau-1)$. Whenever b is equal to the gradient of a line segment of E_1, the maximum of all $m_{10}^q(b)$ shifts along the line segment to the value of q matching the time unit at the end of the segment. The final lower bound curve on $m_s + m_r$ can now easily be determined by taking the maximum of $m_9(b)$ and $\max_q m_{10}^q(b)$.

The trade-off is now performed as follows. Let $m_s'(b)$, $m_r'(b)$, and $m_{s,r}'(b)$ denote the lower bound curves on m_s, m_r, and $m_s + m_r$, respectively. Furthermore, let $\Delta_{m_s'}(b)$, $\Delta_{m_r'}(b)$, and $\Delta_{m_{s,r}'}(b)$ denote their gradients. During the trade-off we

distinguish two cases, viz. $m'_s(b) + m'_r(b) \geq m'_{s,r}(b)$ and $m'_s(b) + m'_r(b) < m'_{s,r}(b)$.

If $m'_s(b) + m'_r(b) \geq m'_{s,r}(b)$ holds, the required values for the buffer shares separately are enough to meet the required value of the total buffer size. In this case, for a given b, $m_s = m'_s(b)$ and $m_r = m'_r(b)$. If now b is increased by an amount ε, the solution costs change by $(c_b + c_s\Delta_{m'_s} + c_r\Delta_{m'_r})\varepsilon$.

If $m'_s(b) + m'_r(b) < m'_{s,r}(b)$ holds, the lower bound on the total buffer size determines the total buffer allocation, as described in Section 3.3.3. If we assume that $c_s \leq c_r$, then for a given b, $m_s = \min\{m'_{s,r}(b) - m'_r(b), M_s\}$ and $m_r = m'_{s,r}(b) - m_s$. If now b is increased by an amount ε, the solution costs change by $(c_b + c_s(\Delta_{m'_{s,r}} - \Delta_{m'_r}) + c_r\Delta_{m'_r})\varepsilon$ if $m'_{s,r}(b) - m'_r(b) \leq M_s$, and by $(c_b + c_r\Delta_{m'_{s,r}})\varepsilon$ if $m'_{s,r}(b) - m'_r(b) > M_s$.

4.2.3 Transmission strategy

Finally, we consider the transmission strategy $X(t)$ for determining an efficient solution for the on-line settings of MSSP and MLBSSP. When a new stream starts, the current transmission states of the current streams impose constraints on their sending and receiving buffer shares, given by (4.1) and (4.2). These transmission states can be influenced by the transmission strategy chosen for each stream. A greedy strategy transmits data as early as possible and leads to an often largely filled receiving buffer share, while a lazy strategy transmits data as late as possible, thus leading to an often largely filled sending buffer share. Consequently, with the greedy strategy, (4.2) is likely to restrict a new solution for a current stream to a large receiving buffer share, and with the lazy strategy, (4.1) is likely to restrict a new solution to a large sending buffer share. If a new stream needs to use one of these buffers, it may find itself denied admission because of a lack of unreserved buffer space, while a feasible solution would have existed if an existing stream had used a different transmission strategy.

Other, intermediate strategies, such as the MVBA transmission schedule presented in Section 2.2.1, are also possible, as long as they satisfy the stream constraints (2.3)–(2.7) for the fully-specified streams, and (3.5)–(3.9) for the leaky-bucket-controlled streams; see Figure 4.9 for an example. Each feasible transmission strategy may lead to a different transmission state $X(\tau - 1)$ for a current stream at the start time of a new stream. The problem is which strategy to choose for a stream such that a future stream is not unnecessarily denied admission to the network. Generally, it is possible to create for each feasible transmission strategy a new stream with a start time such that it is unnecessarily denied admission. Therefore, we propose an approach involving the buffering of data that covers all feasible transmission schedules and gives maximum freedom when a new solution has to be chosen.

For a given bandwidth share b, sending buffer share m_s, and receiving buffer

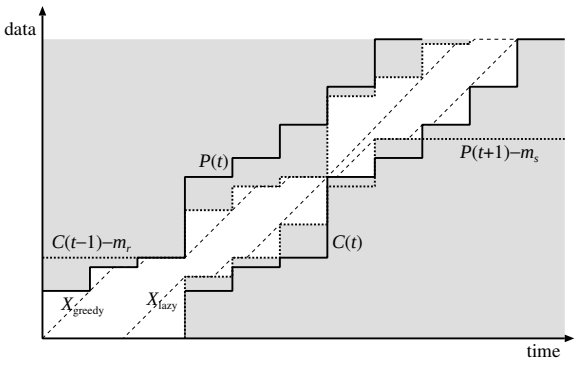

Figure 4.9. Feasible area for cumulative transmission schedules given m_s and m_r. For a given b the greedy and lazy schedule give extra bounds to the feasible area.

share m_r our approach is to transmit data according to a greedy schedule, i.e., $x(t) = \min\{b, P(t) - X(t-1), m_r + C(t-1) - X(t-1)\}$. Normally, data would be removed from the sending buffer after it has been transmitted. However, in our approach we keep data in the sending buffer until it has to be removed to create space for newly supplied data. In this way, it is available to be transmitted again later on, if it is necessary to create buffering space for a new stream at the receiving node. More formally, when a new stream starts, each current stream has a lower bound $X_{\text{low}}(\tau-1)$ and an upper bound $X_{\text{up}}(\tau-1)$ on its current transmission state $X(\tau-1)$. The lower bound is determined by the amount of data that cannot be retransmitted again as it is no longer available at the sending buffer and by the amount of data that is already consumed at the receiving node, i.e.,

$$X_{\text{low}}(\tau-1) = \max\{P(\tau) - m_s, C(\tau-1)\} \,. \tag{4.36}$$

The upper bound is determined by the greedy transmission strategy as the maximum amount of data that already has been transmitted, i.e.,

$$X_{\text{up}}(\tau-1) = X_{\text{greedy}}(\tau-1) \,, \tag{4.37}$$

with $X_{\text{greedy}}(\tau-1)$ the transmission state using the greedy transmission strategy. The transmission state $X(\tau-1)$ of a current stream can then be chosen such that it satisfies its lower and upper bound, i.e.,

$$X_{\text{low}}(\tau-1) \le X(\tau-1) \le X_{\text{up}}(\tau-1) \,. \tag{4.38}$$

Next, we show how to incorporate the freedom to choose the new transmission state $X(\tau-1)$ into the solution methods for the single-stream problems.

Fully-specified streams.

When the bandwidth share of a fully-specified stream needs to be minimized, obviously the initial transmission state $X(\tau-1)$ is maximized as then the amount of data that still needs to be transmitted, is minimized. It is also obvious that the sending buffer share can be minimized by maximizing $X(\tau-1)$, while the receiving buffer share can be minimized by minimizing $X(\tau-1)$. When both buffer shares need to be minimized simultaneously, we first minimize the buffer with the highest cost coefficient and then the buffer with the lowest cost coefficient, analogously to Algorithm 1 on page 25. The buffers are minimized according to the single buffer minimization described above where the value of the minimized 'expensive' buffer is taken into account when the 'cheap' buffer is minimized. Notice that the minimized expensive buffer may affect the upper and lower bound of $X(\tau-1)$ for the minimization of the cheap buffer.

For the trade-off between b and m_s we only need to maximize $X(\tau-1)$ as this minimizes both, after which we can apply the trade-off algorithm. We perform the trade-off between b and m_r as follows. We determine the initial solution by first minimizing m_r and thus also minimizing $X(\tau-1)$, and then minimizing b for which we maximize $X(\tau-1)$ given the obtained value of m_r. We then apply the trade-off algorithm as described in Section 2.2.4 with one difference. Instead of a fixed starting point, the transmission schedule starts at $X(\tau-1)$ which we consider to be a convex critical point during the trade-off. If $X(\tau-1) = C(\tau-2) + m_r$ and $X(\tau-1) < X_{\text{greedy}}(\tau-1)$ holds, then $X(\tau-1)$ can be increased when m_r is increased, and thus it is an m_r-critical point. When $X(\tau-1) = X_{\text{greedy}}(\tau-1)$ holds, $X(\tau-1)$ cannot be increased when m_r is increased, and thus has become fixed analogously to a $P(t)$-critical point.

Finally, the trade-off between b and both m_s and m_r is performed analogously to the method described in Section 2.2.5. Here, we use the above described procedure for minimizing both buffers given a value of b.

Leaky-bucket-controlled streams.

Similarly to fully-specified streams, we maximize $X(\tau-1)$ when b, m_s, or both need to be minimized, and we minimize $X(\tau-1)$ when m_r needs to be minimized. Note that this follows from the constraints which involve $X(\tau-1)$ and b, m_s, or m_r, viz. (4.24) and (4.25) for b, (4.29) for m_s, and (4.13) for m_r. When both m_s and m_r need to be minimized, we again first minimize the buffer with the highest cost coefficient and use this result to minimize the buffer with the lowest cost coefficient. Note that $X(\tau-1)$ for the minimization of the 'cheap' buffer may be constrained by the minimized size of the 'expensive' buffer instead of its initial lower and upper bound.

Trade-offs between b and m_s, and between b and m_r.

For the trade-offs between b and m_s or m_r we apply the methods described in Section 4.2.2 as follows. For the trade-off between b and m_s we just maximize $X(\tau-1)$ after which we perform the trade-off as if $X(\tau-1)$ were given. For the trade-off between b and m_r we determine the initial solution by first minimizing b for which we maximize $X(\tau-1)$ and then we minimize m_r for which we minimize $X(\tau-1)$ given the already obtained minimum value of b. For the trade-off we want to increase b and determine the effect on m_r as described in Section 4.2.2. However, as $X(\tau-1)$ now also is a variable, we first consider constraints that involve $X(\tau-1)$, so we can determine how to deal with $X(\tau-1)$. This concerns constraints (4.7), (4.9), (4.10), (4.13), and (4.38). Note that (4.9) and (4.10) correspond to (4.24) and (4.25), respectively.

Constraint (4.13) gives a lower bound on m_r given $X(\tau-1)$ while (4.7), (4.9), (4.10), and (4.38) give a lower bound on $X(\tau-1)$. During the trade-off b is increased and m_r needs to be minimized for the given values of b. Thus, we want to minimize $X(\tau-1)$ for each value of b. As the minimum value of $X(\tau-1)$ is equal to the maximum value of its lower bounds (4.7), (4.9), (4.10), and (4.38), we can use these lower bounds together with (4.13) to form lower bounds on m_r. This gives using (4.7) with (4.13) for $t = \tau,\ldots,\tau+\delta-1$,

$$m_r \geq C(t) - (t-\tau+1)b - C(\tau-1)\,. \tag{4.39}$$

Furthermore, using (4.9) with (4.13) we get for $t \geq 1$,

$$m_r \geq g(t+\tau-1) - (t+\delta)b - C(\tau-1)\,, \tag{4.40}$$

and using (4.10) with (4.13) we get for $t \geq 1$,

$$m_r \geq g(t+\tau-1) - M_s - (t-1)b - C(\tau-1)\,. \tag{4.41}$$

Finally, using (4.38) with (4.13) leads to

$$m_r \geq X_{\text{low}}(\tau-1) - C(\tau-1)\,. \tag{4.42}$$

These lower bounds should hold for m_r in addition to the lower bounds $m_5(b)$, $m_6(b)$, $m_7^q(b)$, $m_8^q(b)$, and the lower bound curve based on (4.8), as described in Section 4.2.2 for the trade-off between b and m_r with a given $X(\tau-1)$.

Now we consider (4.8) and lower bounds $m_7^\tau(b)$ and $m_8^\tau(b)$. It can be easily shown that m_r satisfies (4.39), (4.40), and (4.41), if it satisfies (4.8), $m_r \geq m_7^\tau(b)$, and $m_r \geq m_8^\tau(b)$, respectively. Thus, we can use the trade-off curve as described in Section 4.2.2 with only the lower bound given by (4.13) replaced by the constant lower bound given by (4.42).

Trade-off between b, m_s, and m_r.

The trade-off between b, m_s, and m_r can be handled in a similar manner as de-

scribed in Section 4.2.2 for a given $X(\tau-1)$. The initial solution is obtained by first minimizing b and thus maximizing $X(\tau-1)$ after which an optimal buffer allocation is determined for the minimum value of b. Then, we construct lower bound curves on m_s, m_r, and m_s+m_r. With these lower bound curves the trade-off can be performed as described in Section 4.2.2. Next, we show how to construct these lower bound curves.

For the lower bound curve on m_s we need to consider (3.27) and (4.29). For these constraints we have previously defined lower bounds $m_1(b)$ and $m_3(b)$, respectively. However, lower bound $m_3(b)$ depends on $X(\tau-1)$. As m_s is minimized by maximizing $X(\tau-1)$, we need to use the upper bounds on $X(\tau-1)$ for given b and m_r to replace $X(\tau-1)$ in $m_3(b)$. The upper bounds on $X(\tau-1)$ are given by (4.13) and (4.38). Together with $m_3(b)$ they give the following lower bounds on m_s.

$$\begin{aligned} m_{11}(b) &= \max\{g(t+\tau-1)-C(\tau-1)-m_r-(t-1)b \mid t \geq 1\}\,, \\ m_{12}(b) &= \max\{g(t+\tau-1)-X_{\text{up}}(\tau-1)-(t-1)b \mid t \geq 1\}\,. \end{aligned}$$

As $m_{11}(b)$ involves m_r, this lower bound will be used for the construction of the lower bound on the total buffer size m_s+m_r. The lower bound curve on m_s is thus given by the maximum of $m_1(b)$ and $m_{12}(b)$.

For the lower bound curve on m_r we need to consider (3.29), (4.8), (4.13), and (4.31). For (3.29) and (4.31) we have previously defined lower bounds $m_5(b)$ and $m_7^q(b)$ which can be used. Furthermore, for (4.8) we can construct a lower bound curve using the trade-off algorithm for fully specified streams. Constraint (4.13) involves $X(\tau-1)$ and can be transformed into four lower bounds on m_r using the lower bounds (4.7), (4.9), (4.10), and (4.38) on $X(\tau-1)$, as described for the trade-off between b and m_r. These lower bounds are given by (4.39), (4.40), (4.42), and for $t \geq 1$,

$$m_r \geq g(t+\tau-1)-m_s-(t-1)b-C(\tau-1)\,. \tag{4.43}$$

As we previously mentioned, the lower bounds given by (4.39) and (4.40) are satisfied if m_r satisfies (4.8) and $m_r \geq m_7^\tau(b)$. Furthermore, (4.43) involves both m_s and m_r and therefore will be used for the lower bound curve on m_s+m_r. The lower bound curve on m_r is thus given by the maximum of $m_5(b)$, $m_7(b)$, the curve based on (4.8), and the lower bound given by (4.42).

Finally, for the lower bound curve on m_s+m_r we need to consider (3.31) and (4.33) for which we previously defined lower bounds $m_9(b)$ and $m_{10}^q(b)$. Furthermore, for m_s we derived lower bound $m_{11}(b)$ which also involves m_r, and for m_r we derived a lower bound given by (4.43) which also involves m_s. Both lower bounds

lead to the following lower bound on $m_s + m_r$:

$$m_{13}(b) = \max\{g(t+\tau-1) - C(\tau-1) - (t-1)b \mid t \geq 1\} .$$

However, as $g(t+\tau) \geq g(t+\tau-1)$ for all $t \geq 1$, it can be easily shown that $m_{10}^{\tau}(b) \geq m_{13}(b)$. The lower bound curve on $m_s + m_r$ is thus given by the maximum of $m_9(b)$ and $m_{10}^{q}(b)$ with $q = \tau, \ldots, \tau+\delta$.

4.3 Results

In this section we present experimental results that we obtained applying the on-line variants of the algorithm as described in this chapter. We first describe the settings of the performed experiments in Section 4.3.1. Then we describe the results in Section 4.3.2.

4.3.1 Experiment setting

For the experiments with the on-line solution methods we again used the streams and settings as presented in Tables 2.1 and 2.2 on page 50 in Chapter 2. Furthermore, we used the fully-specified streams as well as the leaky-bucket-controlled streams which are described in Section 3.4. However, the streams that we used in one experiment, all are of the same type, i.e., different leaky-bucket-controlled and fully-specified streams are not mixed.

In the experiments, we varied the following settings to obtain results.

- Fixed or variable bandwidth and buffer shares. In Section 4.1.2 we described two variants of the on-line problem, viz. a variant in which bandwidth and buffer shares of current streams cannot be changed, i.e., fixed shares, and a variant in which those shares can be changed when a new stream starts, i.e., variable shares. If we use fixed shares, then the size of the shares determined for a stream is very important, as any reserved bandwidth and buffer share cannot be returned during the run time of the stream. If more bandwidth or buffer size is reserved for a stream than it actually requires, results on the number of admitted streams can be negatively affected. Especially for the LP model that only checks feasibility, i.e., the model in which the sum of the penalty variables is minimized, too large bandwidth and buffer shares may be reserved, as only a feasible solution is required. If the bandwidth and buffer shares resulting from the algorithm for this model are used, then in the worst-case situation only the first stream will be admitted; any other stream will be denied admission as then a feasible solution for the first stream may consist of the maximum bandwidth and buffer capacities.
- Use of a final adjustment procedure. To avoid bandwidth and buffer shares that are actually too large for the streams, we can make final adjustments to

the obtained shares in the following way.

1. First, we minimize the bandwidth share of the stream for the sending and receiving buffer shares obtained from the original solution.
2. Next, we minimize the sending buffer share for the bandwidth share that we just obtained and the receiving buffer share that we obtained from the original solution method.
3. Finally, we minimize the receiving buffer share for the bandwidth and sending buffer share that we obtained with these final adjustments.

We remark that we used the same procedure for the results of the off-line settings for the determination of the bandwidth and buffer utilizations. For both fixed and variable shares we have performed the experiments with and without the use of this final adjustment procedure.

- Objectives for the LP. For both fixed and variable shares we tested the effect on the results for the feasibility problem, if either the bandwidth or the maximum relative resource reservation is minimized. Note that if the total bandwidth reservation is minimized, then we do not need to make any adjustments to the bandwidth shares of the streams and thus can skip the first step in the above described final adjustment procedure.
- Transmission approach. We also used the maximum freedom transmission approach (MFT) as described in Section 4.2.3 to determine if it affects the obtained solutions. In the experiments in which we did not use MFT, we transmitted data for all streams using the greedy transmission strategy (GT), i.e., $x_d(t) = \min\{b_d, P_d(t) - X_d(t-1), C_d(t-1) + m_{r_d,d} - X_d(t-1)\}$ for each stream d. Note that for fixed shares, MFT does not have any impact and thus is not used.
- Stream inter-arrival time. In each experiment the inter-arrival time between the start of two successive streams was constant. However, we used three different values for this constant inter-arrival time, viz. for all settings an inter-arrival time equal to 1 and 1000, for settings 3, 4, 7, and 8 also an inter-arrival time equal to 2000, and for settings 1, 2, 5, and 6 also an inter-arrival time equal to 10000. The inter-arrival time of 1 corresponds to an almost simultaneous start of all streams, and thus gives a good set-up to compare to the off-line setting. The maximum inter-arrival times of 2000 and 10000 are chosen to have a relatively long time between the start of two streams while we are also assured that all streams start while any previously started stream has not finished yet.

4.3.2 Experimental results

Fully-specified streams.

The first results that we present, involve results for the fully-specified streams for all settings. Table 4.1 gives for all settings, using the final adjustment procedure as described in the previous section, and for an inter-arrival time equal to 1000, the number of admitted streams, the total bandwidth and buffer utilization as defined in Section 2.3.2, and the average and maximum run time of the solution method, when a new stream starts. The average run time is taken over all run times of the solution method to determine new bandwidth and buffer shares when a new stream starts. Note that in each experiment, the number of times new bandwidth and buffer shares are determined is equal to the number of admitted streams plus one. Only after the last time the new shares are not used as either the total bandwidth or the total buffer size at a node is exceeded by them, and the new stream is denied admission. For each setting the results are first given for fixed shares with greedy transmission, then for variable shares with greedy transmission, and finally for variable shares with maximum freedom transmission. For all these variants, the results are given for the feasibility problem (feas.), for the problem in which the total bandwidth reservation is minimized ($\min \sum b$), and for the problem in which the maximum relative resource reservation is minimized ($\min z$).

First, we consider the number of admitted streams. We see that for almost every setting, the number of admitted streams in the on-line variant with variable shares, is equal to the number of admitted streams in the off-line variant. Only for setting 1 we see that for all objectives only two streams are admitted if MFT is not used. If MFT is used, again three streams are admitted as in the off-line variant. Only for setting 7 where the bandwidth is minimized, there is a similar result. If we look at the results for the on-line variant with fixed shares, then we see some bigger differences. Only for settings 4 and 8, the number of admitted streams is equal to the number of admitted streams for the off-line variant, with the exception for setting 8 with the $\min z$-objective. For the other settings we see that fewer streams are admitted, with the best results for the $\min z$-objective for settings 3 and 7.

Next, we consider the total bandwidth and buffer utilization. We observe that only for settings 1, 2, 5, and 6 the utilization results are comparable to the utilization results of the off-line variant, especially for the on-line variant with variable shares. For the other settings we see that both the bandwidth and buffer utilizations are lower than for the off-line variant. As can be expected, the minimization of the bandwidth shares leads to a higher bandwidth utilization and consequently, a lower buffer utilization for most variants, compared to the other objectives.

Finally, we consider the run times of the different variants in Tables 4.1 and 4.2. These run times are obtained by running the experiments on an AMD XP 2600+.

		Setting 1					Setting 2				
					Run time					Run time	
Variant	*Obj.*	#	U_B	U_M	*Avg.*	*Max.*	#	U_B	U_M	*Avg.*	*Max.*
Off-line		3	62.12	44.43			4	55.79	40.95		
Fixed	*feas.*	2	61.04	31.55	0.136	0.265	2	61.15	30.58	0.099	0.140
share	$\min \sum b$	2	61.15	30.58	0.755	1.562	2	61.15	30.58	0.755	1.562
with GT	$\min z$	2	60.22	40.78	0.521	0.984	2	60.22	40.78	0.240	0.406
Variable	*feas.*	2	61.04	31.55	0.370	0.750	4	54.17	39.76	0.566	1.391
share	$\min \sum b$	2	61.15	30.58	1.135	1.672	4	60.29	40.07	1.463	2.157
with GT	$\min z$	2	60.22	40.78	1.109	1.734	4	53.21	40.33	0.809	1.062
Variable	*feas.*	3	60.34	43.40	0.824	1.656	4	55.30	39.84	0.556	1.422
share	$\min \sum b$	3	61.31	43.67	1.406	1.672	4	62.84	40.58	1.444	1.875
with MFT	$\min z$	3	60.67	44.40	1.172	1.782	4	50.94	41.55	0.747	1.047
		Setting 3					*Setting 4*				
Off-line		10	58.81	47.28			13	68.13	37.55		
Fixed	*feas.*	6	63.40	32.61	0.254	0.484	13	53.94	27.17	0.369	0.610
share	$\min \sum b$	6	61.76	33.08	0.670	1.297	13	53.96	26.99	0.592	1.328
with GT	$\min z$	8	53.20	41.37	0.417	0.656	13	53.83	28.13	0.703	1.375
Variable	*feas.*	10	54.55	39.36	1.183	3.532	13	52.98	29.93	1.241	4.297
share	$\min \sum b$	10	58.92	38.84	2.284	5.469	13	53.96	26.99	2.833	5.109
with GT	$\min z$	10	50.09	41.94	1.92	3.891	13	53.96	26.99	1.712	2.828
Variable	*feas.*	10	50.30	39.28	1.094	3.547	13	53.05	30.24	1.381	5.375
share	$\min \sum b$	10	58.93	38.68	2.543	5.750	13	53.96	26.99	1.981	3.468
with MFT	$\min z$	10	49.65	42.07	1.987	3.203	13	53.96	26.99	2.002	3.516
		Setting 5					*Setting 6*				
Off-line		3	63.80	49.87			4	59.80	52.10		
Fixed	*feas.*	2	67.31	35.23	0.182	0.313	2	67.39	33.71	0.156	0.328
share	$\min \sum b$	2	67.39	33.71	0.818	1.563	2	67.39	33.71	0.818	1.562
with GT	$\min z$	2	60.07	51.47	0.479	0.703	2	59.63	56.56	0.370	0.640
Variable	*feas.*	3	63.97	46.96	0.941	1.859	4	60.25	45.58	0.512	1.093
share	$\min \sum b$	3	63.21	49.03	1.653	2.141	4	65.83	45.33	1.734	2.406
with GT	$\min z$	3	62.59	50.39	1.086	1.422	4	62.26	47.07	1.019	1.390
Variable	*feas.*	3	63.21	46.11	0.938	2.187	4	60.44	47.20	0.500	1.015
share	$\min \sum b$	3	62.56	52.97	1.754	2.765	4	66.73	46.22	1.797	2.578
with MFT	$\min \sum z$	3	64.28	50.71	1.082	1.641	4	61.09	51.27	0.922	1.234
		Setting 7					*Setting 8*				
Off-line		12	62.71	62.46			14	74.88	42.34		
Fixed	*feas.*	6	68.32	37.15	0.275	0.407	14	57.30	30.78	0.452	0.969
share	$\min \sum b$	7	67.62	37.17	0.740	1.656	14	57.95	29.87	0.786	1.532
with GT	$\min z$	9	54.53	51.12	0.581	1.032	13	55.61	34.65	0.737	1.110
Variable	*feas.*	12	51.31	49.83	1.340	3.765	14	55.71	33.10	1.578	6.672
share	$\min \sum b$	11	59.51	47.14	2.948	6.062	14	58.11	29.92	2.407	6.016
with GT	$\min z$	12	51.27	51.79	2.343	3.813	14	57.96	30.61	3.103	5.360
Variable	*feas.*	12	52.52	49.39	1.219	3.375	14	55.68	35.88	1.531	5.266
share	$\min \sum b$	12	56.46	48.84	3.591	7.813	14	58.11	29.92	2.730	6.109
with MFT	$\min z$	12	51.62	52.75	2.464	3.891	14	57.97	30.61	3.711	6.328

Table 4.1. Results using fully-specified streams with final adjustments to the obtained shares and inter-arrival time 1000. The number of admitted streams, the bandwidth and buffer utilization, and the run time are given for all settings and for both fixed and variable shares. For variable shares results using MFT are also given.

The results show that the run times for the on-line variant with fixed shares are lower than the run times for the on-line variant with variable shares. This can be expected, with fixed shares only sub-problems for the new stream are solved while with variable shares sub-problems for all streams may have to be solved. Using MFT can increase the runtime for some settings and objectives slightly further, which is mainly noticeable for settings 3, 7, and 8 when the bandwidth is minimized. If we compare the run times between the different objectives, then we see that in general the feasibility problem has the lowest runtime, while the bandwidth minimization problem has the highest runtime.

Table 4.2 gives the same results as Table 4.1, but now without the use of the final adjustment procedure. As in our implementation without this procedure the total capacity of a resource is reserved for the first stream, if reservation of the resource is not minimized in some way, i.e., the bandwidth and buffers for the feasibility problem, and the buffers for the bandwidth minimization problem, we omit the results of these latter two problems for the on-line variant with fixed shares. The results show that the final adjustment procedure does not appear to have any effect on the number of admitted streams for variable shares. However, for fixed shares we see a lower number of admitted streams for the min z-objective for settings 3, 4, 7, and 8, the settings with the highest number of admitted streams, compared to the results with the final adjustment procedure. With regards to the utilization, we see that for settings 2, 4, and 6 the bandwidth or the buffer utilization or both, is significantly lower than in the results with the final adjustment. This can be expected, as unused parts of the bandwidth and buffer shares now are not released. Finally, the run times of the results without the final adjustment procedure seem comparable to the run times of the results with the procedure for settings 1, 2, 5, and 6, and somewhat higher for the other settings.

Leaky-bucket-controlled streams.

Table 4.3 shows the number of admitted streams for experiments with leaky-bucket-controlled streams for settings 1 and 2. Similarly, Tables 4.4, 4.5, and 4.6 show the number of admitted streams for setting 7. The number of admitted streams for settings 4, 5, 6, and 8, for the on-line variant with variable shares, are equal to the number of admitted streams in the off-line variant, for all tested objectives and inter-arrival times, with and without use of MFT, and with and without use of the final adjustment procedure. Therefore, we omit tables with the specific results for these settings. The results for setting 3 very much resemble the results for setting 7.

For a description how the different leaky-bucket-descriptions are obtained, we refer to Section 3.4.1. For settings 1 and 2 we only give the results for leaky-bucket descriptions ULB 1 and ULB 5. The results for description ULB 3 are equal to the

		Setting 1					*Setting 2*				
					Run time					*Run time*	
Variant	*Obj.*	#	U_B	U_M	*Avg.*	*Max.*	#	U_B	U_M	*Avg.*	*Max.*
Off-line		3	62.12	44.43			4	55.79	40.95		
Fixed	$\min z$	2	56.49	40.04	0.531	0.953	2	13.00	40.77	0.286	0.375
Variable	*feas.*	2	39.82	27.30	0.281	0.671	4	29.35	40.10	0.600	1.500
share	$\min \sum b$	2	61.15	24.53	0.896	1.703	4	60.29	33.26	1.222	1.844
with GT	$\min z$	2	59.69	40.71	1.005	1.437	4	17.15	41.30	0.662	0.890
Variable	*feas.*	3	59.92	38.35	0.570	1.313	4	22.26	37.67	0.422	1.110
share	$\min \sum b$	3	61.31	38.54	1.223	1.688	4	62.84	33.17	1.234	1.703
with MFT	$\min z$	3	60.67	40.16	1.043	1.453	4	30.78	37.20	1.003	1.594
		Setting 3					*Setting 4*				
Off-line		10	58.81	47.28			13	68.13	37.55		
Fixed	$\min z$	6	28.57	31.04	0.324	0.578	12	55.21	14.36	0.529	1.453
Variable	*feas.*	10	40.58	37.16	1.170	3.906	13	49.18	10.45	1.065	3.907
share	$\min \sum b$	10	58.91	37.84	2.673	5.844	13	53.96	11.62	2.338	4.078
with GT	$\min z$	10	48.52	41.83	1.892	3.328	13	53.96	18.61	1.942	3.578
Variable	*feas.*	10	41.89	37.42	1.038	3.625	13	50.00	10.81	1.372	6.562
share	$\min \sum b$	10	58.93	37.66	2.936	6.000	13	53.96	11.21	2.808	4.984
with MFT	$\min z$	10	48.52	42.04	1.933	3.125	13	53.96	15.49	2.212	4.031
		Setting 5					*Setting 6*				
Off-line		3	63.80	49.87			4	59.80	52.10		
Fixed	$\min z$	2	57.46	50.85	0.427	0.640	2	21.06	56.56	0.307	0.563
Variable	*feas.*	3	60.90	44.38	0.528	0.985	4	21.90	45.86	0.409	1.094
share	$\min \sum b$	3	63.21	46.14	1.047	1.656	4	65.83	39.11	1.375	2.219
with GT	$\min z$	3	62.51	48.70	1.043	1.406	4	16.67	44.29	1.012	1.578
Variable	*feas.*	3	62.05	42.19	0.547	1.344	4	28.94	51.85	0.622	1.859
share	$\min \sum b$	3	64.36	47.32	1.301	1.797	4	66.73	39.70	1.472	2.016
with MFT	$\min z$	3	63.11	49.75	1.039	1.406	4	25.83	50.34	1.065	1.546
		Setting 7					*Setting 8*				
Off-line		12	62.71	62.46			14	74.88	42.34		
Fixed	$\min z$	7	40.11	42.17	0.447	1.000	12	51.53	30.89	0.542	1.125
Variable	*feas.*	12	46.58	47.54	1.358	3.828	14	52.40	23.71	1.518	7.094
share	$\min \sum b$	11	59.51	45.90	3.218	6.375	14	58.11	26.62	2.945	6.609
with GT	$\min z$	12	50.66	51.77	2.081	3.344	14	57.94	29.44	2.876	4.391
Variable	*feas.*	12	46.68	46.76	1.348	4.937	14	52.66	25.93	1.603	7.656
share	$\min \sum b$	12	56.46	47.64	3.623	7.782	14	58.11	26.38	3.686	6.688
with MFT	$\min z$	12	51.38	52.68	2.397	4.281	14	57.89	29.48	3.234	5.938

Table 4.2. Results using fully-specified streams without final adjustments, for inter-arrival time 1000. Again, the number of admitted streams, the bandwidth and buffer utilization, and the run time are given for all settings and for both fixed and variable shares. For variable shares results using MFT are also given.

		Setting 1								
		Inter-arr. 1			*Inter-arr. 1000*			*Inter-arr. 10000*		
Variant	*Obj.*	ULB 1	ULB 5	F.S.	ULB 1	ULB 5	F.S.	ULB 1	ULB 5	F.S.
Off-line		2	3	3	2	3	3	2	3	3
					With final adjustment					
Fixed	*feas.*	2	2	2	2	2	2	2	2	2
share	$\min \sum b$	2	2	2	2	2	2	2	2	2
with GT	$\min z$	2	2	2	2	2	2	2	2	2
Variable	*feas.*	2	3	3	2	2	2	2	2	2
share	$\min \sum b$	2	3	3	2	2	2	2	2	2
with GT	$\min z$	2	3	3	2	2	2	2	2	2
Variable	*feas.*	2	3	3	2	3	3	2	3	3
share	$\min \sum b$	2	3	3	2	3	3	2	3	3
with MFT	$\min z$	2	3	3	2	3	3	2	3	3
					Without final adjustment					
Fixed	$\min z$	1	2	2	1	2	2	1	2	2
Variable	*feas.*	2	3	3	2	2	2	2	2	2
share	$\min \sum b$	2	3	3	2	2	2	2	2	2
with GT	$\min z$	2	3	3	2	2	2	2	2	2
Variable	*feas.*	2	3	3	2	3	3	2	3	3
share	$\min \sum b$	2	3	3	2	3	3	2	3	3
with MFT	$\min z$	2	3	3	2	3	3	2	3	3
		Setting 2								
		Inter-arr. 1			*Inter-arr. 1000*			*Inter-arr. 10000*		
Variant	*Obj.*	ULB 1	ULB 5	F.S.	ULB 1	ULB 5	F.S.	ULB 1	ULB 5	F.S.
Off-line		2	4	4	2	4	4	2	4	4
					With final adjustment					
Fixed	*feas.*	2	2	2	2	2	2	2	2	2
share	$\min \sum b$	2	2	2	2	2	2	2	2	2
with GT	$\min z$	2	2	2	2	2	2	2	2	2
Variable	*feas.*	2	4	4	2	2	4	2	2	2
share	$\min \sum b$	2	4	4	2	2	4	2	2	2
with GT	$\min z$	2	4	4	2	2	4	2	2	2
Variable	*feas.*	2	4	4	2	4	4	2	4	4
share	$\min \sum b$	2	4	4	2	4	4	2	4	4
with MFT	$\min z$	2	4	4	2	4	4	2	4	4
					Without final adjustment					
Fixed	$\min z$	2	2	2	2	2	2	2	2	2
Variable	*feas.*	2	4	4	2	2	4	2	2	2
share	$\min \sum b$	2	4	4	2	2	4	2	2	2
with GT	$\min z$	2	4	4	2	2	4	2	2	2
Variable	*feas.*	2	4	4	2	4	4	2	4	4
share	$\min \sum b$	2	4	4	2	4	4	2	4	4
with MFT	$\min z$	2	4	4	2	4	4	2	4	4

Table 4.3. Results for settings 1 and 2 concerning the number of admitted streams using leaky-bucket-controlled streams and fully-specified streams. The results are given for three different inter-arrival times, fixed and variable shares, and also using MFT with variable shares.

Setting 7		*Inter-arrival time 1*						
Variant	*Obj.*	ULB 1	ULB 3	ULB 5	CLB 5	CLB 10	Max. LB	F.S.
Off-line		11	11	12	11	12	12	12
		With final adjustment						
Fixed	*feas.*	6	6	6	6	6	6	6
share	$\min \sum b$	5	7	7	7	7	7	7
with GT	$\min z$	9	9	9	9	9	9	9
Variable	*feas.*	11	11	12	11	12	12	12
share	$\min \sum b$	11	11	12	11	12	12	12
with GT	$\min z$	11	11	12	11	12	12	12
Variable	*feas.*	11	11	12	11	12	12	12
share	$\min \sum b$	11	11	12	11	12	12	12
with MFT	$\min z$	11	11	12	11	12	12	12
		Without final adjustment						
Fixed	$\min z$	7	7	7	7	7	7	7
Variable	*feas.*	11	11	12	11	12	12	12
share	$\min \sum b$	11	11	12	11	12	12	12
with GT	$\min z$	11	11	12	11	12	12	12
Variable	*feas.*	11	11	12	11	12	12	12
share	$\min \sum b$	11	11	12	11	12	12	12
with MFT	$\min z$	11	11	12	11	12	12	12

Table 4.4. Results for setting 7 concerning the number of admitted streams using leaky-bucket-controlled streams and fully-specified streams. The results are given for inter-arrival time 1, fixed and variable shares, and also using MFT with variable shares.

Setting 7		*Inter-arrival time 1000*						
Variant	*Objective*	ULB 1	ULB 3	ULB 5	CLB 5	CLB 10	Max. LB	F.S.
		With final adjustment						
Fixed	*feas.*	6	6	6	6	6	6	6
share	$\min \sum b$	5	7	7	7	7	7	7
with GT	$\min z$	9	9	9	9	9	9	9
Variable	*feas.*	10	10	11	11	11	11	12
share	$\min \sum b$	10	11	11	11	11	11	11
with GT	$\min z$	10	11	11	11	12	12	12
Variable	*feas.*	11	11	12	11	12	12	12
share	$\min \sum b$	11	11	11	11	12	12	12
with MFT	$\min z$	11	11	12	11	12	12	12
		Without final adjustment						
Fixed	$\min z$	7	7	7	7	7	7	7
Variable	*feas.*	10	10	10	11	11	11	12
share	$\min \sum b$	10	11	11	11	11	11	11
with GT	$\min z$	10	11	11	11	12	12	12
Variable	*feas.*	11	11	11	11	12	12	12
share	$\min \sum b$	11	11	11	11	12	12	12
with MFT	$\min z$	11	11	12	11	12	12	12

Table 4.5. Results for setting 7 concerning the number of admitted streams using leaky-bucket-controlled streams and fully-specified streams. The results are given for inter-arrival time 1000, fixed and variable shares, and also using MFT with variable shares.

Setting 7		*Inter-arrival time 2000*						
Variant	*Objective*	ULB 1	ULB 3	ULB 5	CLB 5	CLB 10	Max. LB	F.S.
				With final adjustment				
Fixed	*feas.*	6	6	6	6	6	6	6
share	$\min \sum b$	5	7	7	7	7	7	7
with GT	$\min z$	9	9	9	9	9	9	9
Variable	*feas.*	10	10	10	11	10	10	12
share	$\min \sum b$	10	11	11	11	11	12	12
with GT	$\min z$	10	11	11	11	12	12	12
Variable	*feas.*	11	11	12	11	12	12	12
share	$\min \sum b$	11	11	11	11	12	12	12
with MFT	$\min z$	11	11	12	11	12	12	12
				Without final adjustment				
Fixed	$\min z$	7	7	7	7	7	7	7
Variable	*feas.*	10	10	10	10	12	11	10
share	$\min \sum b$	10	11	11	11	11	12	12
with GT	$\min z$	10	11	11	11	12	12	12
Variable	*feas.*	11	11	12	11	12	12	12
share	$\min \sum b$	11	11	11	11	12	12	12
with MFT	$\min z$	11	11	12	11	12	12	12

Table 4.6. Results for setting 7 concerning the number of admitted streams using leaky-bucket-controlled streams and fully-specified streams. The results are given for inter-arrival time 10000, fixed and variable shares, and also using MFT with variable shares.

results of ULB 1, while the results of CLB 5, CLB 10, and the maximum-leaky-bucket description are equal to the results for ULB 5. We notice that if the inter-arrival time is equal to 1, the number of admitted streams for the on-line variant with variable shares equals the number of admitted streams for the off-line variant. For the larger inter-arrival times, we see that the on-line variant with variable shares requires MFT to admit the same number of streams for ULB 5 described streams, just as for fully-specified streams. For setting 2 this means that using MFT, two more streams can be admitted, which is an increase of 100% to using the plain greedy transmission strategy.

For setting 7 we also observe that the number of admitted streams for the variant with variable shares is equal to the number of admitted streams in the off-line setting when the inter-arrival time is equal to 1. Again, for the larger inter-arrival times, the number of admitted streams can be lower and using MFT improves this number again for some objectives. For the inter-arrival time equal to 2000, we notice some remarkable results for the CLB 10 and maximum-leaky-bucket descriptions. For variable shares without MFT and without the final adjustment procedure, the results for these descriptions show a higher number of admitted streams than for fully-specified streams. An explanation for this lies in the fact that we are dealing with an on-line problem and different solutions can evolve when different decisions are made during time. For the leaky-bucket descriptions we solve the

sub-problems for the current streams in a different manner and with different input than we do for fully-specified streams. This can lead to different intermediate solutions. As the intermediate solutions determine the amount of data that can be transmitted for the current streams, this affects the transmission state of the current streams when a new stream starts, i.e., the value of $X_d(\tau - 1)$ for a current stream d. In the end, this can affect the decision whether a new stream can be admitted or not, as we see here. This characteristic of on-line problems can also explain other differences, such as the number of admitted streams for the ULB 5 description, when the on-line variant with variable shares and MFT is used. There we see twelve admitted streams for the feasibility problem and the min$z$-problem, but only eleven for the problem in which the bandwidth is minimized.

We conclude this section with the following general observations from the results of the experiments. First of all, the on-line variant with variable shares leads to better results than the variant with fixed shares, as can be expected. Furthermore, using the maximum freedom transmission approach also leads to better results than using the plain greedy transmission strategy. When we consider the used LP objectives, we see that the minimization of the maximum relative resource reservation leads to on average better results than the feasibility problem and the minimization of the bandwidth. However, compared to the feasibility problem this comes at a cost of increased run time, which is significant if fully-specified streams are used. Using the final adjustment procedure also leads to on average better results than not using the procedure, especially when fixed shares are involved. Finally, when variable shares are used together with MFT and the minimization of the maximum relative resource reservation, the number of admitted streams was in all experiments equal to the number of admitted streams in the off-line variant.

5

Conclusion

In this thesis we have considered one of a number of resource management problems situated in an IHDN to efficiently and effectively allocate the resources to the applications. These play a role when digital devices in the home are interconnected by an in-home digital network (IHDN), and new and exciting applications will become possible. More specifically, we considered the problem in which multiple video streams have to share the bandwidth of a single bus in an IHDN. Several nodes are connected to the bus, with a buffer of finite size between each node and the bus. Each stream runs from one node over the bus to another node and requires a fixed share from the bandwidth and the relevant buffer sizes to use.

We considered two types of streams for this problem, namely, fully-specified streams and leaky-bucket-controlled streams. In Chapter 2 we defined for the fully-specified streams the Multiple Streams Smoothing Problem (MSSP). It concerns the determination of bandwidth and buffer shares for a given set of fully-specified streams, such that for each stream a feasible transmission schedule exists. We modelled MSSP as a linear program (LP) and we showed how to apply the Dantzig-Wolfe decomposition for LPs to it. The Dantzig-Wolfe decomposition decomposed the original LP into a master LP and a sub-problem per stream. These sub-problems consisted of minimizing the total costs of the bandwidth and buffer shares of a single stream, where the cost coefficients for the bandwidth share, sending buffer share, and receiving buffer share, resulted from the master LP. For these

sub-problems we have described several dedicated solution methods, one for each combination of positive cost coefficients.

For the minimization of the bandwidth share of a single stream we have adapted the minimum variability bandwidth allocation method by Salehi et al. [1998]. For the minimization of a single buffer share, we have adapted the rate-constrained bandwidth smoothing method by Feng [1997]. For the sub-problem in which the cost coefficients of both buffers are positive, we have shown that an optimal solution is obtained by first minimizing the buffer with the highest cost coefficient and then the buffer with the lowest cost coefficient. When the cost coefficients of the bandwidth share and at least one buffer share are positive, a trade-off between bandwidth and buffer size is performed. For the case only one buffer is involved in this trade-off, we have described a trade-off algorithm to construct the trade-off curve by first minimizing the buffer share and then iteratively increasing the buffer share while determining the effect on the bandwidth share. This algorithm can be terminated as soon as an increase of the buffer share does not lead to a decrease in the solution costs. Finally, for the trade-off with both buffers, we have reformulated the problem into the minimization of a convex, univariate function. For this, we used the method to determine optimal buffer sizes for a given bandwidth, to obtain a function on the bandwidth that returns the total bandwidth and buffer costs. To find the minimum of this function we have described the so-called improved golden section method and triangle section method, which both use the convexity property of the function. Experimental results on several video traces have shown that on average the triangle section method requires less function evaluations than the improved golden section method.

Experimental results for the complete solution method for MSSP have shown that a solution consisting of bandwidth and buffer shares for all streams, can be determined within several seconds for instances up to 10–15 streams. The run time of the solution method for MSSP is mostly determined by the number of times a trade-off problem between bandwidth and buffer size has to be solved for a single stream, and the run time of the solution method for this trade-off problem.

In Chapter 3 we considered the leaky-bucket-controlled streams. A leaky-bucket controller controls the amount of data of a stream that can enter the network. Using the parameters of a leaky-bucket controller, we gave a linear upper bound on the supply of data for the stream during a time window. We used this upper bound for the description of the supply and demand of a stream in the definition of the Multiple Leaky-Bucket Streams Smoothing Problem (MLBSSP). In MLBSSP we need to determine bandwidth and buffer shares for a set of streams, where each stream is controlled by one or more leaky buckets, such that for each stream a feasible transmission strategy exists. Analogous to MSSP we modelled MLBSSP as an LP. We showed that by using the upper bound on the data supply as actual

data supply, MLBSSP reduces to MSSP. This allows for fully-specified and leaky-bucket-controlled streams to be combined, with the use of one master LP for the bandwidth and buffer shares of both types of streams. To solve the sub-problems per stream efficiently for leaky-bucket-controlled streams, we derived four necessary and sufficient constraints on the bandwidth and buffer shares. We showed how to solve each sub-problem using these constraints. With this solution method, each sub-problem can be solved in time linear to the number of leaky buckets that control the stream, instead of in time linearly dependent on the length of a stream as is the case for a fully-specified stream. This resulted in a run time of a fraction of a second to solve MLBSSP for instances up to 15 streams.

For experiments with the described solution method for MLBSSP, we have derived several different leaky-bucket descriptions for each fully-specified stream used in the experiments of Chapter 2. Among these descriptions is the maximum-leaky-bucket description of a stream, which we obtained from the convex hull of the empirical envelope of a stream. The results showed that using a leaky-bucket description for a stream that gives a good upper bound on data supply for small time windows, is more important for the number of streams that can be admitted, than a description that gives a good upper bound for medium to large time windows. Furthermore, using the maximum-leaky-bucket descriptions of the streams, results concerning the number of admitted streams as well as results for bandwidth and buffer minimization of a single stream were equal to corresponding results for the fully-specified streams.

In Chapter 4 we presented on-line variants of MSSP and MLBSSP in which the start time of a stream and its characteristics are unknown until it begins. We have showed that the on-line problem can be solved by solving the off-line problem at each time a new stream starts. When previously reserved shares of current streams are fixed and thus cannot be changed at that time, only the sub-problem for the new stream needs to be solved. Otherwise, if the shares are variable, the sub-problem also needs to be solved for the current streams for which we need to take into account their amount of data transmitted up to the start time of the new stream. For this, we showed how to adapt the solution methods for the sub-problems as presented in Chapters 2 and 3.

The described solution methods for the off-line variants of MSSP and MLBSSP only determined whether a feasible solution exists for a set of streams and for a given bandwidth and buffer capacities. For the on-line variants we showed how objectives such as the minimization of the total reserved bandwidth, of the total reserved buffer size at a node, and of the maximum relative resource reservation, can be incorporated into the (master) LP. Furthermore, we described an approach for the transmission of each stream, which increased the freedom of selecting new buffer shares, by sending data as early as possible but removing it from the buffers

as late as possible.

Experimental results showed that for some settings the above transmission approach increased the number of streams that can be admitted. Furthermore, allowing shares to be changed when a new stream starts also lead to better results than keeping shares fixed. The minimization of the maximum relative resource reservation generally gave better results than the minimization of the bandwidth and the feasibility problem. However, compared to the latter these better results are at the cost of an increased run time, especially for fully-specified streams. Finally, to obtain the best results we recommend that the on-line variant with variable shares is used, together with the described transmission approach and the minimization of the maximum relative resource reservation, as this combination admitted the same number of streams as the off-line setting in our experiments.

Bibliography

BOEF, E. DEN, E.H.L. AARTS, J. KORST, AND W.F.J. VERHAEGH [2004], Methods to optimally trade bandwidth against buffer usage for a VBR stream, in: W.F.J. Verhaegh, E.H.L. Aarts, and J. Korst (eds.), *Algorithms in Ambient Intelligence*, Kluwer, Chapter 13, 221–240.

BOEF, E. DEN, AND D. DEN HERTOG [2004], *Efficient Line Searching for Convex Functions*, Technical Report CentER Discussion Paper 2004-52, Tilburg University.

BOEF, E. DEN, J. KORST, AND W.F.J. VERHAEGH [2004], Optimal bus and buffer allocation for a set of leaky-bucket-controlled streams, *Proceedings 11th International Conference on Telecommunications, ICT 2004*, LNCS 3124, 1337–1346.

BOEF, E. DEN, W.F.J. VERHAEGH, AND J. KORST [2003], Smoothing streams in an in-home digital network: Optimization of bus and buffer usage, *Telecommunication Systems* **23**, 273–295.

BOEF, E. DEN, W.F.J. VERHAEGH, AND J. KORST [2004], Bus and buffer usage in in-home digital networks: Applying the Dantzig-Wolfe decomposition, *Journal of Scheduling* **7**, 119–131.

CAO, G., W. FENG, AND M. SINGHAL [2003], Online variable-bit-rate video traffic smoothing, *Computer Communications* **26**, 639–651.

CHANG, C.-S. [2000a], *Performance Guarantees in Communication Networks*, Springer-Verlag.

CHANG, R.-I [2000b], Dynamic window-based traffic smoothing for optimal delivery of online VBR media streams, *Proceedings Seventh International Conference on Parallel and Distributed Systems*, 127–134.

CHANG, R.-I, M.C. CHEN, J.-M. HO, AND M.-T. KO [1997], Designing the ON-OFF CBR transmission schedule for jitter-free VBR media playback in real-time networks, *Proceedings IEEE RTCSA*, 2–9.

CHANG, R.-I, M.C. CHEN, J.-M. HO, AND M.-T. KO [1998], Characterize the minimum required resources for admission control of pre-recorded VBR video transmission by an $O(n \log n)$ algorithm, *Proceedings IEEE 7th International Conference on Computer Communications and Networks*, 674–681.

CHANG, R.-I, M.C. CHEN, J.-M. HO, AND M.-T. KO [1999a], An effective and efficient traffic smoothing scheme for delivery of online VBR media streams, *Proceedings IEEE INFOCOM'99*, 447–454.

CHANG, R.-I, M.C. CHEN, J.-M. HO, AND M.-T. KO [1999b], Optimal bandwidth-buffer trade-off for VBR media transmission over multiple relay-servers, *Proceedings IEEE Multimedia Systems (ICMCS), vol. 2*, 31–35.

CHANG, R.-I, M.C. CHEN, J.-M. HO, AND M.-T. KO [2002], Schedulable region for VBR media transmission with optimal resource allocation and utilization, *Information Sciences* **141**, 61–79.

CHEN, W.Y. [1997], Emerging home digital networking needs, *Fourth International Workshop on Community Networking Proceedings*, 7–12.

CRUZ, R.L. [1991a], A calculus for network delay, part I: Network elements in isolation, *IEEE Transactions on Information Theory* **37**, 114–131.

CRUZ, R.L. [1991b], A calculus for network delay, part II: Network analysis, *IEEE Transactions on Information Theory* **37**, 132–141.

DALGIC, I., AND F.A. TOBAGI [1996], *Characterization of Quality and Traffic for Various Video Encoding Schemes and Various Encoder Control Schemes*, Technical Report CSL-TR-96-701, Stanford University, Departments of Electrical Engineering and Computer Science.

DANTZIG, G.B., AND P. WOLFE [1960], Decomposition principle for linear programs, *Operations Research* **8**, 101–111.

DANTZIG, G.B., AND P. WOLFE [1961], The decomposition algorithm for linear programming, *Econometrica* **29**, 767–778.

DUFFIELD, N.G., K.K. RAMAKRISHNAN, AND A.R. REIBMAN [1998], SAVE: An algorithm for smoothed adaptive video over explicit rate networks, *IEEE/ACM Transactions on Networking* **6**, 717–728.

FENG, W. [1997], Rate-constrained bandwidth smoothing for the delivery of stored video, *Proceedings SPIE Multimedia Networking and Computing*, 316–327.

FENG, W., F. JAHANIAN, AND S. SECHREST [1997], An optimal bandwidth allocation strategy for the delivery of compressed prerecorded video, *ACM Multimedia Systems Journal* **5**, 297–309.

FENG, W., AND J. REXFORD [1999], Performance evaluation of smoothing algorithms for transmitting prerecorded variable-bit-rate video, *IEEE Transactions on Multimedia* **1**, 302–313.

GILL, P.E., W. MURRAY, AND M.H. WRIGHT [1988], *Practical Optimization* (Seventh ed.), Academic Press, San Diego, California.

GILMORE, P., AND R. GOMORY [1961], A linear programming approach to the cutting stock problem, *Operations Research* **9**, 849–859.

GILMORE, P., AND R. GOMORY [1963], A linear programming approach to the

cutting stock problem: Part II, *Operations Research* **11**, 863–888.

GIOVANELLI, F., G. BIGINI, M. SOLIGHETTO, AND P. MAGGI [2003], A UPnP-based bandwidth reservation scheme for in-home digital networks, *Tenth International Conference on Telecommunications (ICT2003), vol. 2*, 1059–1064.

GONDZIO, J., R. SARKISSIAN, AND J.-P. VIAL [1997], Using an interior point method for the master problem in a decomposition approach, *European Journal of Operational Research* **101**, 577–587.

HASKELL, B.G., A. PURI, AND A.N. NETRAVALI [1997], *Digital Video: An Introduction to MPEG-2*, Chapman & Hall.

HO, J.K., AND E. LOUTE [1981], An advanced implementation of the Dantzig-Wolfe decomposition algorithm for linear programming, *Mathematical Programming* **20**, 303–326.

JIANG, Z., AND L. KLEINROCK [1998], A general optimal video smoothing algorithm, *Proceedings, IEEE INFOCOM'98*, 676–684.

KANG, S., AND H.Y. YEOM [1999], Transmission of video streams with constant bandwidth allocation, *Computer Communications* **22**, 173–180.

KANG, S., AND H.Y. YEOM [2000], Aggregated smoothing of VBR video streams, *Proceedings 14th International Conference on Information Networking (ICOIN-14)*.

KNIGHTLY, E.W., AND H. ZHANG [1995], Traffic characterization and switch utilization using deterministic bounding interval dependant traffic models, *Proceedings IEEE INFOCOM'95*, 1137–1145.

KRUNZ, M., AND S.K. TRIPATHI [1997], On the characterization of VBR MPEG streams, *Proceedings of the 1997 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 192–202.

MCMANUS, J.M., AND K.W. ROSS [1998], A dynamic programming methodology for managing prerecorded VBR sources in packet-switched networks, *Telecommunication Systems* **9**, 223–247.

MITCHELL, J.L., W.B. PENNEBAKER, C.E. FOGG, AND D.J. LEGALL [1997], *MPEG Video Compression Standard*, Chapman & Hall.

NG, J.K.-Y. [1999], A reserved bandwidth video smoothing algorithm for MPEG transmission, *The journal of systems and software* **48**, 233–245.

PAPADIMITRIOU, C.H., AND K. STEIGLITZ [1982], *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, Englewood Cliffs.

REXFORD, J., S. SEN, J. DEY, W. FENG, J. KUROSE, J. STANKOVIC, AND D. TOWSLEY [1997], Online smoothing of live, variable-bit-rate video, *Proceedings International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'97)*, 249–257.

REXFORD, J., AND D. TOWSLEY [1999], Smoothing variable-bit-rate video in an

internetwork, *IEEE/ACM Transactions on Networking* **7**, 202–215.

ROOS, C., T. TERLAKY, AND J.-P. VIAL [1997], *Theory and Algorithms for Linear Optimization: An Interior Point Approach, Chapter 19*, John Wiley & Sons, Chichester.

SALAZAR, A.E.S. [2003], Performance analysis of in-home digital networks, *IEEE Transactions on Consumer Electronics* **49**, 312–320.

SALEHI, J.D., Z.-L. ZHANG, J. KUROSE, AND D. TOWSLEY [1998], Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing, *IEEE/ACM Transactions on Networking* **6**, 397–410.

SANJAY, G., AND S.V. RAGHAVAN [1999], Fast techniques for the optimal smoothing of stored video, *Multimedia Systems* **7**, 222–233.

SCHAAR-MITREA, M. VAN DER [2001], *System and Network Constrained Scalable Video Compression*, Ph.D. thesis, Technische Universiteit Eindhoven, The Netherlands.

SCHOLTEN, H., P.G. JANSEN, F. HANSSEN, AND T. HATTINK [2002], An in-home digital network architecture for real-time and non-real-time communication, *Proceedings IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering (TENCON'02), vol. 2*, 728–731.

SGALL, J. [1998], On-line scheduling, in: A. Fiat and G.J. Woeginger (eds.), *Online Algorithms: The State of the Art*, Lecture Notes in Computer Science 1442, Springer-Verlag, 196–231.

TURNER, J.S. [1986, October], New directions in communications (or which way to the information age?), *IEEE Communications Magazine* **24**, 8–15.

WREGE, D.E., AND J. LIEBEHERR [1996], Video traffic characterization for multimedia networks with a deterministic service, *Proceedings IEEE INFOCOM'96*, 537–544.

ZHANG, J., AND J. HUI [1997], Traffic characteristics and smoothness criteria in VBR video traffic smoothing, *Proceedings IEEE International Conference on Multimedia Computing and Systems'97*, 3–11.

ZHANG, J., AND J. HUI [1998], Applying traffic smoothing techniques for quality of service control in VBR video transmissions, *Computer Communications* **21**, 375–389.

ZHANG, L., AND H. FU [2000], Dynamic bandwidth allocation and buffer dimensioning for supporting video-on-demand services in virtual private networks, *Computer Communications* **23**, 1410–1424.

ZHAO, W., T. SETH, M. KIM, AND M. WILLEBEEK-LEMAIR [1998], Optimal bandwidth/delay tradeoff for feasible-region-based scalable multimedia scheduling, *Proceedings IEEE INFOCOM'98*, 1131–1138.

Symbol Index

This index only lists notation that occur in multiple parts of this thesis. The numbers refer to the pages of first occurrence.

Problems general

$\mathcal{T}$	Set of time units.	13
$\mathcal{T}'$	Set of time points where time points $t-1$ and t mark the beginning and end of time unit t.	13
T	Time horizon.	13
$\mathcal{N}$	Set of nodes.	13
n	A node from $\mathcal{N}$.	13
B	Total available bandwidth of the bus during one time unit.	13
M_n	Total available buffer capacity at node n.	13
$\mathcal{D}$	Set of streams.	13
d	A single stream from $\mathcal{D}$.	13
s_d	Sending node of stream d.	13
r_d	Receiving node of stream d.	13
$p_d(t)$	Supply of stream d during time unit t.	13
$P_d(t)$	Cumulative supply of stream d up to time unit t.	14
$c_d(t)$	Demand of stream d during time unit t.	13
$C_d(t)$	Cumulative demand of stream d up to time unit t.	14
$x_d(t)$	Amount of data to be transmitted for stream d during time unit t.	14
$X_d(t)$	Cumulative amount of data to be transmitted for stream d up to time unit t.	14
δ	Delay between supply and demand of a stream.	14
b_d	Bandwidth share of stream d.	14
$m_{s_d,d}$	Buffer share of stream d at its sending node.	14
$m_{r_d,d}$	Buffer share of stream d at its receiving node.	14

Fully-specified streams (MSSP)

$\mathcal{D}^{\text{sup}}$	Set of streams that have a given supply scheme.	13
$\mathcal{D}^{\text{dem}}$	Set of streams that have a given demand scheme.	13
$P_d^{\max}$	Maximum supply rate of a stream d with controllable supply.	13

$C_d^{\max}$	Maximum demand rate of a stream d with controllable demand.	14

LP Model

s	Slack variable in bandwidth constraint.	16
p	Penalty variable in bandwidth constraint.	16
s_n	Slack variable in buffer constraint for node n.	16
p_n	Penalty variable in buffer constraint for node n.	16
$\lambda_{d,q}$	Weight of solution q for stream d in the master LP.	17
b_d^q	Bandwidth share of solution q for stream d.	17
$m_{n,d}^q$	Buffer share at node n of solution q for stream d.	17

Sub-problems

c_b	Cost coefficient of the bandwidth share.	18
c_s	Cost coefficient of the sending buffer share.	20
c_r	Cost coefficient of the receiving buffer share.	20
b	Bandwidth share.	20
m_s	Sending buffer share.	20
m_r	Receiving buffer share.	20
M_s	Total buffer capacity at the sending node.	20
M_r	Total buffer capacity at the receiving node.	20
$P(t)$	Cumulative supply up to time unit t.	21
$C(t)$	Cumulative demand up to time unit t.	21
$X(t)$	Cumulative amount of data transmitted up to time unit t.	21
$L(t)$	Lower bound for $X(t)$.	21
$U(t)$	Upper bound for $X(t)$.	21

Results

U_B	Total bandwidth utilization.	55
U_M	Total buffer utilization.	55
$U_B^{\min}$	Minimum bandwidth utilization as observed for a single stream in an experiment with multiple streams.	55
$U_B^{\max}$	Maximum bandwidth utilization as observed for a single stream in an experiment with multiple streams.	55
$U_M^{\min}$	Minimum buffer utilization as observed for a single stream in an experiment with multiple streams.	55
$U_M^{\max}$	Maximum buffer utilization as observed for a single stream in an experiment with multiple streams.	55

Leaky-bucket-controlled streams (MLBSSP)

$f(t)$	Upper bound on the supply of data during a time window of size t.	58
$f_d(t)$	Upper bound $f(t)$ for a stream d.	60
δ_d	Delay between the supply and demand of a stream d.	60
σ	Token bucket size of a leaky-bucket controller.	58
ρ	Arrival rate of tokens at the bucket of a leaky-bucket controller.	58
σ_i	Token bucket size of leaky-bucket controller i for a sequence of controllers.	71
ρ_i	Arrival rate of tokens at the bucket of leaky-bucket controller i for a sequence of controllers.	71

On-line setting

τ_d	Release date of stream d.	86
e_d	End time of stream d.	86
τ	Start time of a new stream.	87
z	Maximum relative resource reservation.	90
$g(t)$	Upper bound on the cumulative supply for a leaky-bucket-controlled stream up to a time unit $t \geq \tau$, based on the actual supply before τ.	94

Author Index

Samenvatting

In een digitaal huisnetwerk zijn in het huis de verschillende digitale consumenten elektronica apparaten met elkaar verbonden, zoals een set-top-box, tv-scherm of harde schijf. Dit maakt nieuwe applicaties mogelijk, zoals het kunnen bekijken van een film op elke mogelijke plek in huis op elk gewenst moment zonder dat men precies weet waar deze film is opgeslagen. Deze nieuwe applicaties leiden echter tot nieuwe 'resource management' problemen met als doel de 'resources', zoals processoren, opslagapparatuur en communicatieverbindingen, zo efficiënt en effectief mogelijk te gebruiken.

In dit proefschrift beschouwen we een enkele bus (communicatieverbinding) met beperkte bandbreedte, waarmee meerdere apparaten zijn verbonden. Tussen elk apparaat en de bus bevindt zich een buffer met beperkte capaciteit. Verder is er een verzameling video stromen gegeven waarbij elke stroom over de bus van het verzendend apparaat naar het ontvangend apparaat verzonden moet worden. Hierbij willen we voor iedere stroom een vast deel van de bandbreedte en betreffende buffers reserveren. We maken onderscheid tussen twee type stromen, te weten volledig gespecificeerde stromen en 'leaky bucket' gereguleerde stromen. Van een volledig gespecificeerde stroom weten we exact hoeveel data er wanneer wordt aangeboden en gevraagd bij de buffers van zijn verzendend respectievelijk ontvangend apparaat. Van een 'leaky bucket' gereguleerde stroom kennen we alleen de parameters van de 'leaky buckets' die de data-aanvoer van de stroom reguleren. Met deze parameters kunnen we een bovengrens voor de data-aanvoer gedurende elk mogelijk tijdsinterval geven.

Allereerst definiëren wij het Multiple Streams Smoothing Problem (MSSP). In een instantie van MSSP is een verzameling volledig gespecificeerde stromen gegeven, de bandbreedte van de bus en de groottes van de verschillende buffers. Voor elke stroom moet een vast deel van de bandbreedte en de buffergroottes worden bepaald alsmede een verzendschema waarmee alle data voor de stroom op tijd kan worden verzonden. We modelleren MSSP als een lineair programmeringsprobleem en laten zien hoe Dantzig-Wolfe decompositie hierop kan worden toegepast. Dit leidt tot een hoofdprobleem en voor iedere stroom een subprobleem. Het subprobleem voor een stroom bestaat uit het minimaliseren van de kosten van de gereserveerde bandbreedte en buffergroottes, waarbij de kostencoëfficiënten volgen uit

het geoptimaliseerde hoofdprobleem. Voor elke mogelijke combinatie van positieve kostencoëfficiënten beschrijven we voor dit subprobleem een efficiënte methode om een optimale oplossing te bepalen. Voor het minimaliseren van enkel de bandbreedte of enkel de buffergrootte van één van beide buffers passen wij hiervoor bestaande methoden aan. Voor het minimaliseren van beide buffergroottes laten we zien dat een optimale oplossing wordt verkregen door eerst de duurste buffer te minimaliseren en daarna de goedkoopste. Voor het afwegen van de bandbreedte tegen één buffergrootte beschrijven we een specifieke inruilmethode. Voor het afwegen van de bandbreedte tegen beide buffergroottes herleiden we het subprobleem eerst tot het vinden van het minimum van een stuksgewijs lineaire, convexe functie op de bandbreedte. Vervolgens beschrijven we twee efficiënte zoekmethoden om het minimum van deze functie met bijbehorende bandbreedte en buffergroottes te bepalen. Met behulp van experimentele resultaten geven we voor problemen van realistische grootte een indicatie van de rekentijd en van de benuttingsgraad van de bepaalde bandbreedte- en bufferreserveringen.

Voor de 'leaky bucket' gereguleerde stromen definiëren wij het Multiple Leaky-Bucket Streams Smoothing Problem (MLBSSP). In een instantie van MLBSSP is een verzameling 'leaky bucket' gereguleerde stromen gegeven, waarvoor een vast deel van de bandbreedte en buffergroottes moet worden bepaald alsmede verzendstrategieën waarmee alle data op tijd kan worden verstuurd. Ook MLBSSP modelleren we als een lineair programmeringsprobleem. Verder tonen we aan dat MLBSSP te reduceren is tot MSSP door de bovengrens op de data-aanvoer als daadwerkelijke data-aanvoer te gebruiken voor iedere stroom. Deze bovengrens heeft een paar specifieke kenmerken, nl. concaviteit en stuksgewijs lineariteit, die we gebruiken om voor 'leaky bucket' gereguleerde stromen de subproblemen nog efficiënter op te lossen. Hiervoor leiden we vier nieuwe, noodzakelijke en voldoende voorwaarden voor de bandbreedte- en bufferreserveringen van een stroom af. Met behulp van deze voorwaarden is de tijd om een subprobleem op te lossen lineair afhankelijk van het aantal 'leaky buckets' i.p.v. de lengte van een stroom, zoals voor volledig gespecificeerde stromen. Een oplossing kan nu binnen een fractie van een seconde bepaald worden. Om experimenten uit te voeren voor deze methode voor MLBSSP, genereren we verschillende 'leaky bucket' beschrijvingen voor iedere volledig gespecificeerde stroom die gebruikt was in de resultaten voor MSSP. De resultaten van deze experimenten zijn voor stromen die zijn beschreven door hun maximale aantal benodigde 'leaky buckets', gelijk aan de resultaten voor de volledig gespecificeerde stromen.

Behalve de bovengenoemde 'off-line' varianten van MSSP en MLBSSP beschouwen we ook 'on-line' varianten van deze problemen. In de 'on-line' varianten zijn de starttijden van stromen onbekend en zijn de kenmerken van een stroom pas bekend op het moment dat deze wil starten. Een oplossing voor een

'on-line' variant kan worden bepaald door elke keer dat een nieuwe stroom start, de methode voor het 'off-line' probleem te gebruiken om nieuwe bandbreedte- en bufferreserveringen te bepalen. Indien de reserveringen van bestaande stromen dan mogen worden aangepast, dient er bij het oplossen van de subproblemen voor deze stromen rekening gehouden te worden met de hoeveelheid data die er in totaal al verzonden is. Verdere toevoegingen aan de 'off-line' methode die we beschouwen en die kunnen leiden tot een hoger aantal toegelaten stromen, zijn doelfuncties zoals het minimaliseren van de totale gereserveerde bandbreedte of buffergrootte van een specifieke buffer. Ook laten we zien hoe de maximale relatieve 'resource' reservering geminimaliseerd kan worden. Tenslotte beschrijven we een aanpak voor de verzending van data van een stroom, waarbij data pas uit de buffers verwijderd wordt als dat nodig is om ruimte te maken voor nieuw aangeleverde data. Numerieke experimenten laten zien dat verschillende van deze aanpassingen inderdaad tot betere resultaten kunnen leiden. Het aantal toegelaten stromen in deze experimenten is voor een 'on-line' variant met bepaalde toevoegingen net zo hoog als voor de 'off-line' variant.

Curriculum Vitae

Edgar den Boef was born on June 16, 1977 in Tilburg, The Netherlands. In 1995, he obtained his 'gymnasium' (V.W.O.) diploma at the Lorentz Lyceum in Eindhoven. In the same year he started studying Econometrics at Tilburg University. In April 2000, he graduated with honors for this study, with specialization Operations Research. His Master's Thesis, for which the work was carried out at CQM in Eindhoven, described a branch-and-bound approach to the multiple container loading problem. For this thesis, he received an honorable mention from the jury of the 'VVS-scriptieprijs 2001'.

In July 2000, he started as a Ph.D. student at the Eindhoven Embedded Systems Institute of the Eindhoven University of Technology. The research, that resulted in this thesis and several papers, was performed at Philips Research Laboratories under supervision of Emile Aarts, Wim Verhaegh, and Jan Korst. As of April 2005, Edgar works as consultant on advanced planning and scheduling at Quintiq Applications B.V. in 's-Hertogenbosch.