

Master Thesis

# **Application of Network Calculus in IEEE Time-Sensitive Networks: Modeling and End-to-End Analysis**

by

Anja Hamscher

May 9, 2021

Technische Universität Kaiserslautern  
Department of Computer Science  
Distributed Computer Systems (DISCO) Lab

Supervisor: Prof. Dr.-Ing. Jens B. Schmitt  
Examiner: M. Sc. Paul Nikolaus



# Abstract

Time-Sensitive Networking (TSN) is an emerging standard, developed by the IEEE 802.1 Audio/Video Bridging Task Group and based on Ethernet, that aims to provide real-time guarantees on top of Ethernet's other functionality. Each node in a network that is implementing the TSN standard enables the prioritization, and through this, real-time guarantees in the form of guaranteed maximum delays. The prioritization is implemented into the scheduling algorithms using different traffic classes, those being either Control-Data Traffic (CDT), A, B, and Best Effort (BE), which has no priority. One of the two main scheduling algorithms is Asynchronous Traffic Shaping (ATS). It does not rely on network-wide time synchronization, which is expensive to maintain. ATS is implemented using a network element called the Interleaved Regulator (IR).

In the first part of this thesis, we look at the formal definition of this IR by going over Pi-Regularity in detail. We then analyze a proposed end-to-end delay bound for a TSN network using ATS, based on IRs, that is claimed to be tight. We find a counterexample to this claim, and analyze it. We try to utilize these findings in the next part.


In the second part, we first try to find ways of formalizing what we have learned from the counterexample. Afterwards, we define a service curve for a node using an IR based on min-plus Network Calculus (NC). We specify this service curve for an arbitrary number of nodes, using either Constant-Rate (CR) or Rate-Latency (RL) servers. Finally, we compare the delay bounds of this min-plus service curve with the initially proposed tight delay bound.



### **Eidesstattliche Erklärung**

Hiermit versichere ich, die vorliegende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet zu haben. Alle wörtlich oder sinngemäß übernommenen Zitate sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Kaiserslautern, den May 9, 2021

A handwritten signature in black ink, reading "A. Hamscher". The signature is written in a cursive style with a large initial 'A'.

---

Anja Hamscher



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Background</b>	<b>3</b>
2.1	Time-Sensitive Networking Foundations . . . . .	3
2.1.1	Node Structure . . . . .	3
2.1.2	Schedulers . . . . .	3
2.2	Network Calculus Foundations . . . . .	4
2.2.1	Min-plus Network Calculus . . . . .	5
2.2.2	Max-plus Network Calculus . . . . .	7
<b>3</b>	<b>Analyzing The Interleaved Regulator</b>	<b>11</b>
3.1	Interleaved Regulator Modeling . . . . .	11
3.1.1	Pi-Regularity . . . . .	11
3.1.2	Specification Of A CBFS-IR Network Element . . . . .	14
3.2	Delay Bound Evaluation . . . . .	17
3.2.1	Modeling Of A Counterexample . . . . .	18
3.2.2	Minimum Offsets . . . . .	21
<b>4</b>	<b>Modeling Of The Interleaved Regulator In Network Calculus</b>	<b>23</b>
4.1	Initial Approaches . . . . .	23
4.1.1	Basic Network Calculus Modeling . . . . .	23
4.1.2	Greedy Shaper . . . . .	23
4.2	Max-plus Approach . . . . .	24
4.2.1	Finding A Bound On The Backlogged Period In Max-Plus Calculus . . . . .	24
4.2.2	Usefulness For Finding A Tighter Bound . . . . .	26
4.3	Finding A Min-Plus Service Curve . . . . .	26
4.3.1	Effects Of The Interleaved Regulator . . . . .	26
4.3.2	Constructing A Min-Plus CBFS-IR Service Curve . . . . .	29
4.4	Expanding The Min-Plus CBFS-IR Service Curve To Multiple CR Servers	33
4.4.1	Convolution Of Two $\beta_{CR2,IR}$ Service Curves . . . . .	33
4.4.2	Finding A General Convolution Result For $\beta_{CR2,IR}$ Servers . . .	34
4.4.3	Convolution Of Two $\beta_{CR1,IR}$ Service Curves . . . . .	34
4.4.4	Finding A General Convolution Result For $\beta_{CR1,IR}$ Servers With Identical Rates . . . . .	37

4.4.5	Extending The General Convolution Result For $\beta_{CR1,IR}$ Servers To Different Rates . . . . .	38
4.5	Specifying The General Convolution For Rate-Latency Servers . . . . .	40
4.5.1	Adjusting The Single Server Service Curves From Constant-Rate To Rate-Latency . . . . .	41
4.5.2	Convolution Of RL Servers With Identical Rates . . . . .	43
4.5.3	Convolution Of RL Servers With Different Rates . . . . .	44
4.6	Extension To Larger Flow Aggregates . . . . .	48
<b>5</b>	<b>Analysis Of The Proposed Min-Plus CBFS-IR Service Curve</b>	<b>51</b>
5.1	Interpretation Of The Convolution Cases . . . . .	51
5.2	Delay Bounds Of The Different CBFS-IR Service Curves . . . . .	51
5.2.1	Delay Bound Of CR Servers With Identical Rates . . . . .	52
5.3	Simplification Of Analyses . . . . .	54
5.3.1	Approach 1 - RL-Like Approximation . . . . .	54
5.3.2	Approach 2 - Stair Approximation . . . . .	56
5.3.3	Delay Bounds Of CR And RL Servers Using Approximations 1 And 2 . . . . .	58
5.3.4	Miscellaneous Bounds . . . . .	60
5.4	Comparison To The Paper Bound . . . . .	60
5.4.1	Using Additive Bounds . . . . .	61
5.4.2	Using The Server Concatenation . . . . .	61
5.4.3	Comparison To The Counterexample . . . . .	63
<b>6</b>	<b>Conclusion And Future Work</b>	<b>65</b>
	<b>Acronyms</b>	<b>67</b>



# List of Figures

3.1	TSN Node Output Port Architecture . . . . .	15
3.2	Network Topology Used For Tight E2E Delay Bound . . . . .	18
3.3	Topology Of The Altered Network Used In The Counterexample . . . . .	18
3.4	Packet Arrivals At The First Node In The Counterexample . . . . .	19
3.5	Class A Traffic That Arises At The Link Between Switches 1 And 2 . . . . .	20
3.6	Class A Traffic That Arises At The Link Between Switches 2 and 3 . . . . .	21
4.1	Case Distinction For The Convolution Of A CR And IR SC . . . . .	29
4.2	Graphical Convolution Results For Cases 1 And 2 . . . . .	32
4.3	Convex And Concave Inflection Points Of A CR1 Curve . . . . .	35
4.4	First CR Case 1 Convolution With Identical Rates Using Point Clouds . . . . .	36
4.5	Second CR Case 1 Convolution With Identical Rates Using Point Clouds . . . . .	37
4.6	Third CR Case 1 Convolution With Identical Rates . . . . .	38
4.7	First RL Case 1 Convolution With Different Rates Using Point Clouds . . . . .	39
4.8	Second CR Case 1 Convolution With Different Rates Using Point Clouds . . . . .	40
4.9	First RL Case 1 Convolution With Identical Rates . . . . .	42
4.10	First And Second RL Case 1 Convolution With Different Rates . . . . .	45
4.11	Third RL Case 1 Convolution With Different Rates . . . . .	46
5.1	Delay Bound Segment Overview . . . . .	52
5.2	Approaches 1 And 2 For CR Servers With Identical Rates . . . . .	55
5.3	Approaches 1 And 2 For CR Servers With Different Rates . . . . .	56



# List of Tables

5.1	Comparison Of Additive Bounds . . . . .	61
5.2	Comparison Of Combinations With The Paper Bound . . . . .	62
5.3	Comparison Of Counterexample With The Paper Bound . . . . .	64



# 1 Introduction

Ethernet is used in a range of areas, especially in the LAN environment, as the standard for communication and data transfer between nodes. It supports heterogeneous nodes, has high bandwidth capabilities, and allows for flexible network topologies, that can easily be changed. However, Ethernet only provides Best Effort (BE), i.e., does not guarantee a bounded latency, a priority over other traffic, or that the traffic will arrive at its destination [1]. On the other hand, several areas, such as automotives and industrial automation, are experiencing changes in their design philosophies, increasing the need for technologies such as Ethernet. The big problem is that Ethernet does not provide real-time guarantees, i.e., is currently no suitable candidate for the use in these areas.

The prevalent communication systems used in automotives are CAN and FlexRay. While they offer the required real-time guarantees, their bandwidths are too low for today's needs, and they do not mix well with other communication technologies. Instead, a shift towards having a backbone in the car, e.g. using Ethernet, that connects the different communication systems and allows the communication between them is made [2]. A similar argument can be made for the industrial automation. The shift towards a flat communication system allowing to flexibly interconnect different devices, rather than an elaborate communication pyramid, are desired in the context of the Industry 4.0 movement [3].

The critical factor determining whether Ethernet can in fact be used in these areas is its real-time guarantees. As stated above, Ethernet offers BE, which inherently does not support real-time. For this reason, there has been a push towards extending Ethernet in this regard in the form of IEEE Time-Sensitive Networking (TSN), specified in IEEE 802.1. This standard aims at letting Ethernet offer a bounded latency as well as priority groups, which gain preferred service, and providing a network-wide precision clock reference [1]. With this, the usage of Ethernet in the above described areas is made possible by offering the required properties.

As with standard Ethernet, the question of the performance of TSN arises. It is important to analyze the delay and backlog for the different priority groups. Further, it is necessary to determine buffer requirements for each node to realize this new standard. To achieve this, we first need to find an accurate model for TSN in the Network Calculus (NC).

In this thesis, we first introduce TSN and NC in chapter 2. Then, in chapter 3, we introduce the Interleaved Regulator (IR), which is one of the implementation

approaches of the TSN standard, as the main focus of the thesis. After that, in chapter 4, we propose a min-plus NC approach that aims to accurately model the IR, and then analyze it in chapter 5.

## 2 Background

In this chapter, we introduce several topics that are relevant for this thesis. We start with a brief overlook over TSN, talking about the network modeling as well as different approaches regarding scheduling algorithms. Then, we introduce the min-plus and max-plus network calculus, including important results, which are used throughout the thesis.

### 2.1 Time-Sensitive Networking Foundations

The TSN standard is very complex and includes several different mechanisms. We will mainly talk about the properties that are relevant for this thesis, i.e., we will skip exploring the time synchronization aspect. It is discussed in detail by Teener et al. [1], however.

#### 2.1.1 Node Structure

Each node in a TSN network has several input and output ports. The specific scheduling algorithms that we introduce next are realized per output port. Here, we have a range of FIFO queues. Each queue is associated with a priority class, with the classes being Control-Data Traffic (CDT), A, B, and BE. CDT traffic requires hard real-time guarantees, i.e., it always has the highest priority over any other class, and thus achieves the smallest maximum delays. Classes A and B also require real-time, though it does not have to be as hard as for CDT. They still have a maximum bound on their respective delays. BE traffic, on the other hand, is normal Ethernet traffic, which does not require real-time guarantees, and is therefore sent whenever the other priority classes do not utilize the whole server bandwidth.

#### 2.1.2 Schedulers

As stated above, each TSN node output port consists of several queues. Each of these queues implements one or more scheduling algorithms. The general distinction is made between Time-Aware Scheduling (TAS) and Asynchronous Traffic Shaping (ATS). In addition, we can use the Credit-Based Shaper (CBS) on top of either of the previously mentioned algorithms.

### **Time-Aware Scheduling**

This scheduling algorithm is based on a network-wide time synchronization. Each queue is either open or closed. This is achieved through a Time-Triggered Gate (TTG), which in return is controlled by a Gate Control List (GCL). The GCL tells the TTG whether it should be open or closed. If it is open, traffic of the respective queue is allowed to pass. If it is closed, the traffic is held back. For ease of implementation, only one gate can be open at a time, to prevent further priority management.

### **Asynchronous Traffic Shaping**

In contrast to TAS, ATS does not require time synchronization. Instead, each queue shapes itself in such a way that it does not violate existing priorities with other queues, while still keeping its own priority, if it has any. This is achieved by using one IR per queue. We formally introduce this network element in chapter 3. In addition, a queue cannot send when there is traffic of a higher priority currently being sent.

### **Credit-Based Shaper**

The CBS manages the amount of traffic that a class A or B queue can send at a time. The idea is that each of these queues is associated with a credit whose initial value is 0. If the queue is currently not able to send, its credit is increased by using an idle slope  $I$  until it has reached the maximum credit score for its respective class. If the queue is able to send, its credit is decreased by a send slope  $S$  instead. Again, there is a bound on the lowest value the credit score can take on. Once the credit is negative, i.e., the queue was able to pass some traffic through the output port, it has to wait until its credit has risen enough to be at least 0. There are some specifics regarding whether CBS is used with TAS or ATS. For TAS, whenever the gate of the queue is closed, its credit score is frozen. For ATS, the credit is frozen when CDT traffic is being transmitted. CBS is discussed in detail in [4].

## **2.2 Network Calculus Foundations**

Network calculus is a framework used to model networks, using arrival and service curves. It is also used to make statements about buffer dimensioning, as well as several worst-case bounds of flows that are being sent through the modeled network. We distinguish between min-plus and max-plus network calculus. These two approaches can be transformed into the respective other approach [5].



### 2.2.1 Min-plus Network Calculus

In min-plus network calculus functions, time is the parameter that is mapped to traffic. This means that time represents the x axis, while traffic is the y axis.

#### Arrival And Departure Process

In the min-plus network calculus, data flows are represented through arrival processes, which are cumulative functions  $A(t)$ , which add any traffic of a flow arriving in an interval  $[0, t)$ .  $A$  is non-decreasing, i.e.,  $A(s) \leq A(t)$  for  $s \leq t$  and  $A(0) = 0$ , since there cannot be any traffic in the system at time 0. Each arrival process has a respective departure process  $D(t)$ . While the arrival process describes all incoming traffic of a flow, the departure process describes outgoing traffic. Between the two processes, the causality property  $A(t) \geq D(t)$  holds. This guarantees that a system does not create additional traffic, or lets traffic, which did not enter the system yet, depart.

In respect of time, arrival processes are either continuous or discrete. Discrete processes can be described by  $A(t) = \sum_{i=1}^t a_i$ , where  $a_i$  are the arrivals at time instant  $i$ . Continuous arrivals can be described by  $\int_0^t a(x)dx$ , where for all  $x$  it holds that  $a(x) \geq 0$ .

#### Arrival Curves

In reality, we do not know what arrival processes look like, but we have to approximate their shape. This is done using arrival curves. We first introduce the set of increasing functions [6]:

**Definition 2.1.** Define the set of real-valued, non-negative, increasing functions with  $f(t) = 0$  for  $t < 0$  as

$$\mathcal{F} := \{f : \mathbb{R} \rightarrow \mathbb{R}^+ \cup \{+\infty\} \mid \forall s \leq t : 0 \leq f(s) \leq f(t), \forall t < 0 : f(t) = 0\},$$

where  $\mathbb{R}^+ = [0, \infty)$ . It holds that  $f(s) \leq f(t)$ , i.e., the functions do not have to be strictly increasing. In addition, the set of functions  $\mathcal{F}$  for which  $f(0) = 0$  holds, are written as  $\mathcal{F}_0$ .

Then, an arrival curve is defined as:

**Theorem 2.2.** A function  $\alpha \in \mathcal{F}_0$  is an arrival curve for an arrival process  $A$  iff for all  $s \leq t$  it holds that

$$A(t) - A(s) \leq \alpha(t - s).$$

This means that the arrival curve  $\alpha$ , defined over any length  $t - s$ , constrains the arrivals in the interval  $[s, t]$  of the arrival process  $A$ , i.e., arrival curves are upper bounds for arrival processes over any given interval.

If we consider aggregates of several flows, we can specify an aggregate arrival curve which bounds the flow aggregate [6]:

**Theorem 2.3** (Aggregate Arrival Curve). *Let  $\alpha_1$  be an arrival curve for the arrival process  $A_1$  and  $\alpha_2$  an arrival curve for  $A_2$ . The aggregated arrivals are then bounded by  $\alpha_1 + \alpha_2$ .*

A common arrival curve used in this thesis is the Token-Bucket (TB) arrival curve, which is defined as

$$\gamma_{r,b}(t) = \begin{cases} 0 & \text{if } t \leq 0, \\ b + r \cdot t & \text{otherwise.} \end{cases} \quad (2.1)$$

## Service Curves

Much like arrival curves describe the incoming traffic of data flows, service curves are the server equivalent. They describe the available service of a network element over a given interval.

**Theorem 2.4.** *A system offers a service curve  $\beta$  to a flow with arrival process  $A$  iff it holds that*

- (1)  $\beta$  is non-decreasing,
- (2)  $D(t) \geq A \otimes \beta(t)$ .

Two service curves are used in this thesis: The Constant-Rate (CR) and Rate-Latency (RL) service curves. They are defined as

$$\beta_{R,0}(t) = R \cdot t, \quad (2.2)$$

$$\beta_{R,T}(t) = R \cdot [t - T]^+. \quad (2.3)$$

## Min-Plus (De-)Convolution

The convolution and deconvolution are essential tools for describing bounds in network calculus.

**Theorem 2.5.** *Let  $f, g \in \mathcal{F}_0$ . The min-plus convolution is defined as*

$$f \otimes g(t) := \inf_{0 \leq s \leq t} \{f(t - s) + g(s)\}. \quad (2.4)$$

*The min-plus deconvolution is defined as*

$$f \oslash g(t) := \sup_{0 \geq u} \{f(t + u) - g(u)\}. \quad (2.5)$$

## Performance Bounds

We briefly look at the different performance bounds, which are calculated using the previously defined arrival and service curves.

**Theorem 2.6** (Backlog Bound [7]). *Let  $A$  be an arrival process constrained by arrival curve  $\alpha$ , and let the network element offer a service curve  $\beta$ . The backlog bound is then*

$$q(t) \leq \alpha \oslash \beta(0) = v(\alpha, \beta), \quad (2.6)$$

where  $v(\alpha, \beta)$  is the vertical deviation of the two functions, i.e., the largest vertical distance between the functions.

**Theorem 2.7** (Delay Bound [7]). *Let  $A$  be an arrival process constrained by arrival curve  $\alpha$ , and let the network element offer a service curve  $\beta$ . The delay bound is then*

$$d(t) \leq h(\alpha, \beta), \quad (2.7)$$

where  $h(\alpha, \beta)$  is the horizontal deviation of the two functions, i.e., the largest horizontal distance between the functions.

**Theorem 2.8** (Output Bound [6]). *Let  $A$  be an arrival process constrained by arrival curve  $\alpha$ , and let the network element offer a service curve  $\beta$ . The corresponding departure process  $D$  is then constrained by the output bound*

$$\alpha'(t) := \begin{cases} \alpha \oslash \beta(t) & \text{if } t > 0, \\ 0 & \text{otherwise.} \end{cases} \quad (2.8)$$

**Theorem 2.9** (Backlogged Period Bound [6]). *Let  $\alpha$  be an arrival curve for  $A$ , which traverses a network element  $S$ , offering a strict service curve  $\beta$ . The backlogged period of the system is then upper bounded by*

$$\inf \{t > 0 \mid \alpha(t) < \beta(t)\}. \quad (2.9)$$

### 2.2.2 Max-plus Network Calculus

In contrast to min-plus calculus, functions in the max-plus calculus map traffic to time. We regard packets or bits of packets in particular.

## Arrival And Departure Time Functions

Much like arrival processes describe the arrivals of a flow in min-plus calculus, arrival time functions specify the arrival time of certain packets or bits. Here,  $T_A(\nu)$  denotes the arrival time and  $T_D(\nu)$  the departure time of the  $\nu$ -th bit in the data flow [5]. We have a causality condition in max-plus calculus as well, where it holds that  $T_D(\nu) \geq T_A(\nu)$ .

The definition of the arrival time function does not change for a discrete-space or continuous-space setting, only its meaning. With discrete-space, we take each bit  $\nu = 0, 1, \dots$  as parameter, which is mapped to a time instant. This assumes that the whole bit arrives at once. With continuous-space, we accept non-negative regular numbers as parameters, which allows us to model arrivals that do not arrive at once but are instead transmitted over time.

As a convention, negative bit values are always mapped to the time instant  $-\infty$ .

## Max-Plus Traffic Envelopes

**Theorem 2.10** ([5]).  $\lambda_E$  is a max-plus traffic envelope for an arrival time function  $T_A$  if for all  $\nu, \mu \geq 0$  it holds that

$$\lambda_E(\mu) \leq T_A(\nu + \mu) - T_A(\nu). \quad (2.10)$$

Since we swap the axes when going from min-plus to max-plus calculus, we also bound arrivals from the opposite side, i.e., arrival times in max-plus calculus are lower bounded, not upper bounded.

## Service Curves

In contrast to min-plus calculus, here we are only interested in the strict max-plus service curve [5], as this is needed to bound the backlogged period in the max-plus environment.

First, we define the busy sequence:

**Definition 2.11** (Busy Sequence). Let  $I = [\kappa, \mu)$  be an interval for which it holds that  $W(\nu) > 0$  for  $\kappa < \nu < \mu$ . In addition,  $W(\kappa^-) = W(\mu) = 0$ .  $I$  is then a busy sequence.

The start of the last busy sequence before  $\nu$  is  $\underline{\nu} = \sup \{\kappa \mid 0 \leq \kappa \leq \nu, W(\kappa) = 0\}$ , where  $W$  is the max-plus delay bound.

We can then define the strict max-plus service curve.

**Theorem 2.12.** Let  $\gamma_S$  be a non-decreasing function with  $\gamma_S(v) = -\infty$  for  $v < 0$ .  $\gamma_S$  is a strict max-plus service curve if for all  $v, \mu$  in the same busy sequence ( $\underline{v} \leq \mu < v$ ) it holds that

$$T_D(v) - T_D(\mu) \leq \gamma_S(v - \mu), \text{ if } \underline{v} < \mu, \quad (2.11)$$

$$T_D(v) - t_A(\mu) \leq \gamma_S(v - \mu), \text{ if } \underline{v} = \mu. \quad (2.12)$$

### Max-Plus (De-)Convolution

**Theorem 2.13.** Let  $F, G$  be two non-decreasing functions with  $G(v) = F(v) = -\infty$  for  $v < 0$ . The max-plus convolution is defined as

$$F \otimes G(v) := \sup_{0 \leq \kappa \leq v} \{F(\kappa) + G(v - \kappa)\}. \quad (2.13)$$

The max-plus deconvolution is defined as

$$F \oslash G(v) := \inf_{\kappa \geq 0} \{F(v + \kappa) - G(\kappa)\}. \quad (2.14)$$



## 3 Analyzing The Interleaved Regulator

### 3.1 Interleaved Regulator Modeling

As we have discussed in section 2.1, there are two approaches to implementing traffic control in the form of scheduling algorithms into TSN. We focus on ATS in this thesis. An implementation of this approach was proposed by Specht and Samii [8] and formally defined by Le Boudec [9] and its performance was analyzed by Mohammadpour et al. [10]. We discuss the findings of Mohammadpour et al. [10] in detail later in this chapter. For now, we introduce the notion of the Interleaved Regulator (IR).

The IR is a complex network element that, in its essence, is quite similar to a greedy shaper. This means that it reshapes incoming traffic back to a given curve. However, it not only reshapes incoming traffic back to a given curve, but it reshapes each flow of an incoming flow aggregate back to their respective flow constraints. This is done while preserving the FIFO order within the aggregate. To achieve this, only the packet at the head of the queue is examined as to whether it adheres to its flow constraints. If yes, it is let through immediately without delay. If not, this packet, as well as any subsequent packets of any flow in the queue, is delayed until it fulfills its flow constraint again. In the next section, we will talk about how flow constraints are modeled. This generally means that arriving traffic of a flow cannot violate its Arrival Curve (AC), which serves as an upper bound on the traffic that can arrive in a system in a specific interval.

This approach is based on the notion of interleaved shaping [8]. The main idea is that per-flow queues and per-flow states are split. We keep per-flow states, which keep track of respective flow information like burst sizes, packet lengths, and flow indices, while per-flow queues and shaping are replaced by flow aggregate queues. This lowers the number of required queues per node and thus reduces the reshaping complexity of incoming traffic. Le Boudec [9] formally defines the IR by introducing Pi-Regularity, which we discuss next.

#### 3.1.1 Pi-Regularity

The predecessor of an IR is introduced by Specht and Samii [8] as Urgency-Based Scheduler (UBS). One of the issues of formally defining this UBS is that it assumes flows to either have a Length Rate Quotient (LRQ) or Token-Bucket (TB) constraint.

While TB is an AC constraint, and thus defined in min-plus calculus, LRQ specifies the minimum time distance between two packets of the same flow, using Chang's g-regulation [11], which is defined in max-plus calculus. Thus, Pi-Regularity is designed to be able to specify max-plus and min-plus constraints, using the same notation.

We first introduce the basis of the Pi-Regularity definition. This comes in the form of a general packet sequence  $(A, L, F)$ . It is defined as follows [9]:

- $A = (A_1, A_2, \dots)$  is the sequence of packet arrivals, with  $A_i$  the arrival time of packet  $i$ , i.e., the points in time when a packet enters the server.
- $L = (L_1, L_2, \dots)$  is the sequence of packet lengths, with  $L_i$  the length of packet  $i$ . Assume that all lengths are in the same data unit, e.g. byte.
- $F = (F_1, F_2, \dots)$  is the sequence of flow numbers, with  $F_i$  the flow of packet  $i$ . Assume that there is a finite number of flows in an aggregate, but each flow has an infinite number of packets.

In case of single flows,  $F$  is left out, and the packet sequence is defined as  $(A, L)$ . Now assume  $(A, L, F)$  is the input packet sequence, while  $(D, L, F)$  is the output packet sequence. It holds that  $A \leq D$ , i.e., every packet that enters the system cannot leave it at an earlier time than its arrival time. Let  $\mathcal{F}_{gen}$  be the superset of  $\mathcal{F}$  that we defined in Definition 2.1. This means that for functions in  $\mathcal{F}_{gen}$ ,  $f(s) \leq f(t)$  for  $s \leq t$  does not necessarily hold.

With this, we can now formally define the conditions for Pi-Regularity [9]:

- C1 Let  $\Pi : \mathcal{F} \times \mathcal{G} \rightarrow \mathcal{F}_{gen}$ .  $\Pi$  takes a packet sequence  $(A, L)$  and transforms it into a sequence of time instants.
- C2  $\Pi$  is causal, i.e., it has no circular dependencies. This means that if  $\Pi(A, L) = A'$ ,  $A'_n$  can only depend on  $A_1, \dots, A_{n-1}, L_1, \dots, L_{n-1}$ , but not on  $A_m$  for  $m \geq n$  or  $L_m$  for  $m \geq n + 1$ .
- C3  $\Pi$  is homogeneous with respect to  $A$ :  $\Pi(A + h, L) = \Pi(A, L) + h$ , i.e., we can remove any constant from the sequence of packet arrivals and add it onto the result without changing any values.
- C4  $\Pi$  is isotone with respect to  $A$ : If  $A, A' \in \mathcal{F}$  with  $A \leq A'$ , then  $\Pi(A, L) \leq \Pi(A', L)$ , i.e., relationships between packet sequences are preserved.

$\Pi$  implements the flow constraints that we have mentioned before. Each flow in a flow aggregate is given an operator  $\Pi$  specifying its shape. It is defined as

$$\Pi(A, L)_n = \max_{1 \leq m \leq n-1} \{A_m + H_{m,n}(L)\}, \quad (3.1)$$

where  $A_m$  is the  $m$ th packet arrival and  $H_{m,n}(L)$  a transformation of the packet length sequence. This transformation determines the constraint the flow has. We can use



adequate functions that result in  $\Pi$  being equal to either LRQ or TB constraints. This is explained in detail in [9] and is not relevant for this thesis. It must also hold that  $A^f \geq \Pi^f(A^f, L^f)$ . This means that packet arrivals of a flow  $f$  cannot violate the flow constraint defined by  $\Pi^f$ .

Now, we can formally define the IR using Pi-Regularity.

**Theorem 3.1** (Interleaved Regulator [9]). *Let  $(A, L, F)$  be an input packet sequence where each flow in  $F$  has a regulation operator  $\Pi^f$  satisfying conditions C1-C4. The interleaved regulator is then defined as a FIFO system that transforms  $(A, L, F)$  into the output packet sequence  $(D, L, F)$  with  $D_1 = A_1$  and*

$$D_n = \max \left\{ A_n, D_{n-1}, \Pi^{F_n} \left( D^{F_n}, L^{F_n} \right)_{I(n)} \right\},$$

where  $I(n)$  is the index of packet  $n$  in its respective flow.

It is not obvious as to why this defines the IR. We first look at the third argument of the max-operator.  $\Pi^{F_n} \left( D^{F_n}, L^{F_n} \right)_{I(n)}$  returns a time instant. We also know that  $F_n$  specifies the flow that packet  $n$  belongs to, i.e., we apply the flow constraint with parameters  $D$  and  $L$  of the flow that packet  $n$  belongs to. It might be surprising to see  $D$  used here instead of  $A$ , but the meaning is identical to the inequality we have stated above. Departures of flow  $F_n$  must adhere to the flow constraints imposed by the expression above. In summary, this argument specifies the earliest time that the packet  $I(n)$  of flow  $F_n$  complies with the flow constraint  $\Pi^{F_n}$ .

We can then deduce the three situations that are represented in the definition of  $D_n$ :

- If  $D_n = A_n$  is the largest value, this means that there are no earlier packets in the queue that packet  $n$  has to wait for, and it does not violate its flows' constraints. A packet enters the IR and is immediately passed through.
- If  $D_n = D_{n-1}$  is the largest value, packet  $n - 1$  (or an earlier one that  $n - 1$  is waiting for in return) did not adhere to its flow constraints and is being held up, causing packet  $n$  to be delayed as well. Once the backlog is cleared up, the packet can leave immediately, i.e., it is adhering to its flow constraints.
- If  $\Pi^{F_n} \left( D^{F_n}, L^{F_n} \right)_{I(n)}$  is the largest value, packet  $n$  is not adhering to its flow constraints. It might still be held up, but once it is at the head of the queue, it is still delayed until its flow constraint has been restored, after which the packet is passed through.

This is exactly the description that we stated previously, meaning that Theorem 3.1 adequately defines the IR. Further, this definition yields a minimum interleaved regulator, i.e., we cannot find another IR that transforms  $(A, L, F)$  into  $(D', L, F)$  where  $D'_n \leq D_n$ . The proof can be found in [9].

We further know that the worst-case delay of the flow aggregate does not worsen when using an IR. It can and will worsen for single flows in the aggregate, but the aggregate itself does not experience this. The proof for this observation was formulated by Le Boudec [9].

### 3.1.2 Specification Of A CBFS-IR Network Element

We next look at the application of the IR definition we have seen above. Mohammadpour et al. [10] go into great detail about the modeling of a network and potential delay bounds when using the IR. We first start with the general structure of a single node in the network.

Each node in the network consists of multiple IRs and one Class-Based FIFO System (CBFS). Each CBFS has two CBS subsystems for traffic classes A and B, as well as one queue for CDT, where it is assumed that CDT traffic is bounded, and multiple queues for BE traffic. It then decides, based on priorities and credit, which traffic is allowed to be sent to the next node on the path at which time. The general structure can be seen in Figure 3.1. Strict priority means that the higher the priority of the class, the sooner the traffic will be picked for transmission. The amount of IRs at each node depends on the number of input ports for each of the two classes A and B. There is one IR for each input port, which reshapes the incoming traffic from previous nodes according to what we discussed previously. After that, the reshaped traffic is input into the respective class A / B queues at the output ports.

Despite a node being modeled as an IR-CBFS pipeline, we summarize a CBFS and the IR at the respective next node to a CBFS-IR element. The reasoning for this is that we follow a class aggregate<sup>1</sup> along a certain path through the network and the output of the CBFS element of one node directly corresponds to the respective IR of the next node. If we would not do this, but instead looked at the IR of the same node as the CBFS, we would have no guarantee that all crossflows will stay in the same class aggregate as the flow of interest (foi) after the aggregate leaves the CBFS. We see later that the maximum delay in the CBFS directly correlates with the maximum delay in the next IR, i.e., this view on the nodes makes it easier to understand and model the approach.

We next look at the available service the CBFS can offer to the class aggregate. Assume we have a class A aggregate traversing the network, i.e., CDT traffic has priority over the aggregate, while it has priority over the remaining classes B and BE. As a consequence, the class aggregate does not receive the full server capacity to transmit traffic. Instead, we calculate a RL Left-Over (LO)Service Curve (SC), i.e., the SC that specifies the leftover service the server can offer to class A after serving the existing CDT traffic. It is specified as follows:

<sup>1</sup>Note that we use flow aggregate and class aggregate interchangeably. A class aggregate is a flow aggregate where all flows belong to the same class.

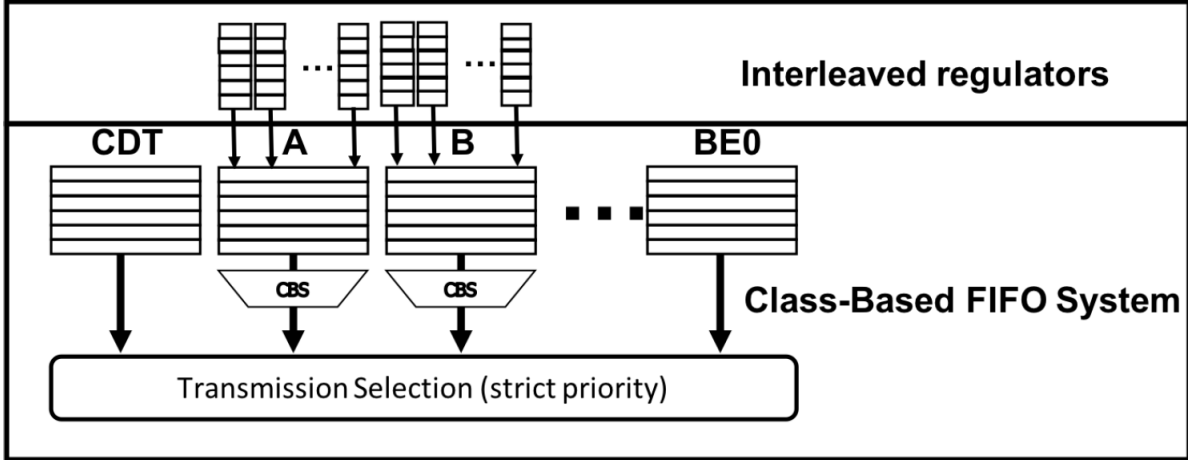


Figure 3.1: Architecture of a TSN node output port [10].

**Theorem 3.2** (LOSC for class A flows [10]). *Let  $i$  be a node in the network that has an existing link to node  $j$  ( $i, j$ ). Let the CDT traffic, which is sent over the link ( $i, j$ ), be bounded by a TB AC with parameters  $b_{ij}$  and  $r_{ij}$ . Let  $I_{ij}^A, S_{ij}^A$  be the idle and send slope of class A on the link ( $i, j$ ),  $\bar{L}_{ij}^A = \max(L_{ij}^B, L_{ij}^E)$  the maximum packet length of class B or BE traffic, and  $\bar{L}_{ij} = \max(L_{ij}^A, L_{ij}^B, L_{ij}^E)$  the maximum packet length of non-CDT traffic. The CBFS then offers a RL SC to class A with parameters*

$$T_{ij}^A = \frac{1}{c_{ij} - r_{ij}} \cdot \left( \bar{L}_{ij}^A + b_{ij} + \frac{r_{ij} \cdot \bar{L}_{ij}}{c_{ij}} \right), \quad (3.2)$$

$$R_{ij}^A = \frac{I_{ij}^A \cdot (c_{ij} - r_{ij})}{I_{ij}^A - S_{ij}^A}. \quad (3.3)$$

We specify the single-node delay bound next. It is important to note here that we look at the delay bound of only the CBFS, not the CBFS-IR element. This is an essential detail. A packet that experiences its worst-case delay in the CBFS does not experience any delay in the following IR. This might seem counter-intuitive to what we have discussed before, but if a packet has been delayed at the CBFS, it adheres to its flow constraints, and therefore there are no earlier packets at the IR that might delay the packet any further. If we recall the formal definition of the IR, this becomes a bit clearer. The packets that experience delays at the IR are the packets that follow after the one that experienced its maximum delay at the CBFS, as they pass the CBFS too fast for them to adhere to the flow constraint, due to the arising burstiness increase. This is caused by the following packets experiencing a shift of delay from the CBFS to the IR, until the flow has regained its shape. It is interesting to note that the maximum delay any packet can experience in the IR is equal to the maximum delay it can experience in the previous CBFS. The reason is very simple. A delay in the CBFS

causes a right-shift by the experienced burstiness increase of the flow. The IR then delays packets until this right-shift has been negated. This is the basis of our proposed CBFS-IR node modeling approach, and is discussed in great detail in chapter 4. For now, we show the delay for a single node proposed by Mohammadpour et al. [10]:

**Theorem 3.3.** *Let  $f$  be a flow of class  $x \in \{A, B\}$ ,  $(i, j)$  a link going from node  $i$  to  $j$  and  $b_{ij}^{tot,x} = \sum_{f \in F_{ij}^x} b_f$ . An upper bound on the response time of node  $i$  is given by*

$$S(f, i, j, x) = T_{ij}^x + \frac{b_{ij}^{tot,x} - \psi_f}{R_{ij}^x} + \frac{\psi_f}{c_{ij}} + T_{ij}^{var,max}, \quad (3.4)$$

where  $\psi_f$  is the maximum packet length  $L_f$  for LRQ constraints, and  $M_f$  for TB constraints.

In its essence, this delay bound looks very similar to the delay bound of a TB AC and RL SC, which is given by  $T + \frac{b}{R}$ . In addition, the maximum transmission time of a packet is given by  $\frac{\psi_f}{c_{ij}}$ , while the maximum packet length is subtracted from the sum of all bursts of the flow aggregate. It is claimed in [10] that the per-node delay bound stated above is tighter than this bound.

The transition to the end-to-end (e2e) delay bound is made by summarizing the maximum delay of any flow in the class  $x$  flow aggregate per node, where the only flows that are considered are those that stay in the aggregate of the foi until node  $k$ , which follows after  $j$ . The reason for this is quite simple. If a flow is only joined with the foi for one server, it is not in the same IR as the foi in the next node, as it is instead sent to a different input port. This maximum delay per node is specified by

$$C(i, j, k, x) = T_{ij}^x + \frac{b_{ij}^{tot,x}}{R_{ij}^x} + T_{ij}^{var,max} + \max_{f' \in F_{ijk}^x} \left( \frac{\psi_{f'}}{c_{ij}} - \frac{\psi_{f'}}{R_{ij}^x} \right) + T_{ij}^{proc,max}, \quad (3.5)$$

where  $\psi_f$  is the maximum packet length  $L_f$  for LRQ constraints, and  $M_f$  for TB constraints. In the remaining thesis, we assume that  $T_{ij}^{var,max}$  and  $T_{ij}^{proc,max}$  are 0. This assumption is also made for the examples in [10]. We briefly discuss why the maximum delay per node is used instead of the delay of the foi. What the single-node delay bound does is subtracting the maximum packet length of the respective flow from the total burst in the aggregate. If all flows have identical maximum packet lengths, their delay bounds are the same. If this is not the case, one delay bound ends up being the largest one. We then assume that the packets of the foi are always the last to be sent, i.e., the largest delay of a packet of the foi is exactly the longest delay of any of the flows.

Assume a flow  $f$  traverses nodes  $(i_1, \dots, i_k)$ , where  $i_1$  is the source and  $i_k$  is the destination. We can assume that the IR at the source node does not reshape any flows, as

they conform to their respective constraints when they are being sent. The e2e delay bound for the foi is then specified as

$$D_f^x = \sum_{j=1}^{k-2} C(i_j, i_{j+1}, i_{j+2}, x) + S(f, i_{k-1}, i_k, x). \quad (3.6)$$

For the remainder of the thesis, this bound will be referenced as paper bound. This concludes the specification of the network nodes. Next, we evaluate the above stated delay bound regarding its tightness, as Mohammadpour et al. [10] claim it is tight.

## 3.2 Delay Bound Evaluation

Suspensions about the tightness of the above stated bound arose quickly. While the paper bound might be tighter than the regular TB-RL delay bound for a single node, it is still an additive bound for  $n$  servers. We add  $\frac{b}{R}$  per server for the paper bound, while the TB-RL bound only adds this once for the whole path that is traversed by the foi. We checked the example given in [10] that is used to prove the tightness of the paper bound. Its network is shown in Figure 3.2. We have the foi  $f_1$ , source  $H_1$ , destination  $H_6$ , and 4 switches that the foi traverses. At each switch, it has one crossflow that joins the class aggregate, and one that leaves it. We could not find a counterexample for this specific network topology. The reason for that is quite simple. This specific structure of flows joining and leaving the aggregate causes the foi to burst at every switch it passes. Because of the additivity of the bound, it will model the delay in a way that the foi bursts at every server. This is, however, not true for arbitrary networks. Due to the "Pay Bursts Only Once (PBOO)" property [7], we know that the fois burst is only supposed to impact the delay bound once per path, not once per server, as is the case with the paper bound. This has the consequence that the additive bound that is claimed to be tight cannot be tight for any network that is not modeled like the one in Figure 3.2.

After making this observation, we decided to build our own example network topology, which would prove the hypothesis of the bound not being tight. This was very tricky, as we had to figure out timings of packets based on the TSN standard and the sparse information about the example that was given in [10]. We tried to keep the modeling of the network as similar to the example in [10] as possible, where the only change is the reduction from five crossflows to one. We also keep both the foi and crossflow in the same aggregate from  $H_1$  to  $H_6$ , i.e., no additional flow joins or leaves the class aggregate on the path. This network topology is depicted in Figure 3.3. We modeled the example per node, including the timing of each packet, much like it was done in [10]. This results in an interesting find, which we discuss after illustrating the counterexample.

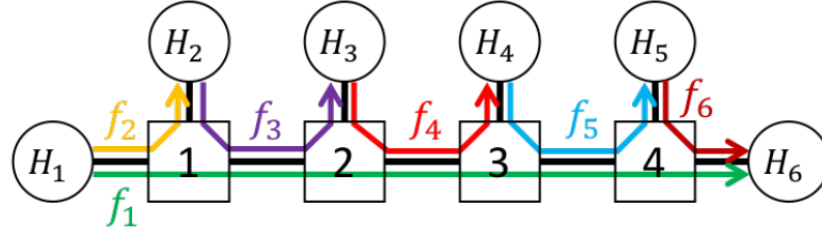


Figure 3.2: Network, including flows, that is used to prove the tightness of the e2e delay bound [10].

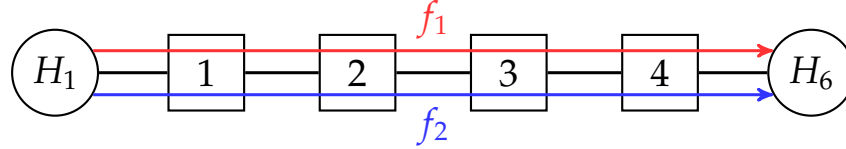


Figure 3.3: Topology of the altered network used in the counterexample.

### 3.2.1 Modeling Of A Counterexample

As mentioned above, we keep all parameters identical to the paper example so we can reasonably compare our results later.

- $f_1$  (foi),  $f_2$  (crossflow) are class A flows which are LRQ regulated with the rates  $r_{f_i} = 20$  Mbit/s and maximum packet lengths  $L_{f_1} = 1$  Kbit,  $L_{f_2} = 2$  Kbit. Assume all sent packets have the respective maximum length specified by their constraint.
- Each node has CDT traffic with the parameters (20 Mbit/s, 4 Kbit), as well as BE traffic with a maximum packet length of 2 Kbit.
- The line rate of any link  $(i, j)$  is  $c_{ij} = 100$  Mbit/s.
- The CBS at each node is modeled with the parameters  $I_{ij}^A = 50$  Mbit/s and  $S_{ij}^A = -50$  Mbit/s.
- The LOSC at each server is then specified with the parameters  $T_{ij}^A = 80\mu s$  and  $R_{ij}^A = 40$  Mbit/s.

We keep the traffic at the first CBFS-IR element identical (compare Figure 3.4), and then take the output of the first switch as input of the second one. We do this for all switches, until we arrive at the destination node  $H_6$ . We experimented with forcing the maximum delay for the foi at each node. Depending on when the respective CDT and BE traffic at the CBFS is sent, we achieve the worst case. The CDT and BE traffic always have to be sent in a specific relation to the crossflow packets, where the first CDT traffic is sent exactly  $20\mu s$  before a packet of  $f_2$  arrives at the node. This is due

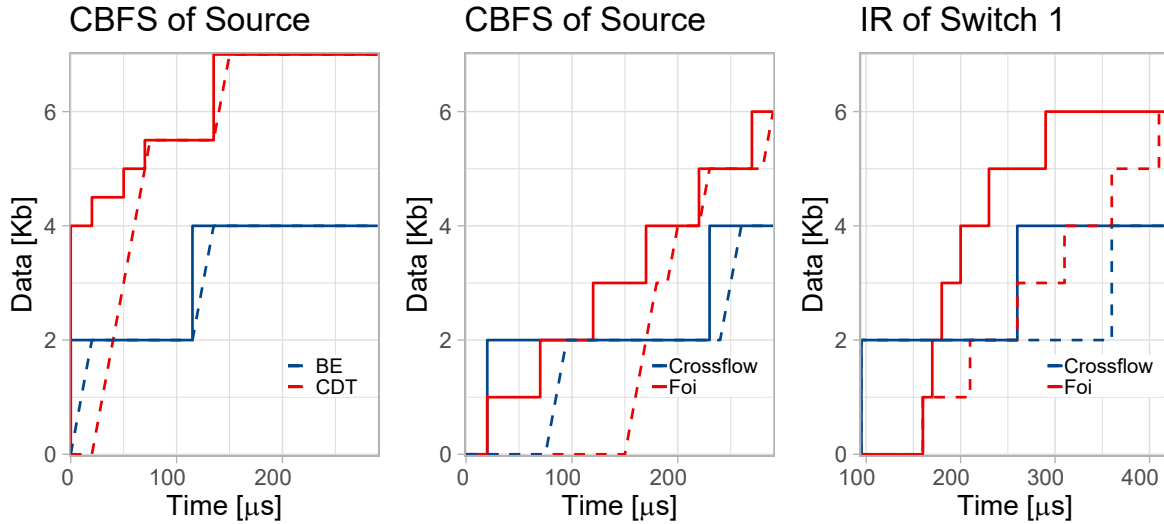


Figure 3.4: Overview of the timings used in the examples in [10]. We assume the same timings at  $H_1$  for our example. Dashed lines are the respective outputs of the different flows, while solid lines are the input.

to the CBS, which is used in addition to ATS. While there is CDT traffic at the node, the credit score of any class A or B flow is frozen, i.e., it cannot increase or decrease. The server first sends the BE traffic, then schedules the CDT traffic exactly when the first crossflow packet arrives (at  $20\mu s$  in the first plot of Figure 3.4). By sending CDT traffic right when crossflow traffic arrives, we effectively prevent the crossflow from building up additional credit score, which leaves it at 0. The general order of traffic at each server is hence BE traffic, followed by CDT and crossflow traffic, where the CDT traffic is passed through first.  $f_2$  can then send one single packet at  $75\mu s$ . As its credit score is still at 0, once it has sent this packet its score is reduced to  $-1$  Kbit, and it needs to wait until its score is at least 0 before it can send another packet. This takes  $20\mu s$  with the specified idle slope value. We schedule more BE and CDT traffic during this period of time, further preventing the crossflow from regaining its credit score. The "trick" to maximizing the delay of the foi is to maximize the time the crossflow cannot build up its credit score, hence delaying its packets for as long as possible. Once all other traffic is served,  $f_1$  can start sending at  $150\mu s$ , exactly  $130\mu s$  after its first packet entered the server. It then leaves the CBFS  $10\mu s$  later, resulting in a total delay of  $140\mu s$ . This timeframe equals the calculated worst case delay of  $f_1$  at  $H_1$ .

We have already mentioned the PBOO property, where the burst of a flow impacts its delay bound only once. We can see this property in effect in our example. At  $H_1$ , both flows are being sent at the same time, i.e., they arrive at the CBFS of  $H_1$  at the same time. The delay in the CBFS then causes the maximum burstiness increase for the foi. We can see this in the CBFS as well as the following IR in Figure 3.4. The first packet of  $f_1$  is delayed by the maximum time of  $140\mu s$ , causing a burstiness increase of the following packets, which results in a deformation of the flow shape in the IR

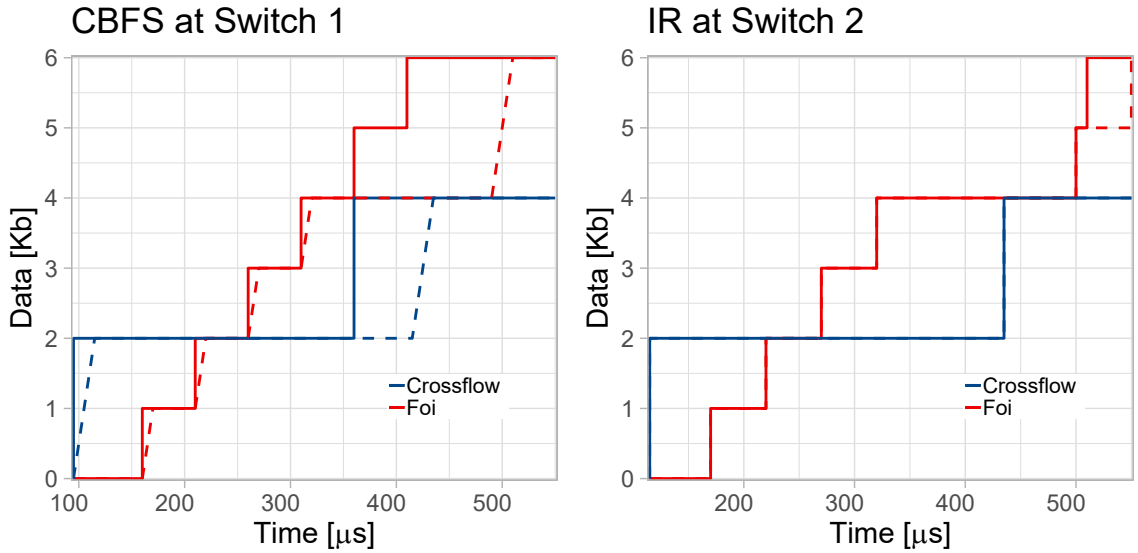


Figure 3.5: Class A traffic that arises at the link between switches 1 and 2.

of switch 1. This deformation is reversed by the IR, where it delays any packet of  $f_1$  until it adheres to its flow constraint again. What we can observe in the right plot is something similar to a desynchronization between flows  $f_1$  and  $f_2$ . Because the foi is delayed by the maximum burstiness increase, the first packet of  $f_1$  does not arrive at the CBFS of switch 1 at the same time as the packet of  $f_2$ , as was the case at  $H_1$ . The IR effectively changes the spacing between packets of the flows in the class aggregate. We can see this in Figure 3.4 on the right. While at  $H_1$ , both first packets of  $f_1$  and  $f_2$  arrive at  $20\mu s$ . These two packets have a distance of  $65\mu s$  when they exit the IR of switch 1. Thinking about what consequences this has, the first thing that comes to mind is that two packets that collided at a previous node cannot arrive at a following node at the same time. In return, this potentially reduces the worst case delay the foi can experience at subsequent nodes, as the further away it is from a packet of the crossflow  $f_2$ , the less it is possibly impacted by a delay of the crossflow.

We can see this in our example as well. In Figure 3.5 on the left, we can see that the second last packet of  $f_1$  and the last packet of  $f_2$  arrive at the CBFS of switch 1 at the same time. This means that  $f_1$  will experience its worst-case burstiness increase at a different time than at  $H_1$ . The IR of switch 2 then changes the spacing of these two packets. Afterwards, there are no packets of the two flows left that could arrive at the same time, as can be seen in Figure 3.6, i.e., we cannot reach the worst-case burstiness increase any more. This results in a lower delay of  $85\mu s$  for the foi at any subsequent node.



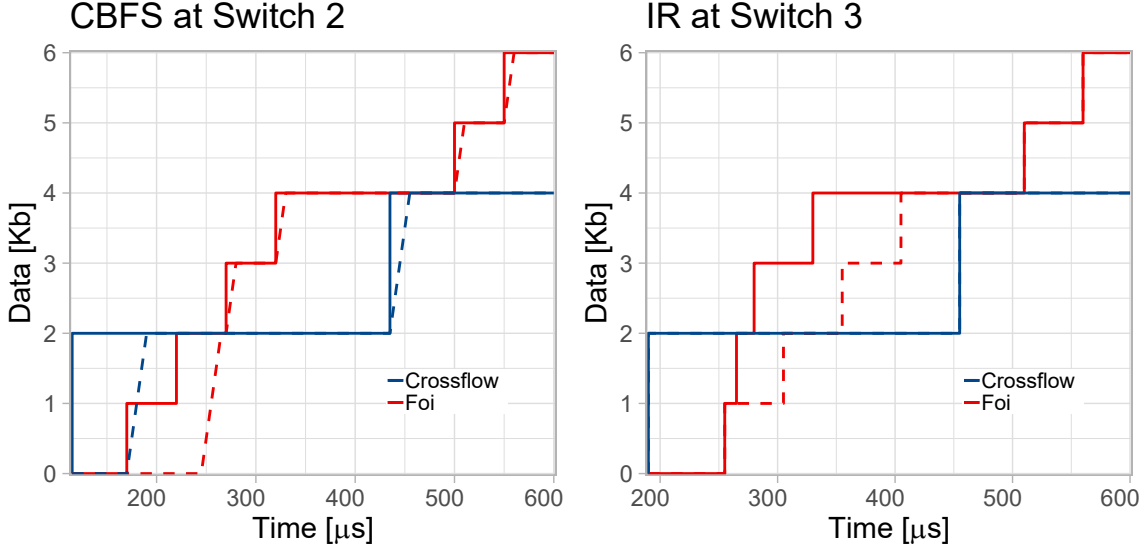


Figure 3.6: Class A traffic that arises at the link between switches 2 and 3, showing the consistency of the minimum packet distance between the aggregate flows.

### 3.2.2 Minimum Offsets

After the initial findings in our example, we experimented further with the desynchronization. We can assume that once a packet of the foi arrived at the same time as a packet of the crossflow, it cannot reach the worst case at another switch any more. The compressed arrivals of the packets of the foi following this worst case packet will also be pulled apart at the following IR, as described in the previous section. The IR does this to negate the right-shift caused by the delay at the previous CBFS. An additional observation is that once the spacing between the packets of  $f_1$  and  $f_2$  is big enough, they do not have an effect on the delay of the respective other flow any more. Conversely, the smaller the spacing between two packets of the flows, the more they impact each other, with a distance of 0 causing the largest impact. Once the desynchronization caused by the IR is completed, packets of both flows will maintain a minimum offset between any two consecutive packets, where the first packet belongs to  $f_2$  and the second to  $f_1$ . This can be seen in Figure 3.6. The first packets of  $f_1$  and  $f_2$  have a minimum distance of  $55\mu s$  when they enter the CBFS at switch 2. When they leave the IR at switch 3, their minimum distance has changed to  $65\mu s$ . Instead, the second packet of  $f_2$  and the fifth packet of  $f_1$  have the minimum distance of  $55\mu s$ , i.e., the minimum distance is not always measured between the same two packets, but its value stays the same. We can measure the impact of the minimum offset on the per-node delay bound:

$$S(f, i, j, x) = \left[ T_{ij}^x + \frac{b_{ij}^{tot,x} - \psi_f}{R_{ij}^x} - \min \left\{ A_{p_{f_1}} - A_{p_{f_2}} \right\} \right]^+ + \frac{\psi_f}{c_{ij}}, \quad (3.7)$$

where  $A_{p_{f_1}}, A_{p_{f_2}}$  are the arrival times of packets of flows  $f_1$  and  $f_2$  and  $A_{p_{f_1}} \geq A_{p_{f_2}}$ . This is a very loose description of this phenomenon, but we could not find a proper closed form that could be calculated without going through the example node by node. We discuss this further in chapter 4.

This finding makes it very obvious why the bound for the example given in [10] is tight. The network topology is designed in a way that this desynchronization cannot happen. Each crossflow only meets the foi at exactly one switch, where it causes the maximum burstiness increase for the foi, and then leaves. Another crossflow comes in at the exact same time as the first packet of  $f_1$ , causing another maximum burstiness increase, et cetera. However, for topologies where this is not the case, we showed that the bound can indeed be improved. This confirms our suspicion that e2e bounds that do not make use of the concatenation theorem cannot be tight.

## 4 Modeling Of The Interleaved Regulator In Network Calculus

In this chapter, we briefly describe failed attempts at accurately modeling the nodes and delay bound we have seen in [10]. We then discuss a min-plus service curve in detail, which represents the CBFS-IR network element. Here, we start with single CR servers, assuming identical and different rates, and expand our findings to an arbitrary number of servers in the network. After that, we apply these results to RL servers.

### 4.1 Initial Approaches

#### 4.1.1 Basic Network Calculus Modeling

After we finished analyzing the interleaved regulator, our first attempt at a min-plus SC was to model the CBFS-IR network element as a simple RL server convoluted with the IR SC given in [10]. The e2e delay bound would then have the shape

$$d_{e2e} = \bigotimes_{i=1}^n \beta_{R,T}^i \otimes \bigotimes_{i=1}^n \delta_{D(i,j,k,x)}^i. \quad (4.1)$$

We know the convolution of the first part has the closed form  $\beta_{\min\{R_i\}, T_{tot}}$ . The second part is equal to a right-shift by the sum of all delays in the IRs. This modeling approach leaves little optimization potential and yields a very bad delay bound in comparison to the paper bound [10]. The result is unsurprising, taking into account how the delay in the CBFS-IR network element works. No packet can experience a delay in both parts of the server, and the maximum delay a packet can experience is exactly the SC of the IR. This makes it clear that adding a RL server on top of the maximum delay a packet can experience can never yield a better delay bound.

#### 4.1.2 Greedy Shaper

After realizing that using the IR SC always yields a too pessimistic bound, we explored alternative ways to model the behaviour of the IR. In its essence, it reshapes

any flow that experienced a burstiness increase due to delays in the prior CBFS, such that the flow does not violate its constraints any more. This is exactly what a greedy shaper does as well. However, after choosing different curves to which the incoming flows would be reshaped, the resulting bounds were not satisfactory, as they were still too pessimistic. At this point, we decided that "simple" modeling approaches will most likely not lead to any meaningful results, and we started looking into different approaches instead. These are discussed in the following sections.

## 4.2 Max-plus Approach

Based on the idea of a minimum offset between any two successive packets of two different flows in the class aggregate (subsection 3.2.2), we explored the possibilities of the max-plus network calculus, since we look at functions of traffic, rather than time, with this framework.

At first glance, the ability to describe time distances between bits of different packets is exactly what is needed to model the minimum offset. Therefore, our first idea was to take a look at the capabilities of modeling this minimum offset in the max-plus calculus. However, we soon realized that even with a different look on network calculus, there was still no obvious way of describing the changing minimum offset between servers. We tried to bound the max-plus backlogged period, since we have seen during our experiments with the paper bound, that the minimum offset would have a direct effect on the backlogged period at a server. Having a bound on the backlogged period might get us closer to finding a closed expression for the minimum offset between packets of different flows in the aggregate.

### 4.2.1 Finding A Bound On The Backlogged Period In Max-Plus Calculus

While a bound on the backlogged period in min-plus network calculus exists, there is no proof for a max-plus equivalent. Since we know that the min-plus and max-plus calculus are equivalent [5], we know that there must be a bound in the max-plus calculus as well.

However, as discussed by Liebeherr [5], the basis of the respective bounds cannot be translated as is, i.e., the busy period in min-plus calculus and the busy sequence in max-plus calculus.

We already introduced the busy sequence in Definition 2.11. The busy period is defined as follows:

**Definition 4.1.** (Busy Period) Let  $I = [\kappa, \mu)$  be an interval, for which it holds that  $q(t) > 0$  for  $\kappa < \nu < \mu$ . In addition,  $q(t^-) = q(\mu) = 0$ .  $I$  is then a busy period.

A constructed example is given in [5]. There, we have different situations of the max-plus view having a busy sequence while there is no busy period in the min-plus view and vice versa. A workaround for this phenomenon is given in [5], as follows:

**Lemma 4.2.** *The time-domain backlog satisfies  $q(t) > 0$  iff there exists  $\nu, \mu$  with  $\nu > \mu$  such that  $T_A(\nu) < t \leq T_D(\mu)$ .*

Using this lemma, we can find a bound on the max-plus backlogged period.

**Theorem 4.3** (Maximum Length of a Max-Plus Backlogged Period). *Let  $T_A$  be an arrival time function that is constrained by a max-plus traffic envelope  $\lambda_E$ . The network element it traverses offers a strict max-plus service curve  $\gamma_S$ . The duration of the max-plus backlogged period of the network element is upper bounded by  $\gamma_S(\kappa)$ , with*

$$\kappa = \inf \{a > 0 \mid \lambda_E(a) \geq \gamma_S(a)\}. \quad (4.2)$$

*Proof.* We first introduce the notion of a max-plus backlogged period.

**Definition 4.4.** Let  $I = [\underline{\nu}, \kappa]$  be an interval with corresponding arrival time function  $T_A(\underline{\nu})$  and departure function  $T_D(\kappa)$ . The interval  $I_b = [T_A(\underline{\nu}), T_D(\kappa)]$  is a max-plus backlogged period iff for each  $t$  with  $T_A(\underline{\nu}) < t < T_D(\kappa)$  it holds that

$$q(t) > 0 \text{ iff } \exists \mu, \nu, \nu > \mu, \text{ where } T_A(\nu) < t \leq T_D(\mu). \quad (4.3)$$

Since we require the network element to provide a strict max-plus SC, we assume that the start of a max-plus backlogged period doubles as the start of a busy sequence. The reasoning behind this is that if we encounter a busy sequence, the delay as well as the backlog are non-zero. As a consequence, at the start of a busy sequence, we also fulfil the requirement towards the start of a backlogged period, proving this assumption to be correct.

We know the length of this max-plus backlogged period, as per definition it spans exactly over the interval  $I_b$ . This means that there is no time instant directly after  $T_D(\kappa)$  where the theorem 4.3 holds. The length of  $I_b$  is then  $T_D(\kappa) - T_A(\underline{\nu})$ . Recalling the definition of the strict max-plus SC (Definition 2.12), this allows us to upper bound the interval by  $\gamma_S(\kappa - \underline{\nu})$ .

We now need to determine the maximum interval length of  $I$ . We know that during the whole backlogged period  $I_b$ , all bits experience a backlog at the network element. This means that in order to bound the length of  $I$ , we need to find the maximum bit value  $\kappa$  for which the max-plus traffic envelope of  $\kappa$  is larger than the strict max-plus

SC of  $\kappa$ :

$$\begin{aligned}
\gamma_S(\kappa) &= \gamma_S((\kappa + \underline{\nu}) - \underline{\nu}) \\
&\stackrel{(2.12)}{\geq} T_D(\kappa + \underline{\nu}) - T_A(\underline{\nu}) \\
&\stackrel{\text{causality}}{\geq} T_A(\kappa + \underline{\nu}) - T_A(\underline{\nu}) \\
&\stackrel{(2.10)}{\geq} \lambda_E(\kappa).
\end{aligned}$$

This inequality holds for all  $\underline{\nu} < \nu \leq \kappa$ , as all bits in  $I$  experience a backlog, i.e., a bound on the maximum length of  $I$  is found once  $\gamma_S(\kappa) \leq \lambda_E(\kappa)$  holds. Since it holds for all  $\nu$  that  $T_A(\nu) < T_D(\nu)$ , we can further specify this condition to  $\gamma_S(\kappa) = \lambda_E(\kappa)$ .

As a result, the max-plus backlogged period bound is  $\gamma_S(\kappa)$  with  $\kappa = \inf \{a > 0 \mid \lambda_E(a) \geq \gamma_S(a)\}$ . □

## 4.2.2 Usefulness For Finding A Tighter Bound

While the idea of determining the max-plus backlogged period seemed promising at first, we realized that it achieves very little in regards to bounding the minimum offset. In fact, since the constraint of each flow does not change during its path, the only factor that changes the backlogged period bound is a different strict service curve at the different network elements along the path. This means that the desynchronization effect we are looking for cannot be reproduced by this bound.

As this bound was not determined before, we still included the result in this thesis, as it might prove useful in future work.

## 4.3 Finding A Min-Plus Service Curve

### 4.3.1 Effects Of The Interleaved Regulator

The reasons for the failure of previously presented min-plus network calculus (NC) approaches all boil down to the same issue, which is the lack of understanding on what the effects of the interleaved regulator (IR) are in particular. We know that the IR enforces the constraints of each flow while keeping the FIFO ordering of the class aggregate intact. This means that if a flow in the class aggregate does not adhere to its constraints, it is delayed until it does so. At this point, the remainder of this flows traffic, as well as conform traffic that comes directly afterwards, is allowed to pass the IR.

The reason for flow constraint violations is known. They always start in a CBFS that causes a burstiness increase, which in return causes delays of at least one flow in the class aggregate at the following IR. This IR then has to withhold any non-conform arriving flows until their burstiness increases have subsided, i.e., they adhere to their respective flow constraints again.

We know what the burstiness increase of a TB AC in a standard RL server looks like. The flow is delayed, hence its burst increases from  $b$  to  $b + rT$ , i.e., the TB AC is shifted to the left by  $rT$ . As a consequence, this flow needs to be delayed by exactly  $rT$  at the IR to adhere to its original TB constraint again.

With this knowledge, we can start to model a network, which is then used to derive an IR SC. We assume that we have an aggregate of two flows, traversing along a path of servers consisting of one CBFS and one IR element each. The flow aggregate consists of a foi and a crossflow. We can model the CBFS element fairly easily, as in its essence, this is a normal RL server, whose LO SC is available. We assume a latency  $T = 0$  for now, i.e., we look at a CR rather than a RL server. At a later point, we show that the modeling approach can be transitioned from a CR to a RL SC very easily by making use of basic min-plus algebra.

We first need to understand exactly what the burstiness increase looks like. As we consider a flow aggregate traversing the CR server, we can calculate the burstiness increase of this scenario. Let  $\alpha_1(t) = b_1 + r_1 \cdot t$  be the TB AC for the foi  $f_1$  and let  $\alpha_2(t) = b_2 + r_2 \cdot t$  be the TB AC for the crossflow  $f_2$ . The TB AC of the aggregate of  $f_1$  and  $f_2$  is then  $\alpha_{agg}(t) = b_1 + b_2 + (r_1 + r_2) \cdot t$ . The output bound for the aggregate AC is as follows:

$$\begin{aligned}\alpha'_{agg}(t) &= \alpha_{agg} \odot \beta(t) \\ &= \gamma_{r_1+r_2, b_1+b_2} \odot \beta_{R,0}(t) \\ &= \gamma_{r_1+r_2, b_1+b_2}(t).\end{aligned}$$

As can be seen, the aggregate experiences no burstiness increase at a CR server. While this is an unexpected observation, the matter becomes more interesting once we calculate the burstiness increase for the specific flows of the aggregate. For this, we first need to determine the FIFO LOSC of the CR server and  $\alpha_2$ . The general shape of this LOSC is

$$\beta_{\theta}^{LO}(t) = \begin{cases} [\beta(t) - \alpha_2(t - \theta)]^+, & t > \theta, \\ 0 & \text{otherwise.} \end{cases} \quad (4.4)$$

We then determine the closed form of case one for the optimal  $\theta = \frac{b_2}{R}$  [7],[6]:

$$\begin{aligned}\beta_{\theta}^{LO}(t) &= [\beta(t) - \alpha_2(t - \theta)]^+ \\ &= \left[ \beta(t) - \alpha_2\left(t - \frac{b_2}{R}\right) \right]^+ \\ &= \left[ Rt - \left( b_2 + r_2 \cdot \left( t - \frac{b_2}{R} \right) \right) \right]^+\end{aligned}$$

$$\begin{aligned}
 &= \left[ Rt - r_2 t + r_2 \frac{b_2}{R} - b_2 \right]^+ \\
 &= \left[ t(R - r_2) + b_2 \cdot \left( \frac{r_2}{R} - 1 \right) \right]^+ \\
 &= \left[ \left( 1 - \frac{r_2}{R} \right) \cdot Rt + b_2 \cdot \left( \frac{r_2}{R} - 1 \right) \right]^+ \\
 &= \left[ \left( 1 - \frac{r_2}{R} \right) \cdot (Rt - b_2) \right]^+ \\
 &= (R - r_2) \cdot \left[ t - \frac{b_2}{R} \right]^+ \\
 &= \beta_{R-r_2, \frac{b_2}{R}}.
 \end{aligned}$$

The LOSC for  $\alpha_2$  is of the same shape; we just need to replace the latency with  $\frac{b_1}{R}$ . With this, we can calculate the output bound of the crossflow  $f_2$ , which gives us its resulting burstiness increase:

$$\begin{aligned}
 \alpha'_2(t) &= \alpha_2 \odot \beta_{\theta}^{LO}(t) \\
 &= \gamma_{r_2, b_2 + r_2 T}(t) \\
 &= b_1 + r_2 \cdot t + r_2 \frac{b_1}{R} \\
 &= b_1 + r_2 \cdot \left( t + \frac{b_1}{R} \right).
 \end{aligned}$$

We have seen before that the aggregate experiences no burstiness increase, but the flows of the aggregate do. This means that the burst of at least one of the flows is partially delayed, to adhere to the aggregate constraint. If we want to reach the worst case for the foi, its burst has to be delayed as much as possible, i.e., we assume that the crossflow is delayed in the previous CBFS, to reach its maximum burstiness increase  $r_2 \frac{b_1}{R}$ , which then results in an increased burst of  $b_2 + r_2 \frac{b_1}{R}$ . Respectively, the foi has a reduced burst of  $b_1 - r_2 \frac{b_1}{R}$ . Normally, this would be the wrong assumption for reaching the worst-case delay bound of the foi. To still be able to make accurate statements about the delay of the foi, we assume that there is an infinitesimal portion of the foi that is delayed until the remainder of the aggregate flow is served.

This gives us a good idea on what a service curve for the IR might look like. It serves the crossflow burst  $b_2$  first, then transmits traffic of  $f_2$  with rate  $r_2$ , until its burstiness increase has been negated, i.e., until the left shift of  $r_2 \frac{b_1}{R}$  has passed. Then, the remaining burst of the foi is served. After this, both flows will have regained their original shape, and the IR offers the aggregate rate to the flow aggregate.

This can be expressed in an IR SC:

$$\beta_{IR}(t) = \begin{cases} 0, & t \leq 0, \\ b_2 + r_2 \cdot t, & 0 < t \leq r_2 \frac{b_1}{R^2}, \\ b_1 + b_2 + (r_1 + r_2) \cdot (t - r_2 \frac{b_1}{R^2}) & \text{otherwise.} \end{cases} \quad (4.5)$$



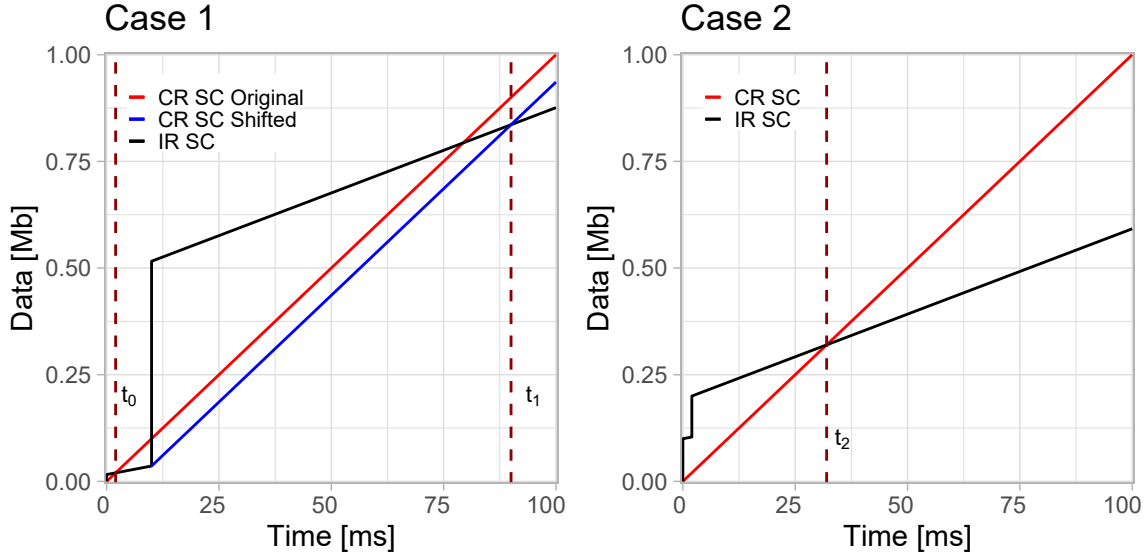


Figure 4.1: Case distinction that is made in regards to the convolution of a CR and IR SC.

From now on, we refer to the point in time  $r_2 \frac{b_1}{R^2}$  as jump point. This is the time that is needed to negate the right-shift of the crossflow.

### 4.3.2 Constructing A Min-Plus CBFS-IR Service Curve

Once we have acquired the IR SC, we should be able to determine the service curve of the combined CBFS-IR network element. There are a few challenges in doing so. We need to consider where the CR SC intersects the IR SC. Depending on whether the intersection point lies before or after the jump point, the convolution of both curves looks vastly different. We also need to consider the parameter requirements to fall into these cases.

We first look at the different cases, in regards to the convolution of a CR and IR SC. Figure 4.1 shows this distinction. Case 1 is entered when the CR SC intersects the IR SC before its jump point. The resulting convolution consists of four segments. The first and third segments consist of the (shifted) CR SC, where the third segment starts at the jump point of the IR SC it is convoluted with. To determine the interval limits of the segments, we need to calculate the two intersection points  $t_0, t_1$ , which are represented by the two dark red vertical lines in Figure 4.1. If  $t_0$  is larger than the jump point of the IR SC, we enter case 2. Here, we have a SC with two segments, as result of the convolution. To know the segment limits, we calculate the intersection point  $t_2$ .

### Specifying The Cases By Parameter Requirements

There are several parameters determining the case the convolution falls into, those being the bursts  $b_1, b_2$  of the aggregate flows  $f_1, f_2$ , as well as the server rate  $R$ . First, we find closed formulas for the three intersection points  $t_0, t_1, t_2$ . After that, we propose an equation that allows us to derive specific parameter requirements for the two cases.

Find  $t_0$  :

$$\begin{aligned} R t_0 &= b_2 + r_2 \cdot t_0 \\ \Leftrightarrow t_0(R - r_2) &= b_2 \\ \Leftrightarrow t_0 &= \frac{b_2}{R - r_2}. \end{aligned}$$

Find  $t_1$  :

$$\begin{aligned} b_2 + r_2^2 \frac{b_1}{R^2} + R \cdot \left( t_1 - r_2 \frac{b_1}{R^2} \right) &= b_1 + b_2 + (r_1 + r_2) \cdot \left( t_1 - r_2 \frac{b_1}{R^2} \right) \\ \Leftrightarrow t_1 \cdot (R - (r_1 + r_2)) &= b_1 + r_2 \frac{b_1}{R^2} \cdot (R - (r_1 + r_2) - r_2) \\ \Leftrightarrow t_1 &= \frac{b_1 + r_2 \frac{b_1}{R^2} \cdot (R - r_1 - 2r_2)}{R - (r_1 + r_2)}. \end{aligned}$$

Find  $t_2$  :

$$\begin{aligned} R \cdot t_2 &= b_1 + b_2 + (r_1 + r_2) \cdot \left( t_2 - r_2 \frac{b_1}{R^2} \right) \\ \Leftrightarrow t_2 \cdot (R - (r_1 + r_2)) &= b_1 + b_2 - (r_1 + r_2) \cdot r_2 \frac{b_1}{R^2} \\ \Leftrightarrow t_2 &= \frac{b_1 + b_2 - (r_1 + r_2) \cdot r_2 \frac{b_1}{R^2}}{R - (r_1 + r_2)}. \end{aligned}$$

Looking closely at the two cases in Figure 4.1, we know that we are in case 2 if at most, the CR SC intersects the IR SC at the lower end of the jump point. We can express this with the equation

$$r_2 \frac{b_1}{R^2} = \frac{b_2}{R - r_2}. \quad (4.6)$$

With this equation, we can now derive parameter requirements for cases 1 and 2. We start with the bursts.

We derive a lower bound on the value of  $b_1$ , and an upper bound on the value of  $b_2$ .

We start with  $b_1$ , where the largest value it can take on and still fall into case 2 is

$$\begin{aligned} r_2 \frac{b_1}{R^2} &= \frac{b_2}{R - r_2} \\ \Leftrightarrow r_2 b_1 &= \frac{b_2 R^2}{R - r_2} \\ \Leftrightarrow b_1 &= \frac{b_2 R^2}{r_2 \cdot (R - r_2)}. \end{aligned}$$

The largest value  $b_2$  can take on and still fall into case 1 is

$$\begin{aligned} r_2 \frac{b_1}{R^2} &= \frac{b_2}{R - r_2} \\ \Leftrightarrow b_2 &= (R - r_2) \cdot r_2 \frac{b_1}{R^2}. \end{aligned}$$

Using the threshold formula for  $b_2$ , we can further derive a restriction for the server rate  $R$ :

$$\begin{aligned} b_2 &= (R - r_2) \cdot r_2 \frac{b_1}{R^2} \\ \Leftrightarrow R^2 b_2 &= r_2 b_1 R - r_2^2 b_1 \\ \Leftrightarrow R^2 b_2 - r_2 b_1 R + r_2^2 b_1 &= 0. \end{aligned}$$

The resulting quadratic formula can be solved for  $R$ , using Muller's method:

$$R_{1,2} = \frac{2r_2^2 b_1}{r_2 b_1 \pm \sqrt{(-r_2 b_1)^2 - 4b_1 b_2 r_2^2}}.$$

We ignore the negative case, and get the maximum server rate  $R$  to still fall within case 2, i.e., if the server rate is larger, we enter case 1.

The server rate requirement has limited uses while working with practical examples, as the stability condition is disregarded. While experimenting with parameters, the minimum server rate for case 1 was always well below the minimum required server rate, as per the stability condition. Therefore, a better design choice is to choose a server rate and calculate the burst thresholds accordingly.

### Convolution Results For Cases 1 And 2

As we have already seen above, the convolution of a CR and IR SC can have two outcomes, either having two or four segments in the resulting curve. We show the results of the convolutions here, but postpone the formal proof for these to a later point in time. The correctness of the curves can be proven by a graphical convolution.

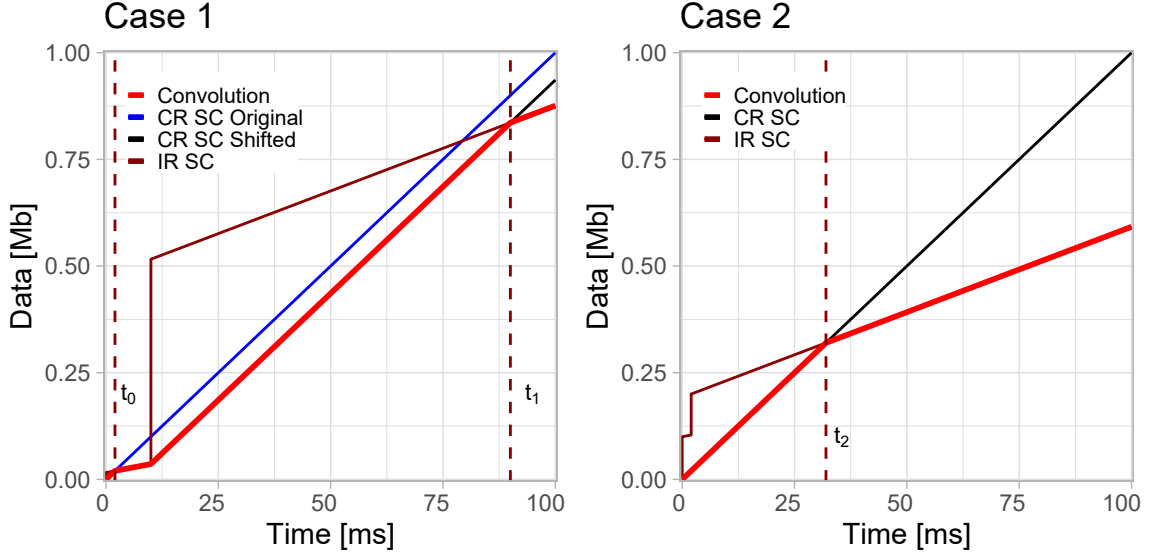


Figure 4.2: Graphical convolution results of the two previously introduced examples for cases 1 and 2.

Since the convolution of two curves always follows the smaller of the two curves for each point in time, we know what the result must look like.

For case 1, the convolution follows the (shifted) CR SC in segments one and three, then swaps over to the crossflow AC between  $t_0$  and the jump point, and to the aggregate AC after  $t_1$ . This can be seen in Figure 4.2. In case 2, we have only two segments. The first segment follows the CR SC until its intersection point  $t_2$  with the aggregate AC, then follows this curve after passing  $t_2$ .

The convolution results for cases 1 and 2 are then as follows:

$$\beta_{CR1,IR}(t) = \begin{cases} Rt, & 0 \leq t < t_0, \\ b_2 + r_2 t, & t_0 \leq t \leq r_2 \frac{b_1}{R^2}, \\ b_2 + r_2^2 \frac{b_1}{R^2} + R \cdot \left( t - r_2 \frac{b_1}{R^2} \right), & r_2 \frac{b_1}{R^2} < t < t_1, \\ b_1 + b_2 + (r_1 + r_2) \cdot \left( t - r_2 \frac{b_1}{R^2} \right), & t_1 \leq t, \end{cases} \quad (4.7)$$

$$\beta_{CR2,IR}(t) = \begin{cases} Rt, & 0 \leq t < t_2, \\ b_1 + b_2 + (r_1 + r_2) \cdot \left( t - r_2 \frac{b_1}{R^2} \right), & t_2 \leq t. \end{cases} \quad (4.8)$$

It holds for both SCs that they are in  $\mathcal{F}_0$ .

This leaves us with a potential min-plus NC modeling of the interleaved regulator, which we analyze in detail in chapter 5. In the following sections, we deal with the specifics of these two curves in regards to multiple servers with either identical or different rates, as well as the transition to RL servers.

## 4.4 Expanding The Min-Plus CBFS-IR Service Curve To Multiple CR Servers

To calculate a meaningful service curve for multiple servers, we first introduce the concatenation of two servers [6].

**Theorem 4.5** (Server Concatenation). *Let an arrival process  $A_{sys} = A_1$  traverse along a tandem<sup>1</sup> of servers  $S_1$  and  $S_2$ . They offer the service curves  $\beta_1$  and  $\beta_2$ , respectively. The concatenation of the two systems,  $S = \langle S_1, S_2 \rangle$ , offers a service curve of  $\beta_{e2e} = \beta_1 \otimes \beta_2$ .*

This also holds for an arbitrary number of servers. We use this theorem to extend the CBFS-IR service curves, which we found in the previous section, to multiple servers. To do this, we determine the convolution of two curves of each case.

### 4.4.1 Convolution Of Two $\beta_{CR2,IR}$ Service Curves

Convoluting SCs with multiple cases each by hand proves not to be as straightforward as expected. In the case of  $\beta_{CR2,IR}$ , this is still manageable. But once we look at  $\beta_{CR1,IR}$  SCs, we need to apply a different method for calculating the convolution. Hence, we first define the convolution result for two  $\beta_{CR2,IR}$  curves.

**Theorem 4.6.** *Assume two services curves  $\beta_1, \beta_2$  are  $\beta_{CR2,IR}$  service curves with respective intersection points  $t_2^1, t_2^2$ . The convolution of the two service curves is*

$$\beta_{\beta_1, \beta_2}(t) = \begin{cases} \min\{R_1, R_2\} \cdot t, & 0 \leq t < \max\{t_2^1, t_2^2\}, \\ b_1 + b_2 + (r_1 + r_2) \cdot t, & \max\{t_2^1, t_2^2\} \leq t. \end{cases} \quad (4.9)$$

*Proof.* Calculate the convolution  $\beta_1 \otimes \beta_2$  by calculating the result for each different case:

$$\beta_1 \otimes \beta_2(t) = \inf_{0 \leq s \leq t} \{\beta_1(t-s) + \beta_2(s)\}.$$

**Case 1:**  $t < \min\{t_2^1, t_2^2\}$

$$\begin{aligned} \beta_1 \otimes \beta_2(t) &= \inf_{0 \leq s \leq t} \{R_1(t-s) + R_2s\} \\ &= \min\{R_1, R_2\} \cdot t, \end{aligned}$$

where  $s = t$  if  $R_1 > R_2$  and  $s = 0$  if  $R_1 \leq R_2$ .

---

<sup>1</sup>A tandem is a sequence of servers.

**Case 2:**  $\max\{t_2^1, t_2^1\} = x \leq t$

$$\begin{aligned}
\beta_1 \otimes \beta_2(t) &= \inf_{0 \leq s < x} \{R_1(t-s) + R_2s\} + \inf_{x \leq s \leq t} \{R_1(t-s) + b_1 + b_2 + (r_1 + r_2) \cdot t\} \\
&\stackrel{\text{Case 1}}{=} \{\min\{R_1, R_2\} \cdot t\} \wedge \inf_{x \leq s \leq t} \{R_1(t-s) + b_1 + b_2 + (r_1 + r_2) \cdot t\} \\
&\stackrel{\text{"s=t"}}{=} \{\min\{R_1, R_2\} \cdot t\} \wedge \{b_1 + b_2 + (r_1 + r_2) \cdot t\} \\
&= \{b_1 + b_2 + (r_1 + r_2) \cdot t\}.
\end{aligned}$$

□

#### 4.4.2 Finding A General Convolution Result For $\beta_{CR2,IR}$ Servers

Since repeatedly convoluting  $\beta_{CR2,IR}$  SCs always yields the same two-segmented SC that we have seen in equation 4.9, finding a convolution result for  $n$  servers is very easy. We simply need to find the minimum rate  $R_i$  out of the  $n$  servers we convolute with each other, and calculate its intersection point  $t_2^i$  with the aggregate AC. The general case of the convolution is then

$$\beta_{e2e} = \bigotimes_{i=1}^n \beta_i(t) = \begin{cases} \min\{R_i\} \cdot t, & 0 \leq t < \max\{t_2^i\}, \\ b_1 + b_2 + (r_1 + r_2) \cdot t & \text{otherwise.} \end{cases} \quad (4.10)$$

This e2e SC also holds for the scenario where we convolute case 1 with case 2 SCs.

#### 4.4.3 Convolution Of Two $\beta_{CR1,IR}$ Service Curves

In contrast to  $\beta_{CR2,IR}$  SCs, calculating the convolution for two  $\beta_{CR1,IR}$  SCs is not as straightforward. Attempts at calculating the convolution by hand failed, due to a lack of knowledge of which segments are created during the convolution. To get a general idea of the shape of the result, we implemented the convolution in Java and plotted the obtained data points. After still failing to calculate the convolution by hand while knowing the result already, we decided to obtain the convolution result using point clouds [12].

##### Point Clouds

Point clouds are used to describe information points that are created during the application of the  $\mathfrak{P}$ -transform [12]. This transform is able to calculate the convolution of functions that are neither strictly convex or concave. Since the basis of our convolution is exactly a function that is neither convex nor concave, this is a perfect match. The specifics of the transform are described in [12]. We are only interested in its application to the min-plus convolution, specifically the result of the convolution for arbitrary functions:

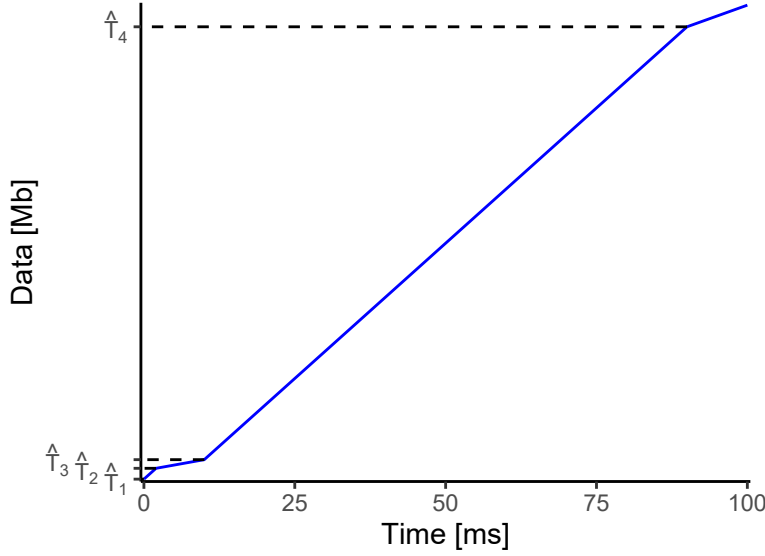


Figure 4.3: Convex and concave inflection points of the chosen  $\beta_{CR1,IR}$  example curve.

**Theorem 4.7** (Min-Plus Convolution For Arbitrary Functions [12]). *Let  $f(t), g(t)$  be two arbitrary functions. Their min-plus convolution is given by*

$$(f \otimes g)(t) = \min \left\{ \min_{1 \leq i \leq \hat{m}} \{f(\hat{T}_i^f) + g(t - \hat{T}_i^f)\}, \min_{1 \leq j \leq \hat{n}} \{g(\hat{T}_j^g) + f(\hat{T}_j^g)\} \right\}. \quad (4.11)$$

$\hat{T}_i^f, \hat{T}_j^g$  are the convex inflection points of functions  $f$  and  $g$ , i.e., the points where the functions have left turns.  $\hat{m}$  is the number of convex inflection points that  $f$  has,  $\hat{n}$  the respective amount for  $g$ .

A graphical representation of this theorem is that we shift both functions to start at each of the convex inflection points of the other function. We then select the point that is furthest to the right and bottom for each point in time, and put these minimal points together into a curve. This newly created curve then equals the min-plus convolution of the functions  $f$  and  $g$ . We use this theory in the following sections to calculate the convolution of two  $\beta_{CR1,IR}$  SCs and to define a general formula for the convolution of  $n$  servers.

### Applying Point Clouds To Calculate The Convolution For Identical Server Rates

We first analyze the  $\beta_{CR1,IR}$  SC, to determine its convex inflection points. Let  $b_1 = 0,5$  Mbit,  $b_2 = 0,016$  Mbit,  $r_1 = r_2 = 2$  Mbit/s, and  $R = 10$  Mbit/s. The resulting curve can be seen in Figure 4.3. Here, we also marked all inflection points of the curve. While  $\hat{T}_1$  and  $\hat{T}_3$  are convex inflection points,  $\hat{T}_2$  and  $\hat{T}_4$  are concave inflection points. Since we are only interested in the convex points, we can discard  $\hat{T}_2$  and  $\hat{T}_4$ .

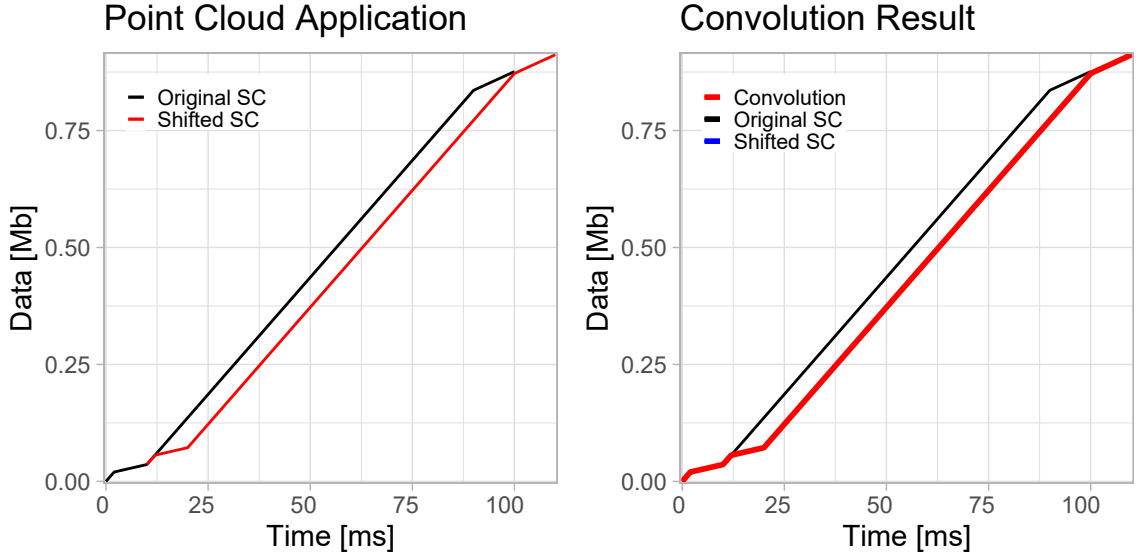


Figure 4.4: Application of point clouds to calculate the convolution (left) and the convolution result (right).

In the next step, we assume that we convolute two identical curves that are modeled as in Figure 4.3. Calculating the convolution by hand is very time-intensive, and does not yield a proper result, which matches the result of the simulation we mentioned previously. Instead, we use point clouds to obtain a result. We know that the convex inflection points are  $\hat{T}_1$  and  $\hat{T}_3$ , so we shift the original curve to start at  $\hat{T}_1$  and  $\hat{T}_3$ , then take the minimum resulting curve as the convolution result. Since both curves are identical, we can disregard  $\hat{T}_1$ , and only need to shift the curve to start at  $\hat{T}_3$ .

The SC shift as well as the convolution result are shown in Figure 4.5. As we can see, the convolution causes the creation of an additional stair after the initial one having the same shape. This follows from the assumption that both functions, which we convolute with each other, are identical. We later look at examples where this is not the case.

The resulting convolution  $\beta_{CR11}$  is defined as follows:

$$\beta_{CR11}(t) = \bigotimes_{i=1}^2 \beta_{CR1,IR}^i(t) = \begin{cases} Rt, & 0 \leq t < t_0, \\ b_2 + r_2 t, & t_0 \leq t \leq r_2 \frac{b_1}{R^2}, \\ b_2 + r_2^2 \frac{b_1}{R^2} + R(t - r_2 \frac{b_1}{R^2}), & r_2 \frac{b_1}{R^2} < t < r_2 \frac{b_1}{R^2} + t_0, \\ 2b_2 + r_2 t, & r_2 \frac{b_1}{R^2} + t_0 \leq t \leq 2r_2 \frac{b_1}{R^2}, \\ 2b_2 + 2r_2^2 \frac{b_1}{R^2} + R(t - 2r_2 \frac{b_1}{R^2}), & 2r_2 \frac{b_1}{R^2} < t < r_2 \frac{b_1}{R^2} + t_1, \\ b_1 + b_2 + 2r_2^2 \frac{b_1}{R^2} + (r_1 + r_2) \cdot (t - 2r_2 \frac{b_1}{R^2}) & \text{otherwise.} \end{cases} \quad (4.12)$$



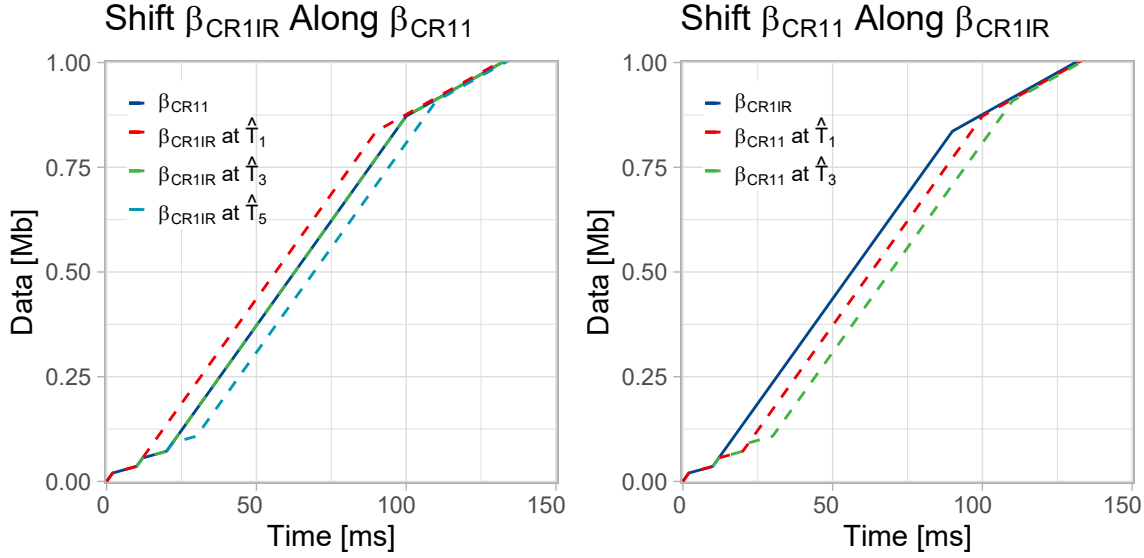


Figure 4.5: Point cloud application to calculate the convolution result of one  $\beta_{CR1,IR}$  SC and one  $\beta_{CR11}$  SC.

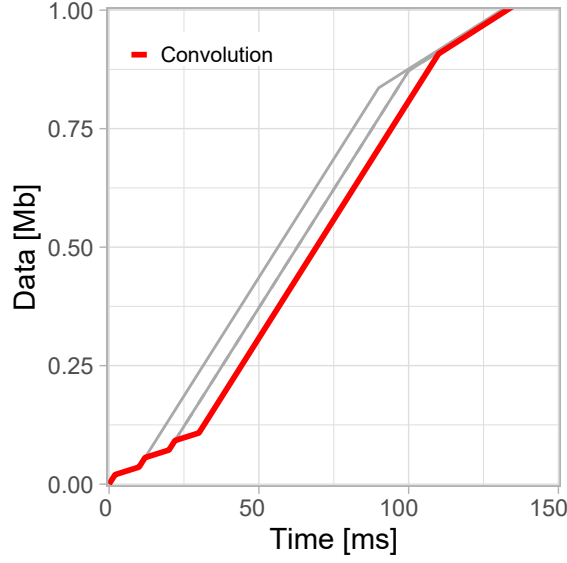
#### 4.4.4 Finding A General Convolution Result For $\beta_{CR1,IR}$ Servers With Identical Rates

Before we define the convolution of the general case, we first calculate another convolution using point clouds to ensure that our understanding of how the convolution works is correct. We assume that each additional convolution adds another stair, which is identical to the stair of the original SC. The general case then has as many stairs as there are servers in the convolution. After that, we add the second to last segment of the original  $\beta_{CR1,IR}$  SC, shifted to the right by the width and to the top by the height of the sum of the stairs, and then we end in the aggregate AC as the last segment.

Figure 4.5 shows the point cloud application that is used to calculate the convolution  $\beta_{CR11} \otimes \beta_{CR1,IR}$ . Equation 4.11 states that we have to shift each of the two functions  $\beta_{CR11}$  and  $\beta_{CR1,IR}$  along the convex inflection points of the respective other function, then take the resulting minimum of both shifts to get the convolution result. Here, we can see that both shifts result in the same minimum curve (compare Figure 4.6). It follows the two stairs of  $\beta_{CR11}$ , adds another stair from  $\beta_{CR1,IR}$  afterwards, and finally follows the two shifted last segments we have already seen in the previous convolution.

This confirms our assumption of how each additional convolution changes the curve, and allows us to define a general convolution result. Let  $n$  be the number of servers, with  $R$  their identical server rate. We then build the convolution as follows:

- First build the stairs. For  $m \leq n, m \in \mathbb{N}^+$ :


 Figure 4.6: Convolution result of  $\beta_{CR11} \otimes \beta_{CR1,IR}$ .

- $(m-1) \cdot b_2 + (m-1) \cdot r_2^2 \frac{b_1}{R^2} + R(t - (m-1) \cdot r_2 \frac{b_1}{R^2})$   
over the interval  
 $(m-1) \cdot r_2 \frac{b_1}{R^2} < t < (m-1) \cdot r_2 \frac{b_1}{R^2} + t_0$ ,
- $m \cdot b_2 + r_2 t$   
over the interval  
 $m \cdot b_2 + r_2 t, (m-1) \cdot r_2 \frac{b_1}{R^2} + t_0 \leq t \leq m \cdot r_2 \frac{b_1}{R^2}$ .
- Afterwards, build the last two segments of the curve:
  - $n \cdot b_2 + n \cdot r_2^2 \frac{b_1}{R^2} + R(t - n \cdot r_2 \frac{b_1}{R^2})$   
over the interval  
 $n \cdot r_2 \frac{b_1}{R^2} < t < (n-1) \cdot r_2 \frac{b_1}{R^2} + t_1$ ,
  - $b_1 + n \cdot b_2 + (n-1) \cdot r_2^2 \frac{b_1}{R^2} + (r_1 + r_2) \cdot (t - n \cdot r_2 \frac{b_1}{R^2})$   
over the interval  
 $(n-1) \cdot r_2 \frac{b_1}{R^2} + t_1 \leq t$ .

#### 4.4.5 Extending The General Convolution Result For $\beta_{CR1,IR}$ Servers To Different Rates

While building stairs when assuming similar server rates is very straightforward, we need to take special care when building the general case for different rates. In fact, since server rates in a tandem possibly fluctuate heavily, we can encounter the case that a server with a very large rate, compared to the smallest rate in the tandem, does

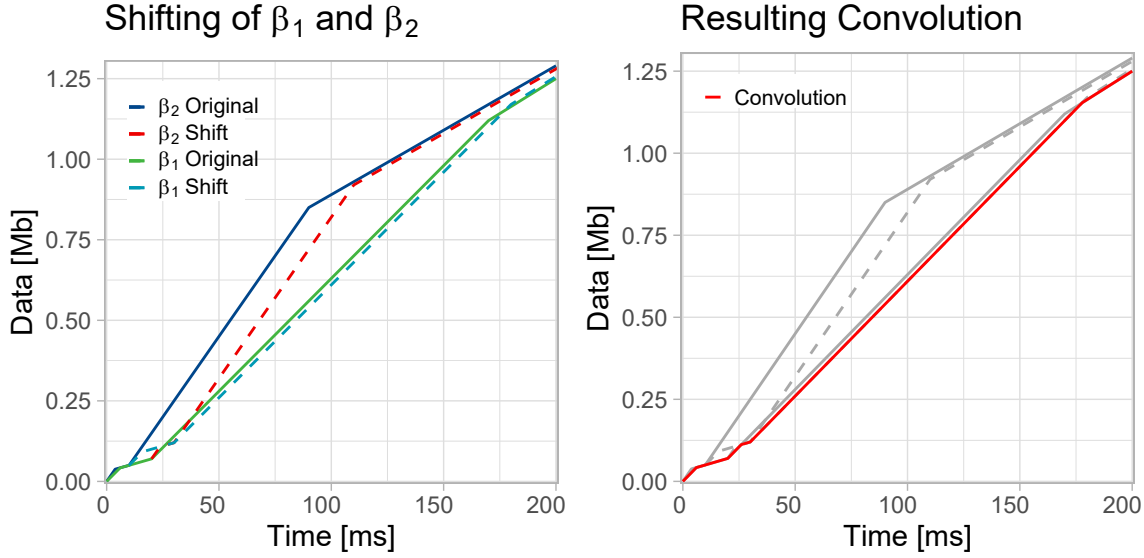


Figure 4.7: First convolution between  $\beta_1$  and  $\beta_2$ , which adheres to the requirement for stair creation.

not create a stair when being convoluted. We look at two example convolutions to show this effect.

Let  $b_1 = 0.5$  Mbit,  $b_2 = 0.016$  Mbit,  $r_1 = r_2 = 2$  Mbit/s. Assume there are three CR servers  $\beta_1, \beta_2, \beta_3$ , in a tandem with server rates  $R_1 = 7$  Mbit/s,  $R_2 = 10$  Mbit/s,  $R_3 = 25$  Mbit/s. We first convolute  $\beta_1$  and  $\beta_2$  (compare Figure 4.7). In this case, the convolution result is as expected. We gain a second stair, then build the two last segments through shifting. For the second convolution, the result is not as clear any more. As we can see in Figure 4.8, the shifts of  $\beta_3$  with rate  $R_3$  do not intersect the first convolution result  $\beta_{1,2}$ , i.e.,  $R_3$  is too large in relation to  $R_1$  to create a stair during the convolution. This results in the convolution being identical to the previous one.

We can determine a condition for the creation of stairs during the convolution of CR servers with different rates. We select the server with the smallest rate, then compare its  $t_0$  value to the jump points of all other server curves. It must hold that the jump point of a SC is larger than the  $t_0$  value of the lowest rate server. If this does not hold, no stair will be created during the convolution. Taking another look at Figure 4.8, we can see this condition in effect. The red curve has its jump point well before the blue curve changes into the curve of  $\alpha_2$ , i.e., before  $t_0$ . We remove any server that does not meet this requirement, and calculate the convolution with the remaining servers.

The actual definition of the general case is done in section 4.5.3. We simply set  $n$  to the number of servers that adhere to the requirement specified above, and let  $T = 0$  for all servers.

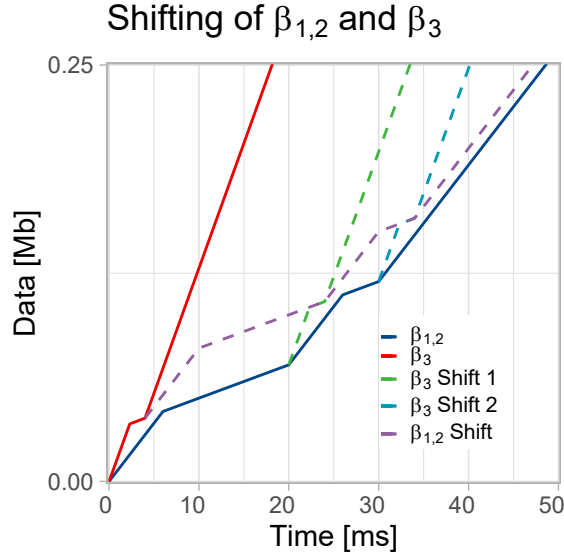


Figure 4.8: Second convolution between  $\beta_{1,2}$  and  $\beta_3$ , which does not adhere to the stair creation requirement. Hence, the convolution is identical to  $\beta_{1,2}$ .

## 4.5 Specifying The General Convolution For Rate-Latency Servers

Having talked in detail in the previous sections about the convolution for CR servers, we now look at RL servers instead. Recall their definition:

$$\beta_{R,T}(t) = \begin{cases} 0, & t \leq T, \\ R \cdot [t - T] & \text{otherwise.} \end{cases} \quad (4.13)$$

This means that the server does not offer a rate to a flow immediately, but instead lets it wait for time  $T$ , also called the latency, and then serves the flow with the right-shifted constant-rate curve  $R[t - T]$  we have seen before.

We have mentioned before that the extension from CR to RL servers can be done by making use of min-plus algebra. It holds that

$$\beta_{R,T}(t) = \beta_{R,0} \otimes \delta_T, \quad (4.14)$$

where we make use of the fact that the convolution with the impulse function  $\delta_d$  always results in a right-shift by the provided  $d$ . Hence, assuming  $d = T$  results in a right-shift of the CR curve  $\beta_{R,0}$ , turning it into  $\beta_{R,T}$ . This is a very convenient property for the results we obtained from CR servers. We can add as many impulse functions to the convolution as there are servers in the convolution, where each impulse function sets its  $d$  to one of the latencies of the RL servers that we want to convolute. This allows us to add the latency of each server in the convolution as a right-shift to the

convolution result of the CR servers, which in turn gives us the convolution result for the respective RL servers.

This allows us to reuse the general convolution results we have found in sections 4.3 and 4.4, by adding right-shifts and basic latencies.

#### 4.5.1 Adjusting The Single Server Service Curves From Constant-Rate To Rate-Latency

In subsection 4.3.2, we specified the results of the convolution of an IR and CR SC for the two possible cases the convolution could take on (compare Figure 4.2). We revisit this convolution shortly, to show exactly how the jump from CR to RL is made.

First, we adjust the CR curve to be an RL curve. This is achieved by changing the curve's latency from 0 to  $T$ . As a result, the intersection points  $t_0, t_1, t_2$ , which we determined for CR servers and the IR SC, are shifted to the right by  $T$ . As we shift one of the curves to the right by the same value, the point in time where the curves intersect is shifted as well. We name these new points  $t_0^*, t_1^*, t_2^*$ . They are defined as follows:

$$t_0^* = t_0 + T = \frac{b_2}{R - r_2} + T, \quad (4.15)$$

$$t_1^* = t_1 + T = \frac{b_1 + r_2 \frac{b_1}{R^2} \cdot (R - r_1 - 2r_2)}{R - (r_1 + r_2)} + T, \quad (4.16)$$

$$t_2^* = t_2 + T = \frac{b_1 + b_2 - (r_1 + r_2) \cdot r_2 \frac{b_1}{R^2}}{R - (r_1 + r_2)} + T. \quad (4.17)$$

We also need to adjust the parameter requirements for cases 1 and 2, as  $t_0$  changes to  $t_0^*$  for RL servers, and thus all derivations from equation 4.6 need to be changed as well:

$$r_2 \frac{b_1}{R^2} = \frac{b_2}{R - r_2} + T. \quad (4.18)$$

The left part of the equation does not change, as the IR is defined independently of its CBFS counterpart, i.e., changes to the latency do not affect it. First, we find the maximum value  $b_1$  can take on and still fall into case 2:

$$\begin{aligned} r_2 \frac{b_1}{R^2} &= \frac{b_2}{R - r_2} + T \\ \Leftrightarrow r_2 b_1 &= \frac{b_2 R^2}{R - r_2} + R^2 T \\ \Leftrightarrow b_1 &= \frac{b_2 R^2}{r_2 \cdot (R - r_2)} + \frac{R^2 T}{r_2} \\ \Leftrightarrow b_1 &= \frac{R^2 (b_2 + T(R - r_2))}{r_2 \cdot (R - r_2)}. \end{aligned}$$

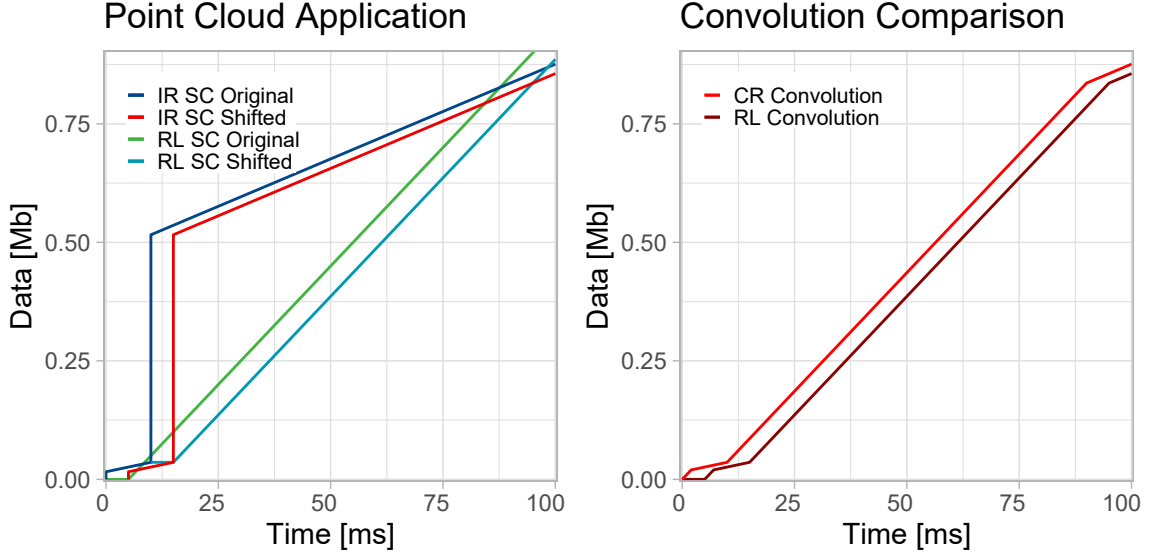


Figure 4.9: Calculation of  $\beta_{RL} \otimes \beta_{IR}$  using point clouds (left). Comparison of the resulting convolution to the  $\beta_{CR} \otimes \beta_{IR}$  curve (right).

Then, we find the maximum value  $b_2$  can take on and fall into case 1:

$$\begin{aligned}
 r_2 \frac{b_1}{R^2} &= \frac{b_2}{R - r_2} + T \\
 \Leftrightarrow r_2 \frac{b_1}{R^2} &= \frac{b_2 + T(R - r_2)}{R - r_2} \\
 \Leftrightarrow b_2 + T(R - r_2) &= r_2 \frac{b_1}{R^2} \cdot (R - r_2) \\
 \Leftrightarrow b_2 &= r_2 \frac{b_1}{R^2} \cdot (R - r_2) - T(R - r_2).
 \end{aligned}$$

We omit the minimum server rate, as it has limited use for practical applications. Then, we define the convolution for a  $\beta_{IR}$  and  $\beta_{R,T}$  curve:

$$\begin{aligned}
 \beta_{RL1,IR}(t) &= \\
 \beta_{R,T} \otimes \beta_{IR}(t) &= \begin{cases} 0, & 0 \leq t \leq T, \\ R(t - T), & T < t < t_0^*, \\ b_2 + r_2(t - T), & t_0^* \leq t \leq r_2 \frac{b_1}{R^2} + T, \\ b_2 + r_2 \frac{b_1}{R^2} + R(t - T - r_2 \frac{b_1}{R^2}), & r_2 \frac{b_1}{R^2} + T < t < t_1^*, \\ b_1 + b_2 + (r_1 + r_2) \cdot (t - T - r_2 \frac{b_1}{R^2}), & t_1^* \leq t, \end{cases} \quad (4.19)
 \end{aligned}$$

$$\begin{aligned}
 \beta_{RL2,IR}(t) &= \\
 \beta_{R,T} \otimes \beta_{IR}(t) &= \begin{cases} 0, & 0 \leq t \leq T, \\ R(t - T), & T < t < t_2^*, \\ b_1 + b_2 + (r_1 + r_2) \cdot (t - T - r_2 \frac{b_1}{R^2}), & t_2^* \leq t. \end{cases} \quad (4.20)
 \end{aligned}$$

Comparing these SCs to the ones we defined for CR servers makes the "trick" of using min-plus algebra as described above very obvious. We add a new segment, before the old first segment, representing the latency  $T$ , then append the convolution result for a CR server while shifting it to the right by  $T$ .

To doublecheck this assumption, we compare the results of the RL convolution to the CR convolution, using point clouds. In Figure 4.9, we can see the shifts of the  $\beta_{RL}$  and  $\beta_{IR}$  curves along the convex inflection points of the respective other curve. This results in the dark red curve on the right, and confirms our previous claim that the extension from CR to RL servers can be made by adding the latency of the RL server on top of the results we have obtained for CR servers.

## 4.5.2 Convolution Of RL Servers With Identical Rates

Adapting the general convolution case for RL servers boils down to adding latency at the right places, accomodating for right-shifts. We start with the assumption that we have  $n$  servers, which all have identical rates. We only look at the convolution of case 1 SCs, as the distinction between identical and different rates has little impact on the results for case 2 SCs.

### Adjustment Of $\beta_{RL,IR}$ Convolutions

Recall the result of the convolution of two  $\beta_{CR,IR}$  SCs (4.12). We need to think about where the latency would come into play when convoluting two  $\beta_{RL,IR}$  SCs instead. We have already established that we only need to shift the existing convolution result to the right by  $T$ . When calculating  $\beta_{RL11}$ , we have two latencies, as each of the curves we convolute already have a latency  $T_i$ , i.e.,  $\beta_{RL11}$  is shifted to the right by the sum  $T_{tot} = \sum_{i=1}^n T_i$  of the latencies of the two curves. For that, we add a new segment over the interval  $I = [0, T_{tot}]$ , which maps to 0. After that, we continue with the already known segments from the CR convolution, but we need to add a right-shift by  $T_{tot}$  to all time occurences in the segment formulas.  $\beta_{RL11}$  is then defined as

$$\beta_{RL11}(t) = \begin{cases} 0, & 0 \leq t \leq T_{tot}, \\ R(t - T_{tot}), & T_{tot} < t < T_{tot} + t_0, \\ b_2 + r_2(t - T_{tot}), & T_{tot} + t_0 \leq t \leq T_{tot} + r_2 \frac{b_1}{R^2}, \\ b_2 + r_2^2 \frac{b_1}{R^2} + R(t - T_{tot} - r_2 \frac{b_1}{R^2}), & T_{tot} + r_2 \frac{b_1}{R^2} < t < T_{tot} + r_2 \frac{b_1}{R^2} + t_0, \\ 2b_2 + r_2(t - T_{tot}), & T_{tot} + r_2 \frac{b_1}{R^2} + t_0 \leq t \leq T_{tot} + 2r_2 \frac{b_1}{R^2}, \\ 2b_2 + 2r_2^2 \frac{b_1}{R^2} + R(t - T_{tot} - 2r_2 \frac{b_1}{R^2}), & T_{tot} + 2r_2 \frac{b_1}{R^2} < t < T_{tot} + r_2 \frac{b_1}{R^2} + t_1, \\ b_1 + 2b_2 + 2r_2^2 \frac{b_1}{R^2} + \\ (r_1 + r_2)(t - T_{tot} - 2r_2 \frac{b_1}{R^2}) & \text{otherwise.} \end{cases} \quad (4.21)$$

As can be seen, the intervals of the cases are right-shifted by adding  $T_{tot}$  on top of the old interval boundaries, since they specify time instances. The same goes for any occurrence of  $t$  in the formulas. Here, we need to subtract  $T_{tot}$  to achieve a similar right-shift.

The general case is adjusted in the same manner. We add the combined latencies of all  $n$  servers at the start, then right-shift all remaining interval bounds and formulas by  $T_{tot}$ . We define the general case as follows:

- Up until  $T_{tot}$ , the convolution evaluates to 0. Then build the stairs. For  $m \leq n, m \in \mathbb{N}^+$ :
  - $(m-1) \cdot b_2 + (m-1) \cdot r_2^2 \frac{b_1}{R^2} + R(t - T_{tot} - (m-1) \cdot r_2 \frac{b_1}{R^2})$   
over the interval  
 $T_{tot} + (m-1) \cdot r_2 \frac{b_1}{R^2} < t < T_{tot} + (m-1) \cdot r_2 \frac{b_1}{R^2} + t_0,$
  - $m \cdot b_2 + r_2(t - T_{tot})$   
over the interval  
 $T_{tot} + (m-1) \cdot r_2 \frac{b_1}{R^2} + t_0 \leq t \leq T_{tot} + m \cdot r_2 \frac{b_1}{R^2}.$
- Afterwards, build the last two segments of the curve:
  - $n \cdot b_2 + n \cdot r_2^2 \frac{b_1}{R^2} + R(t - T_{tot} - n \cdot r_2 \frac{b_1}{R^2})$   
over the interval  
 $T_{tot} + n \cdot r_2 \frac{b_1}{R^2} < t < T_{tot} + (n-1) \cdot r_2 \frac{b_1}{R^2} + t_1,$
  - $b_1 + n \cdot b_2 + (n-1) \cdot r_2^2 \frac{b_1}{R^2} + (r_1 + r_2) \cdot (t - T_{tot} - n \cdot r_2 \frac{b_1}{R^2})$   
over the interval  
 $T_{tot} + (n-1) \cdot r_2 \frac{b_1}{R^2} + t_1 \leq t.$

### 4.5.3 Convolution Of RL Servers With Different Rates

#### Adjustment Of $\beta_{RL1,IR}$ Convolutions

We have seen that the convolution of  $\beta_{CR1,IR}$  curves has specific requirements as to when an additional stair is created. Once we look at  $\beta_{RL1,IR}$  curves, however, these requirements are discarded. Instead, we need to adhere to a certain shape of the resulting curve. To determine this shape, we calculate three consecutive convolutions, using point clouds.

Let there be four servers with  $R_1 = 7$  Mbit/s,  $R_2 = 10$  Mbit/s,  $R_3 = 15$  Mbit/s,  $R_4 = 25$  Mbit/s. Further, let  $\alpha_1$  be the foi, and  $\alpha_2$  the crossflow with  $b_1 = 2$  Mbit,  $b_2 = 0.016$  Mbit,  $r_1 = r_2 = 2$  Mbit/s. We set  $b_1 = 2$  Mbit, since the minimum required value for it is 1.78 Mbit, assuming  $R = 25$  Mbit/s.



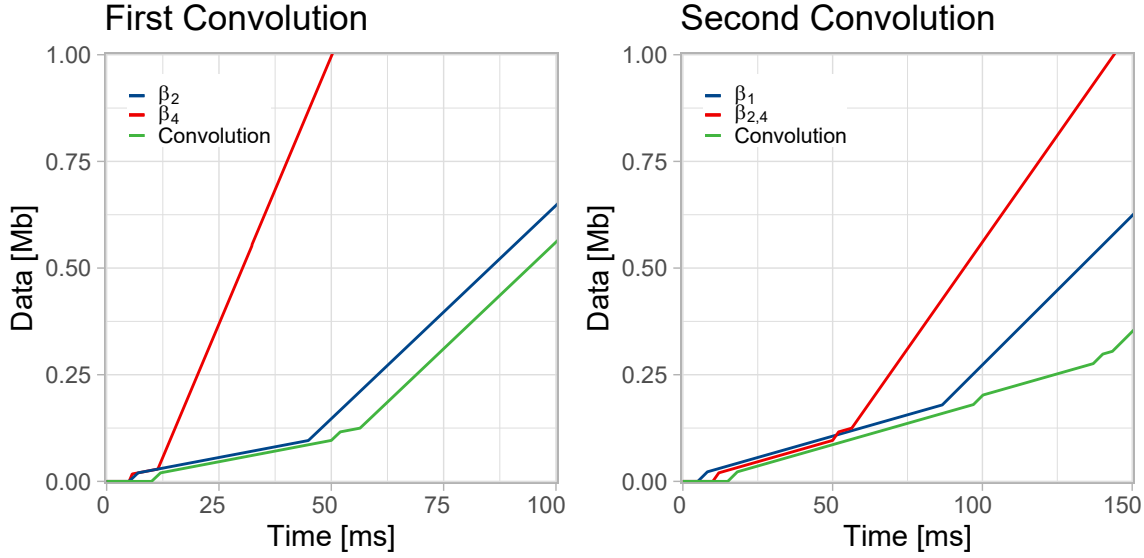


Figure 4.10: Stairs of the first and second convolution of RL SCs with different rates.

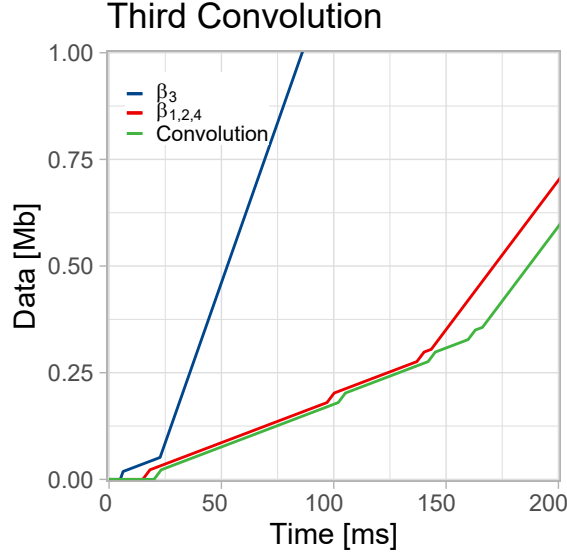
### Convolution One - $\beta_2 \otimes \beta_4$

If we recall the convolution for CR SCs, the convolution of  $\beta_2$  and  $\beta_4$  would be  $\beta_2$  again. This is different for RL servers, however, since we do not lose any stairs during the convolution. Instead, the stairs are ordered by size, starting from the longest stair. This holds for every additional convolution as we see in the following. The segments in which the rate equals the respective server rate also change. For each convolution, these segments change to the minimum server rate of any convoluted curve thus far, i.e., for the general case, the minimum out of all server rates is chosen.

The left side of Figure 4.10 shows the stairs of the first convolution of  $\beta_2$  and  $\beta_4$ . We can see that the sum of the latencies of both convolutions  $T_{tot}$  is prepended to the curve. We then build the stairs, whose steep parts take on the server rate of  $\beta_2$ , which is 10 Mbit/s. The flat parts stay the same. We also keep the stair lengths intact, i.e., the length of each stair is equal to the jump point of its respective SC. The last two segments of the curve are built in the same way we have seen with previous convolutions, where we shift the server curve to start at the end of the last stair, and end in the second intersection point  $t_1$ . Bear in mind that  $t_1$  is shifted further to the right for each additional convolution. The curve then changes into the aggregate AC.

To specify the first convolution, we first need to define a few variables:

- $R = \min\{R_2, R_4\} = R_2$  is the smaller server rate,
- $j = \max\left\{r_2 \frac{b_1}{R_2^2}, r_2 \frac{b_1}{R_4^2}\right\} = r_2 \frac{b_1}{R_2^2}$  is the larger jump point,
- $i = \min\left\{r_2 \frac{b_1}{R_2^2}, r_2 \frac{b_1}{R_4^2}\right\} = r_2 \frac{b_1}{R_4^2}$  is the smaller jump point,


 Figure 4.11: Stairs of the third convolution between  $\beta_{1,2,4}$  and  $\beta_3$ .

- $t_0 = \max \{t_0^2, t_0^4\}$  is the larger first intersection point,
- $t_1 = \max \{t_1^2, t_1^4\}$  is the larger second intersection point.

The first convolution is then defined as follows:

$$\beta_{2,4}(t) = \begin{cases} 0, & 0 \leq t \leq T_{tot}, \\ R(t - T_{tot}), & T_{tot} < t < T_{tot} + t_0, \\ b_2 + r_2(t - T_{tot}), & T_{tot} + t_0 \leq t \leq T_{tot} + j, \\ b_2 + r_2j + R(t - T_{tot} - j), & T_{tot} + j < t < T_{tot} + j + t_0, \\ 2b_2 + r_2(t - T_{tot}), & T_{tot} + j + t_0 \leq t \leq T_{tot} + j + i, \\ 2b_2 + r_2(i + j) + R(t - T_{tot} - i - j), & T_{tot} + j + i < t < T_{tot} + t_1 + i, \\ b_1 + 2b_2 + r_2 \cdot i + \\ (r_1 + r_2) \cdot (t - T_{tot} - i - j) & \text{otherwise.} \end{cases} \quad (4.22)$$

### Convolution Two - $\beta_{2,4} \otimes \beta_1$

As mentioned before, the stairs of the general case are ordered by their respective sizes, starting from the largest one. We can see this effect in the second convolution. Where we convoluted curves  $\beta_2$ ,  $\beta_4$  with the rates 10 Mbit/s and 25 Mbit/s in the first convolution, we now convolute  $\beta_1$  with the rate 7 Mbit/s on top of it. Looking at the singular curves, we notice that the stair of  $\beta_1$  is longer than the ones of  $\beta_2$  and  $\beta_4$ , since its rate is smaller. The right plot in Figure 4.10 shows what is happening once we convolute  $\beta_1$  with  $\beta_{2,4}$ . First, the steep parts of the curve on the rate of  $\beta_1$ . Then, the stair of  $\beta_1$  is pushed in front of the existing stairs of  $\beta_{2,4}$ .

To specify the second convolution, we need to extend  $i$ , which we have defined before. Instead of having a singular smallest jump point, we now have one smallest jump point for each convolution. We also sort these smallest jump points, where  $i_1$  is the largest of the smallest jump points, whereas  $i_x$  is the actual smallest jump point. With this, we can define the second convolution:

$$\beta_{1,2,4}(t) = \begin{cases} 0, & 0 \leq t \leq T_{tot}, \\ R(t - T_{tot}), & T_{tot} < t < T_{tot} + t_0, \\ b_2 + r_2(t - T_{tot}), & T_{tot} + t_0 \leq t \leq T_{tot} + j, \\ b_2 + r_2j + R(t - T_{tot} - j), & T_{tot} + j < t < T_{tot} + j + t_0, \\ 2b_2 + r_2(t - T_{tot}), & T_{tot} + j + t_0 \leq t \leq T_{tot} + j + i_1, \\ 2b_2 + r_2(j + i_1) + \\ R(t - T_{tot} - j - i_1), & T_{tot} + j + i_1 < t < T_{tot} + j + i_1 + t_0, \\ 3b_2 + r_2(t - T_{tot}), & T_{tot} + j + i_1 + t_0 \leq t \leq T_{tot} + j + i_1 + i_2, \\ 3b_2 + r_2(j + i_1 + i_2) + \\ R(t - T_{tot} - j - i_1 - i_2), & T_{tot} + j + i_1 + i_2 < t < T_{tot} + t_1 + i_1 + i_2, \\ b_1 + 3b_2 + r_2(i_1 + i_2) + \\ (r_1 + r_2) \cdot (t - T_{tot} - j - i_1 - i_2), & T_{tot} + t_1 + i_1 + i_2 \leq t. \end{cases} \quad (4.23)$$

### Convolution Three - $\beta_{1,2,4} \otimes \beta_3$

We briefly look at this convolution without formally defining it, as it does not give us any new insights. Rather, it confirms the general shape we have established in the previous two examples. In Figure 4.11, we see that the stair of  $\beta_3$  is pushed in between the stairs of  $\beta_2$  and  $\beta_4$  while the rate of the steep parts remains the same, as the minimum server rate has not changed.

### Definition Of The General Convolution Result

Looking at the example convolutions above allows us to define the general convolution result. For this, we first specify the different variables when not doing the convolutions step by step.

$R = \min\{R_i\}$  stays the same, but we instead determine the minimum rate over all  $n$  servers. The same goes for  $j$ , which is the largest jump point. The server with the smallest rate always has the largest jump point, since our flow constraints are constant throughout the convolution, i.e., the jump point only changes with different rates. Conversely, the server with the highest rate always has the smallest jump point. We add all smallest jump points to a list  $i$  where  $i_k$  refers to the  $k$ -th element in the list. This list is sorted in descending order from the largest to the smallest value. For the general case, all jump points except for the largest one  $j$  are added to  $i$ . Since there is no uniform jump point anymore, we have to specify the height of the steep parts of

the stairs differently. For this, let  $i[1 \dots k]$  be the sum of all entries of  $i$  from 1 to  $k$  and  $i_{tot}$  be the sum of all entries of  $i$ . In addition, let  $1_x$  be the indicator function, where the expression is 1 if it holds that  $k \geq x$  for a value  $k$ , and 0 else.

With this, we can define the general convolution result:

- Start the curve with the interval  $I = [0, T_{tot}]$ , which evaluates to 0 in every point.
- Then build the stairs. For  $m \leq n, m \in \mathbb{N}^+$ :
  - $(m - 1) \cdot b_2 + r_2 j \cdot 1_2 + r_2 \cdot i[1 \dots m] \cdot 1_3 + R(t - T_{tot} - j \cdot 1_2 - i[1 \dots m] \cdot 1_3)$   
over the interval  
 $T_{tot} + j \cdot 1_2 + i[1 \dots m - 1] \cdot 1_3 \leq t < T_{tot} + j \cdot 1_2 + i[1 \dots m] \cdot 1_3 + t_0$ ,
  - $m \cdot b_2 + r_2(t - T_{tot})$   
over the interval  
 $T_{tot} + j \cdot 1_2 + i[1 \dots m - 1] \cdot 1_3 + t_0 \leq t \leq T_{tot} + j \cdot 1_2 + i[1 \dots m] \cdot 1_3$ .
- Then build the last two segments:
  - $n \cdot b_2 + r_2(j + i_{tot}) + R(t - T_{tot} - j - i_{tot})$   
over the interval  
 $T_{tot} + j + i_{tot} < t < T_{tot} + i_{tot} + t_1$ ,
  - $b_1 + n \cdot b_2 + r_2 \cdot (i_{tot}) + (r_1 + r_2) \cdot (t - T_{tot} - j - i_{tot})$   
otherwise.

### Adjustment Of $\beta_{RL2,IR}$ Convolutions

Much like the changes for convolutions of  $\beta_{RL1,IR}/\beta_{RL2,IR}$  servers, we do the same for  $\beta_{RL2,IR}$  servers. Start the curve with an interval  $I = [0, T_{tot}]$  which equals 0 in every point. Then, append the convolution for  $\beta_{CR2,IR}$  curves which are right-shifted by  $T_{tot}$ .

$$\beta_{e2e}(t) = \bigotimes_{i=1}^n \beta_i(t) = \begin{cases} 0, & 0 \leq t \leq T_{tot}, \\ \min\{R_i\} \cdot (t - T_{tot}), & T_{tot} < t < t_2^*, \\ b_1 + b_2 + (r_1 + r_2) \cdot (t - T_{tot} - r_2 \frac{b_1}{R^2}) & \text{otherwise.} \end{cases} \quad (4.24)$$

## 4.6 Extension To Larger Flow Aggregates

So far we have only considered a flow aggregate consisting of two flows. This is a strong restriction, so the increase to an arbitrary amount of flows in the aggregate is desirable in order to model more complex networks. We do not define the general

case in this thesis, but increase the amount of flows to three. There are some things to consider which will help establishing the general case in the future.

Assume we have a flow aggregate consisting of three flows  $f_1, f_2, f_3$ . Each flow is constrained by a TB AC with respective bursts  $b_1, b_2, b_3$  and rates  $r_1, r_2, r_3$ . It does not matter which of the flows is the *foi*, as we have already described previously. We assume that an infinitesimal amount of the *foi*'s traffic is withheld until all other traffic has passed the server and is then sent. This guarantees that any delay bound we calculate is accurate for the *foi*.

We want to find the most pessimistic IR SC that is not too pessimistic at the same time. For this, we need to answer the question whether we consider all other flows in the aggregate when calculating the maximum burstiness increase of the current flow, even if we have already considered one or more of the other flows in previous segments of the curve. Does every flow experience its worst case sample path in the previous CBFS? We came to the conclusion that assuming every flow experiences its worst case sample path is too pessimistic, as in realistic situations, only one flow is actually the one that is delayed by the maximum possible time. The other flows of the aggregate are served faster, i.e., their burstiness increase is smaller. With this, we can specify the general form of the IR SC for three flows.

Let  $f_1$  be the flow with the smallest burst  $b_1$  that experiences its worst case sample path in the previous CBFS. Hence, we start the IR SC with a jump to  $b_1$ , then follow  $f_1$ 's AC  $b_1 + r_1 t$ .

**Case 1:**  $b_1 \leq b_2, b_3$  and  $r_1 \leq r_2, r_3$

No other flow intersects  $f_1$  until its burstiness increase has been negated, which is the case at  $t = r_1 \frac{b_2+b_3}{R^2}$

$$\Rightarrow \left\{ b_1 + r_1 t, \quad 0 \leq t \leq r_1 \frac{b_2+b_3}{R^2} \right\}.$$

**Case 2:**  $b_1 < b_2, b_3$  and  $r_1 > r_2, r_3$

We first calculate the intersection points of  $f_1$  with  $f_2$  and  $f_3$ . We do this to determine whether one of the other two curves falls below the curve of  $f_1$ , as its rate is larger than the rates of the other two flows. If both of the intersections lie before  $f_1$ 's jump point  $t_1$ , we choose the smaller one. If only one of the points lies before  $t_1$ , we choose this one. If none of both intersection points lie before  $t_1$ , we go back to case 1. We assume that the intersection point of  $f_1$  and  $f_2$  lies before  $t_1$  and is the smaller point.

We construct the segment by first following the curve of  $f_1$  until we reach the intersection point  $x$ , then swap to the curve of  $f_2$ . We know that the segment end is at  $t_1$ , but  $f_2$  takes until  $t_2 = r_2 \frac{b_1+b_3}{R^2}$  to negate its burstiness increase. An interesting question is what relation these two points in time have. Since we assume that  $r_1 > r_2$  and  $b_1 < b_2$ ,  $t_1$  is always larger than  $t_2$ . This means that we need to change to another curve in the interval  $I = [t_1, t_2]$ . There are two choices on what to change to. Either  $b_1$  is smaller than the jump from  $b_2 + r_2 t_2$  to  $b_1 + r_1 t_2$ . In this case, we jump to  $b_1 + b_2 + r_2 t_2$ . In the

other case, we jump to the initial curve, which is  $b_1 + r_1 t_2$ , that we have followed in this interval. We could also check whether  $b_3$  is smaller than  $b_1 + r_1 t_2$ , but since we assume that  $b_1 < b_3$ , if this holds for  $b_3$  it also holds for  $b_1$ , with the latter producing the smaller service curve. After this jump, we continue with the curve of  $f_1$ , shifted to the right by  $t_2$  and to the top by  $b_1 + b_2 + r_2 t_2$ .

Next, we calculate the maximum value for  $b_1$  to fall into case 1:

$$\begin{aligned}
 & b_1 + r_1 t_2 - b_2 - r_2 t_2 = 0 \\
 \Leftrightarrow & \quad b_1 - b_2 + t_2(r_1 - r_2) = 0 \\
 \Leftrightarrow & \quad b_1 - b_2 + r_1 \frac{b_2 + b_3}{R^2} (r_1 - r_2) = 0 \\
 \Leftrightarrow & \quad b_1 = b_2 - r_1 \frac{b_2 + b_3}{R^2} (r_1 - r_2).
 \end{aligned}$$

We would need to derive parameter requirements as we did for the two-flow aggregate earlier in this chapter to comment about the feasibility of this threshold. However, we leave that for future work.

The resulting segment definition is then

$$\begin{cases} b_1 + r_1 t, & 0 \leq t \leq x, \\ b_2 + r_2 t, & x < t \leq t_2, \\ \min\{b_1 + r_1 t, b_1 + b_2 + r_2 t_2 + r_1(t - t_2)\}, & t_2 < t \leq t_1. \end{cases}$$

These two cases are applied to the segments one and two. In segment two  $f_1$  is no longer considered. In segment three, only  $f_3$  is left, which jumps up to  $b_{agg} = b_1 + b_2 + b_3$ , then continues with the aggregate AC.

We briefly consider the fact that the aggregate AC does not experience a burstiness increase, but its flows do. We hence need to check whether we do not violate the maximum burst  $b_{agg}$ . Since we know that  $f_1$  experiences its maximum burstiness increase,  $b_1$  is increased to  $b_1^{bi} = b_1 + r_1 \frac{b_2 + b_3}{R}$ , which in return reduces the remaining burst to  $b_{agg} - b_1^{bi}$ . When  $f_2$  is served, we are left with a burst of  $b_3^{lo} = b_3 - r_1 \frac{b_2 + b_3}{R} - r_2 \frac{b_3}{R}$  for  $f_3$ . We have found examples where  $b_{agg}$  is already exhausted before the third flow had its turn. To prevent this from happening, we need to guarantee that the flows are ordered and served by burst sizes, starting from the smallest burst. Our specification of the IR SC guarantees this as we always pick the flow with the current smallest burst for the next segment.

# 5 Analysis Of The Proposed Min-Plus CBFS-IR Service Curve

## 5.1 Interpretation Of The Convolution Cases

Before we analyze the curves we found in the previous chapter, we briefly talk about their meaning in the context of the IR. Case 1 is quite straightforward. One of the flows has a high enough burstiness increase, i.e., it loses its shape and does not regain it before it enters the following IR, at a CBFS. This requires the IR to reshape the flow in question until it adheres to its constraints again. We have only looked at the case where the foi has a very high burst. This stems from our initial assumption that the foi always has to be served last. We have revised this assumption when we considered class aggregates with more than two flows., i.e., the expected result should be that either the foi or the crossflow can have very large bursts in comparison to the respective other flow. If it is the crossflow, we start the SC with negating the fois burstiness increase. This modeling is however not discussed further in this thesis, but instead left for future work.

Case 2 is not that simple any more, despite the resulting curves having fewer cases. We have seen in depth that the IR will hold back any flows that do not adhere to their constraints. We still have a burstiness increase when we are in case 2, it is just not nearly as big as in case 1. It is in fact so small that the IRs effect of the reshaping offers a faster SC than if we let the CBFS transmit the flows without reshaping, i.e., the CBFS does the reshaping itself without the help of the IR.

## 5.2 Delay Bounds Of The Different CBFS-IR Service Curves

Since we have built the results of this thesis on top of the bound from Mohammadpour et al. [10], we now want to compare its results to our proposed service curve. For this, we need to find formulas for calculating the delay bound of either CR or RL servers. However, this is not as straightforward as calculating the delay bound for normal RL servers without IR elements. We find formulas for the delay of CR servers with identical rates in the following and use this example to show how complex seemingly simple examples can become.

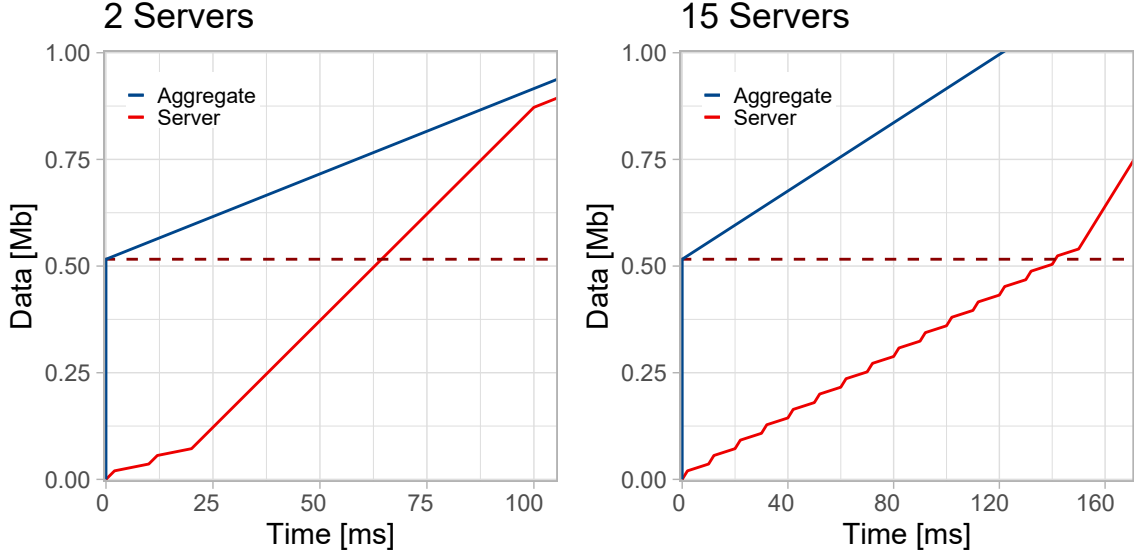


Figure 5.1: Two out of three possible segments that can be selected for the delay bound.

### 5.2.1 Delay Bound Of CR Servers With Identical Rates

Calculating the delay bound of normal CR servers is very straightforward. Once we add the IR element to it things become complicated, however. As we convolute  $n$  servers, we also have  $n$  stairs in the final curve, which we use to calculate the delay bound. We therefore have three different formulas for the delay bound, using a different one depending on the network size. The larger the network size in comparison to the aggregate burst, the higher the chance of the maximum horizontal deviation hitting a stair. We start with the simplest of cases, in which the second to last segment is hit.

Here, we equal the aggregate burst  $b_1 + b_2$  to the second to last segment, then solve it for  $t$ . We choose  $b_{agg} = b_1 + b_2$ , as the part of the CBFS-IR SC that is on the same horizontal line as  $b_{agg}$  produces the largest horizontal deviation between the two lines.  $t$  is then

$$\begin{aligned}
 b_1 + b_2 &= n \cdot b_2 + n \cdot r_2^2 \frac{b_1}{R^2} + R \cdot \left( t - n \cdot r_2 \frac{b_1}{R^2} \right) \\
 \Leftrightarrow Rt &= b_1 - (n - 1) \cdot b_2 - n \cdot r_2^2 \frac{b_1}{R^2} + n \cdot R \cdot r_2 \frac{b_1}{R^2} \\
 \Leftrightarrow t &= \frac{b_1 - (n - 1) \cdot b_2 + n \cdot r_2 \frac{b_1}{R^2} (R - r_2)}{R}.
 \end{aligned}$$

We see this time  $t$  on the left side of Figure 5.1. It is the dashed horizontal line that intersects the CBFS-IR curve at its value that is equal to  $b_{agg}$ .



As can be seen on the right in Figure 5.1, we can also reach this maximum horizontal deviation at the steep or flat part of the stairs if the network consists of sufficiently many servers. We also calculate  $t$  for these two cases. To do this, we need to know which stair we are hitting and whether it is the steep or flat part. We can calculate the stair that is hit by dividing  $b_{agg}$  by the stair height. This tells us the full stairs we pass plus the percentage of the next stair we do not pass. To determine whether we stop at the steep or flat part, we calculate at which percentage of the stair height we change from steep to flat.

To demonstrate this, let the flow aggregate consist of the flow  $f_1$  and the crossflow  $f_2$  with  $b_1 = 0.5$  Mbit,  $b_2 = 0.016$  Mbit,  $r_1 = r_2 = 2$  Mbit/s. Further, assume we have 15 servers in the network, which are convoluted to an end-to-end SC  $\beta_{e2e}$  with  $R = 10$  Mbit/s. The shape of the stairs of  $\beta_{e2e}$  can be seen in the right plot of 5.1. As can be seen, we hit the steep part of the 15th stair for the delay bound. This can be confirmed as described above by calculating  $n_{stair} = \frac{0.516Mb}{0.036Mb} = 14.33$ . This means that we pass 14 complete stairs, then hit the next stair at 33% of its height. Calculating the turning point from steep to flat yields  $tp = \frac{0.02 \text{ Mbit}}{0.036 \text{ Mbit}} = 0.55$ , as we know that the height of the stair is 0.036 Mbit and the steep part has a height of 0.02 Mb. This means that until we reach 14.56 when calculating  $n_{stair}$ , we hit the steep part of the 15th stair. This conforms to the graphical representation.

We now proceed to derive the two remaining bounds, starting with the steep part:

$$\begin{aligned}
 b_1 + b_2 &= \lfloor n_{stair} \rfloor \cdot b_2 + \lfloor n_{stair} \rfloor \cdot r_2^2 \frac{b_1}{R^2} + R \left( t - \lfloor n_{stair} \rfloor \cdot r_2 \frac{b_1}{R^2} \right) \\
 \Leftrightarrow \quad Rt &= b_1 - (\lfloor n_{stair} \rfloor - 1) \cdot b_2 - \lfloor n_{stair} \rfloor \cdot r_2^2 \frac{b_1}{R^2} + \lfloor n_{stair} \rfloor \cdot R \cdot r_2 \frac{b_1}{R^2} \\
 \Leftrightarrow \quad t &= \frac{b_1 - (\lfloor n_{stair} \rfloor - 1) \cdot b_2 + \lfloor n_{stair} \rfloor \cdot r_2 \frac{b_1}{R^2} (R - r_2)}{R}, \\
 \\ 
 b_1 + b_2 &= \lceil n_{stair} \rceil \cdot b_2 + r_2 t \\
 \Leftrightarrow \quad r_2 t &= b_1 - \lfloor n_{stair} \rfloor \cdot b_2 \\
 \Leftrightarrow \quad t &= \frac{b_1 - \lfloor n_{stair} \rfloor \cdot b_2}{r_2}
 \end{aligned}$$

As can be seen, the delay bound for hitting the steep part of the stairs is quite similar to the delay bound we have seen before. This is logical, as they have the same slope and rate and only differ in their shifts.

The identification of where the maximum horizontal deviation for CR servers with identical rates is reached is complicated, but manageable. However, once we drop the assumption that all server rates are equal, we run into further problems. This causes the stair heights to not be consistent anymore, i.e., we cannot determine which stair is hit without fully specifying the CBFS-IR SC and then subtracting the respective

stair heights from  $b_{agg}$ . Especially for large networks this is not an efficient way to calculate the delay bound. To combat the situation, we next look at approximations for the stair part of the SCs and how big of an impact they have on the delay bound.

## 5.3 Simplification Of Analyses

There are two general ways of approximating the CBFS-IR curve. We either take the second to last segment and extend it to intersect the x axis and let the SC be equal to 0 from  $t = 0$  to the intersection point  $t_{int}^s$ . We remove all stairs and transform the curve into a RL SC, which is convoluted with the aggregate AC. The second approach is to lower bound the stairs.

### 5.3.1 Approach 1 - RL-Like Approximation

For this approach, we first find the intersection point of the second to last segment of the original curve and the x axis. Start with the assumption that we have CR servers and identical rates:

$$\begin{aligned}
 n \cdot b_2 + n \cdot r_2^2 \frac{b_1}{R^2} + R \cdot \left( t_{int}^s - n \cdot r_2 \frac{b_1}{R^2} \right) &= 0 \\
 \Leftrightarrow R \cdot t_{int}^s &= n \cdot R \cdot r_2 \frac{b_1}{R^2} - n \cdot b_2 - n \cdot r_2^2 \frac{b_1}{R^2} \\
 \Leftrightarrow R \cdot t_{int}^s &= n \cdot \left( R \cdot r_2 \frac{b_1}{R^2} - b_2 - r_2^2 \frac{b_1}{R^2} \right) \\
 \Leftrightarrow t_{int}^s &= \frac{n \cdot \left( r_2 \frac{b_1}{R^2} (R - r_2) - b_2 \right)}{R}.
 \end{aligned}$$

Next, assume CR servers with different rates:

$$\begin{aligned}
 n \cdot b_2 + r_2 (i_{tot} + j) + R \cdot \left( t_{int}^d - (i_{tot} + j) \right) &= 0 \\
 \Leftrightarrow R \cdot t_{int}^d &= R \cdot (i_{tot} + j) - n \cdot b_2 - r_2 \cdot (i_{tot} + j) \\
 \Leftrightarrow t_{int}^d &= \frac{(i_{tot} + j) \cdot (R - r_2) - n \cdot b_2}{R}
 \end{aligned}$$

We then determine the interval over which the curve evaluates to 0. For identical rates, the interval is  $I^s = [0, t_{int}^s]$ , for different rates  $I^d = [0, t_{int}^d]$ . The approximation

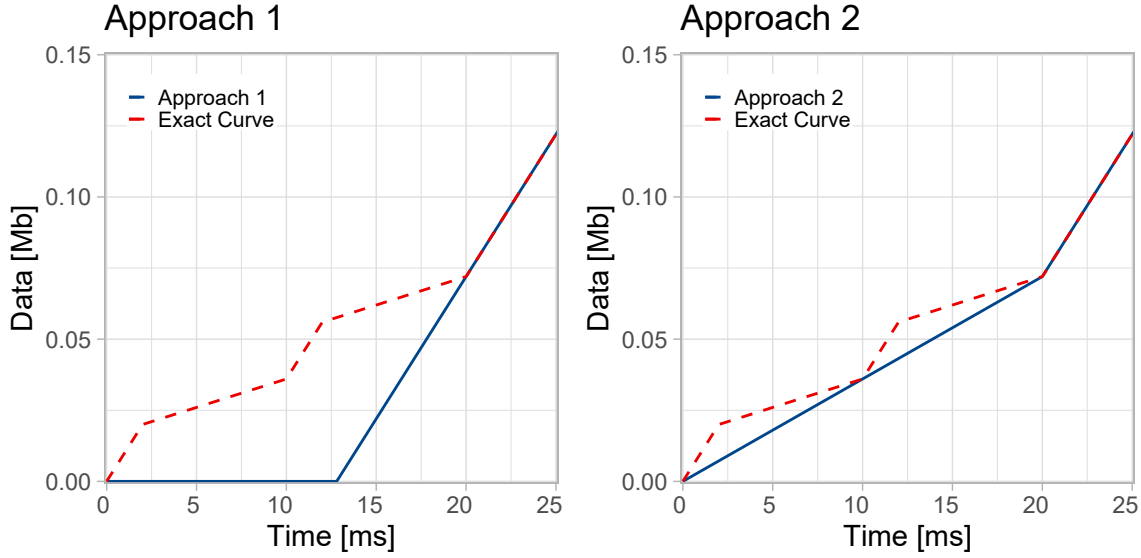


Figure 5.2: Comparison of approaches 1 and 2 with the exact curve, assuming CR servers and identical rates.

for identical and different rates are defined as follows:

$$\beta_{e2e,s}^{appr1}(t) = \begin{cases} 0, & 0 \leq t \leq t_{int}^s, \\ n \cdot b_2 + n \cdot r_2^2 \frac{b_1}{R^2} + R \cdot \left( t - n \cdot r_2 \frac{b_1}{R^2} \right), & t_{int}^s < t < (n-1) \cdot r_2 \frac{b_1}{R^2} + t_1, \\ b_1 + n \cdot b_2 + n \cdot r_2^2 \frac{b_1}{R^2} + (r_1 + r_2) \cdot \left( t - n \cdot r_2 \frac{b_1}{R^2} \right) & \text{otherwise.} \end{cases} \quad (5.1)$$

$$\beta_{e2e,d}^{appr1}(t) = \begin{cases} 0, & 0 \leq t \leq t_{int}^d, \\ n \cdot b_2 + r_2 (i_{tot} + j) + R \cdot (t - (i_{tot} + j)), & t_{int}^d < t < i_{tot} + t_1, \\ b_1 + n \cdot b_2 + r_2 (i_{tot}) + (r_1 + r_2) \cdot (t - (i_{tot} + j)) & \text{otherwise.} \end{cases} \quad (5.2)$$

We can adjust these two curves to apply to RL servers much like we did in chapter 5, i.e., we add an additional right-shift by  $T_{tot}$  to all occurrences of  $t$  and shift the respective intervals accordingly by adding  $T_{tot}$ .

Comparing this to the specified general service curve in subsection 4.4.4 makes it very obvious that this approximation is easier to handle. We can see the comparison of this approximation and the exact curve in Figures 5.2 and 5.3 on the left.

Of interest is now how much worse the results of this approximation. Generally, the deviation of the two curves increases depending on the size of the aggregate burst

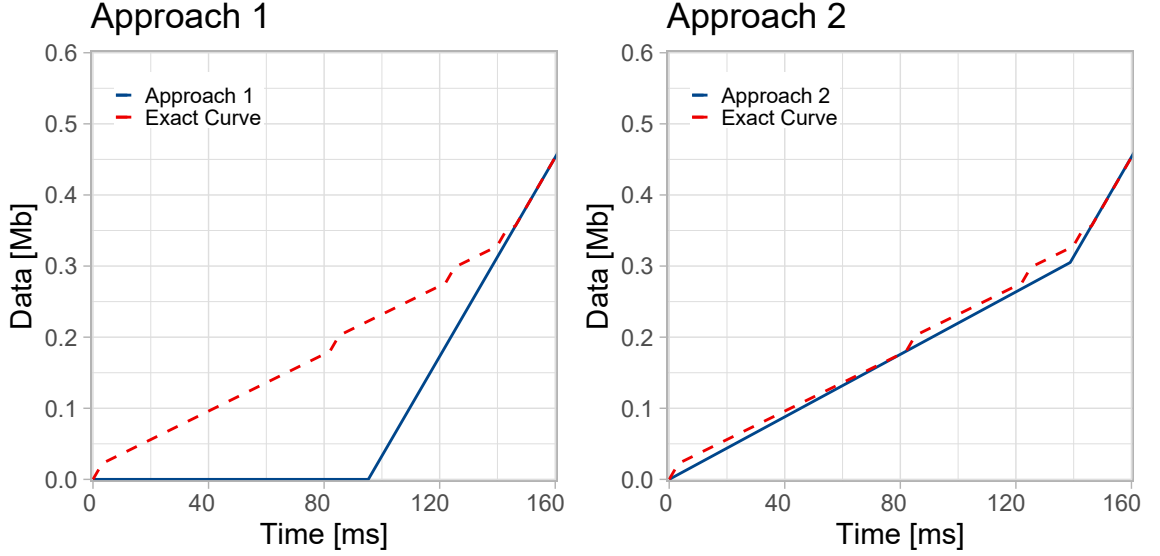


Figure 5.3: Comparison of approaches 1 and 2 with the exact curve, assuming CR servers and different rates.

and the number of servers in the network. The more servers we have, the higher the chance of the delay bound being measured at the approximation of the stairs. Similarly, if the burst is very small, the same thing happens. However, we can upper bound the deviation by the interval end at the start of both curves. At worst, we measure the delay bound for  $b_{agg} = 0$  Mbit, where the deviation is then equal to  $t_{int}^s$  or  $t_{int}^d$ . Realistically, we do not have a burst that is close to 0 Mbit, so this upper bound is very loose. We can calculate the exact difference between the steep and flat parts of the stairs and the approximation, however, we concluded that this knowledge of the exact formal difference is of no help. Rather, the knowledge of what worsens the approximation bound, as we have stated above, is more important.

### 5.3.2 Approach 2 - Stair Approximation

As a second approach we lower bound the stairs. For this, we calculate the required slope of the segment. We can choose any stair for calculating the slope when assuming identical rates. For different rates, we calculate the slope of the longest stair, i.e., the first stair. The formula for calculating the slope is

$$m = \frac{y_2 - y_1}{x_2 - x_1}, \quad (5.3)$$

where  $P_1 = (x_1, y_1) = (0, 0)$  is the origin, and  $P_2 = \left(r_2 \frac{b_1}{R^2}, b_2 + r_2^2 \frac{b_1}{R^2}\right)$  is the end point of the first stair. These points are the same for identical and different rates. The

resulting slope is then

$$m = \frac{b_2 + r_2^2 \frac{b_1}{R^2}}{r_2 \frac{b_1}{R^2}}. \quad (5.4)$$

When assuming identical rates, we can add a new segment, spanning from  $t = 0$  to  $n \cdot r_2 \frac{b_1}{R^2}$  with the above calculated slope  $m$ , that replaces the stairs, then transition into the two last segments of the exact curve. For different rates, we do not reach the same end point of the last stair as the slope is too flat for that. Instead, we extend the second to last segment much like we did in approach 1. The only difference is that we lengthen this segment to intersect the new first segment, not the x axis:

$$\begin{aligned} m \cdot t_{int}^2 &= n \cdot b_2 + r_2 \cdot (i_{tot} + j) + R \cdot (t_{int}^2 - (i_{tot} + j)) \\ \Leftrightarrow t \cdot (R - m) &= R \cdot (i_{tot} + j) - n \cdot b_2 - r_2 \cdot (i_{tot} + j) \\ \Leftrightarrow t_{int}^2 &= \frac{(i_{tot} + j) \cdot (R - r_2) - n \cdot b_2}{R - m}. \end{aligned}$$

With this, we can define the approximations:

$$\beta_{e2e,s}^{appr2}(t) = \begin{cases} \frac{b_2 + r_2^2 \frac{b_1}{R^2}}{r_2 \frac{b_1}{R^2}} \cdot t, & 0 \leq t \leq n \cdot r_2 \frac{b_1}{R^2}, \\ n \cdot b_2 + n \cdot r_2^2 \frac{b_1}{R^2} + R \cdot \left(t - n \cdot r_2 \frac{b_1}{R^2}\right), & n \cdot r_2 \frac{b_1}{R^2} < t < (n-1) \cdot r_2 \frac{b_1}{R^2} + t_1, \\ b_1 + n \cdot b_2 + n \cdot r_2^2 \frac{b_1}{R^2} + \\ (r_1 + r_2) \cdot \left(t - n \cdot r_2 \frac{b_1}{R^2}\right) & \text{otherwise.} \end{cases} \quad (5.5)$$

$$\beta_{e2e,d}^{appr2}(t) = \begin{cases} \frac{b_2 + r_2^2 \frac{b_1}{R^2}}{r_2 \frac{b_1}{R^2}} \cdot t, & 0 \leq t \leq t_{int}^2, \\ n \cdot b_2 + r_2 (i_{tot} + j) + R \cdot (t - (i_{tot} + j)), & t_{int}^2 < t < i_{tot} + t_1, \\ b_1 + n \cdot b_2 + r_2 (i_{tot}) + \\ (r_1 + r_2) \cdot (t - (i_{tot} + j)) & \text{otherwise.} \end{cases} \quad (5.6)$$

As already explained in approximation 1, we can also adjust these curves for RL servers. This is done by adding an additional right-shift by  $T_{tot}$  to all occurrences of  $t$  and shifting the intervals to the right by the same value.

Similar to approximation 1, we can determine an upper bound on the difference between the delay bound of the approximation and exact curve. We do this by calculating the maximum distance between the two curves, which is reached at the intersection point  $t_0$ . For identical rates, where all stairs have the same maximum horizontal distance to the approximation, it does not matter where we derive the maximum

distance. However, for different rates, it does make a difference. We can observe that the distance between the exact curve and the approximation becomes larger the further along the stairs we are. This makes sense, since the approximation assumes the slope of the longest stair, resulting in a larger maximum horizontal distance for shorter stairs. However, we do not use the last stair to calculate the distance as one would suspect. Instead, we use the second to last stair. The reasoning for this is that the shifted  $t_0$  of the last stair does not hit the approximation, but the second to last segment of the curve, which is identical for the exact curve and the approximation. It follows that its distance is smaller than the horizontal distance of the second to last stair, which hits the approximation.

First, we determine the maximum distance for identical rates:

$$\begin{aligned} b_2 + r_2 \cdot t_0 &= \frac{b_2 + r_2^2 \frac{b_1}{R^2}}{r_2 \frac{b_1}{R^2}} \cdot t \\ \Leftrightarrow \left( b_2 + r_2^2 \frac{b_1}{R^2} \right) \cdot t &= r_2 \frac{b_1}{R^2} \cdot (b_2 + r_2 \cdot t_0) \\ \Leftrightarrow t &= \frac{r_2 \frac{b_1}{R^2} \cdot (b_2 + r_2 \cdot t_0)}{b_2 + r_2^2 \frac{b_1}{R^2}}. \end{aligned}$$

Then, we determine the maximum distance for different rates:

$$\begin{aligned} (n-1) \cdot b_2 + r_2 \cdot (j + i[1 \dots n-2] + t_0) &= \frac{b_2 + r_2^2 \frac{b_1}{R^2}}{r_2 \frac{b_1}{R^2}} \cdot t \\ \Leftrightarrow \left( b_2 + r_2^2 \frac{b_1}{R^2} \right) \cdot t &= r_2 \frac{b_1}{R^2} \cdot ((n-1) \cdot b_2 + r_2 \cdot (j + i[1 \dots n-2] + t_0)) \\ \Leftrightarrow t &= \frac{r_2 \frac{b_1}{R^2} \cdot ((n-1) \cdot b_2 + r_2 \cdot (j + i[1 \dots n-2] + t_0))}{b_2 + r_2^2 \frac{b_1}{R^2}}. \end{aligned}$$

With the definitions done, we can now determine the delay bounds of the different approximations.

### 5.3.3 Delay Bounds Of CR And RL Servers Using Approximations 1 And 2

As we have seen before, using the exact curve requires us to calculate which part of which stair of the curve is hit for identical server rates. For different rates, we did not specify any bounds for the exact curve, as calculating which part of which stair is hit is not easily done. Using approximations, we can skip the calculation of which stair is hit entirely. Instead, for approximation 1, we can calculate the delay bound

without any additional checks. For approximation 2, we need to determine whether  $b_{agg}$  is smaller or larger than the approximation at the last point of the (former) stair segment.

### Delay Bounds For Approximation 1

We can skip the calculation of this bound for identical rates, as it is equal to the delay bound we get when assuming the second to last part of the exact curve is hit. We can adjust this bound for RL servers by adding  $T_{tot}$  on top of the bound for CR servers.

For different rates, we calculate the bound as follows:

$$\begin{aligned} b_1 + b_2 &= n \cdot b_2 + r_2 \cdot (i_{tot} + j) + R \cdot (t_{CR,d}^{appr1} - (i_{tot} + j)) \\ \Leftrightarrow R \cdot t_{CR,d}^{appr1} &= b_1 - (n - 1) \cdot b_2 - r_2 \cdot (i_{tot} + j) + R \cdot (i_{tot} + j) \\ \Leftrightarrow t_{CR,d}^{appr1} &= \frac{b_1 - (n - 1) \cdot b_2 + (i_{tot} + j) \cdot (R - r_2)}{R}. \end{aligned}$$

The delay bound for RL servers is then

$$t_{RL,d}^{appr1} = t_{CR,d}^{appr1} + T_{tot}.$$

### Delay Bounds For Approximation 2

Here, we need to check whether  $b_{agg}$  is larger or smaller than the value of the approximation at the end of the stair segment. For identical rates, the condition is

$$b_{agg} \geq \beta_{e2e,s}^{appr2} \left( n \cdot r_2 \frac{b_1}{R^2} \right).$$

For different rates, it is

$$b_{agg} \geq \beta_{e2e,d}^{appr2} \left( \sum_{i=1}^n r_2 \frac{b_1}{R_i^2} \right).$$

If this condition does not hold, we default to the bounds we have calculated for approximation 1. If it does hold, the delay bound for identical and different server rates is identical and specified as

$$\begin{aligned} b_1 + b_2 &= \frac{b_2 + r_2^2 \frac{b_1}{R^2}}{r_2 \frac{b_1}{R^2}} \cdot t_{CR,sd1}^{appr2} \\ \Leftrightarrow \left( b_2 + r_2^2 \frac{b_1}{R^2} \right) \cdot t_{CR,sd1}^{appr2} &= r_2 \frac{b_1}{R^2} (b_1 + b_2) \\ \Leftrightarrow t_{CR,sd1}^{appr2} &= \frac{r_2 \frac{b_1}{R^2} (b_1 + b_2)}{b_2 + r_2^2 \frac{b_1}{R^2}}. \end{aligned}$$

The bound is equal for both cases, as we use the same slope definition for approximation 2. We adjust this bound for RL servers as follows:

$$t_{RL,sd1}^{appr2} = t_{CR,s1}^{appr2} + T_{tot}.$$

### 5.3.4 Miscellaneous Bounds

Here, we specify some general bounds that are used in the next section. We start with the delay bound of a single CBFS-IR server assuming case 1:

$$\begin{aligned} b_1 + b_2 &= b_2 + r_2^2 \frac{b_1}{R^2} + R \cdot \left( t - r_2 \frac{b_1}{R^2} \right) \\ \Leftrightarrow \quad R t &= b_1 + r_2 \frac{b_1}{R^2} \cdot (R - r_2) \\ \Leftrightarrow \quad t &= \frac{b_1 + r_2 \frac{b_1}{R^2} (R - r_2)}{R}. \end{aligned}$$

Next, we do the same for case 2:

$$\begin{aligned} b_1 + b_2 &= R t \\ \Leftrightarrow \quad t &= \frac{b_1 + b_2}{R}. \end{aligned}$$

With all delay bounds specified, we can now compare them to the paper bound from [10].

## 5.4 Comparison To The Paper Bound

First, we specify the parameter values, then look at different networks. For these networks, we calculate the paper [10] as well as the min-plus CBFS-IR delay bounds, and compare them.

For the first part of the analysis the flows and servers are modeled as follows:

- Let  $f_1$  be the foi,  $f_2$  the crossflow with the parameters  $r_i = 2$  Mbit/s,  $b_1 = 0.5$  Mbit,  $b_2 = 0.016$  Mbit,  $M_{f_1} = M_{f_2} = 2$  Kbit.
- Let the CBS be modeled with the parameters  $I_{ij}^A = 50$  Mbit/s,  $S_{ij}^A = -50$  Mbit/s.
- The line rate between all server pairs equals  $c_{ij} = 50$  Mbit/s.
- The LOSC at each server is then modeled with the parameters  $T_{ij}^A = 0$ s,  $R_{ij}^A = 25$  Mbit/s.

We also assume a similar network structure to that seen in Figure 3.3, i.e., the specified number of servers is in a tandem through which the foi and crossflow are being sent.



Bound Type	1 Server	20 Servers	Bound Type	1 Server	20 Servers
Paper	20.6ms	412ms	Paper	8ms	160ms
Min-Plus Case 1	21.4ms	428ms	Min-Plus Case 2	8ms	160ms

Table 5.1: Overview of additive bounds for different numbers of servers.

### 5.4.1 Using Additive Bounds

First, we are interested in a comparison of single servers, i.e., we do not use the server concatenation (Theorem 4.5), resulting in our bound being additive as well. We do this to confirm that, for additive bounds, the two approaches are identical. Once we start using the concatenation of servers, our proposed bound will certainly leave us with smaller bounds. As can be seen on the left side in Table 5.1, the paper bound is slightly better in the additive setting. Our min-plus SC produces a bound that is 3,8% worse for the same network. This is a reasonably close result, which reaffirms us that the SC is neither too optimistic nor pessimistic.

We next set  $b_1 = b_2 = 0.1$  Mbit, leaving all other parameters identical, then compare the additive bounds of the paper bound and the min-plus SC for case 2. The results can be seen on the right side in Table 5.1. These two bounds are indeed identical, further convincing us that the modeling approach we took is accurate.

We only looked at CR servers here, but, as we have seen in the previous chapters, the transition to RL servers is done by adding the sum of latencies of all servers to the delay bound. The same holds for the paper bound, i.e., the relation between the two results stays the same.

### 5.4.2 Using The Server Concatenation

Next, we compare the bounds which use the server concatenation with the paper bound. We will use approximations 1 and 2 instead of the exact bound, as discussed previously. We focus on case 1, while only doing a small case 2 comparison.

We calculate the delay bounds for 5 and 30 servers, using the previously specified parameters, as well as three additional parameter combinations for flows  $f_1$  and  $f_2$ :

C2:  $b_1 = 0.22$  Mbit,

C3:  $r_i = 12.5$  Mbit/s,  $b_1 = 0.07$  Mbit,

C4:  $r_i = 12.5$  Mbit/s,  $b_1 = 2$  Mbit.

The remaining parameters are identical to the specification above.

First, we need to check whether approximation 2 or the second to last segment is used for calculating the delay bound for case 1. For 5 servers, we obtain the following results:

Bound Type	Comb.	5 Servers	Ratio <sub>5</sub>	30 Servers	Ratio <sub>30</sub>
Paper	C1	103ms	-	618ms	-
	C2	47ms		282ms	
	C3	17ms		102ms	
	C4	403ms		2418ms	
Approximation 1	C1	24.8ms	24.08%	45.6ms	7.38%
	C2	9.5ms	20.21%	9.7ms	3.44 %
	C3	3.7ms	21.76%	5.2ms	5.1%
	C4	177.4ms	44.13%	661ms	27.34%
Approximation 2	C1	24.8ms	24.08%	43ms	6.96%
	C2	9.5ms	20.21%	9.5ms	3.37%
	C3	3.6s	21.18%	3.6ms	3.53%
	C4	156ms	38.71%	156ms	6.45%

Table 5.2: Comparison of the approximation delay bounds and paper bound for the different parameter combinations C1-C4.

C1:  $\beta_{e2e,s}^{appr2}(0.008s) = 0.096 \text{ Mbit} \leq b_{agg} = 0.516 \text{ Mbit} \Rightarrow$  second to last segment,

C2:  $\beta_{e2e,s}^{appr2}(0.0035s) = 0.087 \text{ Mbit} \leq b_{agg} = 0.236 \text{ Mbit} \Rightarrow$  second to last segment,

C3:  $\beta_{e2e,s}^{appr2}(0.0067s) = 0.161 \text{ Mbit} \geq b_{agg} = 0.086 \text{ Mbit} \Rightarrow$  approximation segment,

C4:  $\beta_{e2e,s}^{appr2}(0.192s) = 2.384 \text{ Mbit} \geq b_{agg} = 2.016 \text{ Mbit} \Rightarrow$  approximation segment.

For 30 servers, the following segments are hit:

C1:  $\beta_{e2e,s}^{appr2}(0.048s) = 0.576 \text{ Mbit} \geq 0.516 \text{ Mbit} \Rightarrow$  approximation segment,

C2:  $\beta_{e2e,s}^{appr2}(0.021s) = 0.522 \text{ Mbit} \geq 0.236 \text{ Mbit} \Rightarrow$  approximation segment,

C3:  $\beta_{e2e,s}^{appr2}(0.04s) = 0.964 \text{ Mbit} \geq 0.086 \text{ Mbit} \Rightarrow$  approximation segment,

C4:  $\beta_{e2e,s}^{appr2}(0.0067s) = 14.304 \text{ Mbit} \geq b_{agg} = 2.016 \text{ Mbit} \Rightarrow$  approximation segment.

The four different combinations allow us to deduce which parameters have a positive or negative effect on the different delay bounds. For combinations C1 and C2, we used a small flow rate and varying burst sizes. The results for these combinations can be seen in Figure 5.2. For 5 servers, approximations 1 and 2 produce the same values, which are roughly 20% of the paper bound. For 30 servers, however, this drops to around 7%. This is a very drastic difference, which is unexpected considering how close the results were when using additive bounds.

For combinations C3 and C4, we raised the flow rate to its maximum possible value, which is 12.5 Mbit/s. For C3 we chose a value for  $b_1$  that is close to its minimum bound for case 1. For C4 we chose a very large value of  $b_1$  instead. While the results

for C3 fall in line with the ones for C1 and C2, the result for C4 is much closer to the paper bound, at least for 5 servers. For 30 servers, we can see the difference between the tightness of approximations 1 and 2 quite clearly. While the result of approximation 1 is 27.34% of the paper bound, approximation 2, which is the tighter one, is at only 6.45%. We generally notice that the values for C2-C4 do not change for 5 or 30 servers when using approximation 2. Looking at the delay bound we derived for this approximation, we know why this is the case: The bound does not scale with the number of servers. This is unsurprising, as we measure the delay bound with a line that has a server-independent slope. This means that we always hit the same point on the line, be it 5 or 100 servers.

We briefly take a look at case 2. For this, let  $b_1 = b_2 = 0.1$  Mbit. All other variable parameters are identical to C1. The values for both bounds can be found on the right side in Table 5.3. As expected, the min-plus delay bound for case 2 performs significantly better than the paper bound. This stems from the fact that the delay is calculated using the standard CR-TB bound  $\frac{b_{tot}}{R}$ . Assuming all servers have identical rates, this standard bound will be very small in comparison to the paper bound. However, we expect the bounds to be closer once we look at tandems of servers with drastically different rates, as the min-plus bound for case 2 is calculated with the smallest server in this tandem.

From the case 1 combinations, we can derive a few conclusions. Since the paper bound is calculated on the packet level, flow rates are not considered. This means that by increasing the flow rate, we only affect the approximations, as they are calculated on the flow level. Conversely, the packet lengths do not affect the min-plus delay bound. Interestingly enough, they do not impact the paper bound as much as we expected either. This is why we did not change the values in the combinations. The biggest impact lies with the burst sizes, especially together with larger flow rates. For bigger values of  $b_1$ , the difference to the paper bound becomes smaller, at least for small server sizes. For a large numbers of servers in the tandem the effect of the server concatenation becomes very impactful.

This leads us to another consideration, which is the network structure. Due to time restrictions, we only modeled SCs for basic networks in a tandem. In addition, no flow joins or leaves the class aggregate. We expect the difference between the min-plus delay bound and the paper bound to shrink further once we model and analyze more complex networks.

### 5.4.3 Comparison To The Counterexample

At last, we compare our delay bound to the seemingly tight counterexample we found in section 3.2.1. Recall the network structure shown in Figure 3.3. We have 5 servers and 2 LRQ-regulated flows in the class aggregate. Since we cannot reproduce this constraint due to LRQ being modeled in max-plus calculus, we exchange

Bound Type	Delay	Ratio	Bound Type	Delay <sub>30</sub>	Ratio
Paper	11.015ms	-	Paper	238.8ms	-
Approximation 1	2.34ms	21.24%	Min-Plus	8ms	3.35%
Approximation 2	2.25ms	20.43%			

Table 5.3: Comparison of the approximation delay bounds and paper bound for the counterexample given in Figure 3.2.1 (left), and comparison of case 2 min-plus delay bound and paper bound (right).

it with a TB constraint. We also consider CDT and BE traffic. We briefly restate the parameter selection:

- $f_1, f_2$  are TB-regulated with  $r_{f_i} = 20$  Mbit/s,  $b_{f_1} = 0.07$  Mbit,  $b_{f_2} = 0.016$  Mbit, maximum packet lengths  $M_{f_1} = 1$  Kbit,  $M_{f_2} = 2$  Kbit.
- Each node has CDT traffic with the parameters (20 Mbit/s, 4 Kbit), as well as BE traffic with a maximum packet length of 2 Kbit.
- The link rate is  $c_{ij} = 100$  Mbit/s for all links, and the CBS has the parameters  $I_{ij}^A = 50$  Mbit/s,  $S_{ij}^A = -50$  Mbit/s.
- The LOSC has the parameters  $T_{ij}^A = 0.00008s$ ,  $R_{ij}^A = 40$  Mbit/s.

In the counterexample, the paper bound produced a delay of  $700\mu s$ . Here, its delay is 11.015ms (compare Figure 5.3). For approximation 2, we need to determine which bound is used first. The jump point of a single server equals 0.875ms, therefore the height of each of the 5 stairs equals 0.0335 Mbit. This means that we need to calculate the delay bound for approximation 2 with the first segment. We observe similar results to the ones for 5 servers in the previous section, further confirming that the ratio of the min-plus delay bound and paper bound lies in the same range, independent of the server rate or CDT and BE traffic.

## 6 Conclusion And Future Work

We have glanced over the different approaches regarding the ensurance of real-time properties in Ethernet by the means of Time-Sensitive Networking. The approach of Asynchronous Traffic Shaping modeled by the usage of the Interleaved Regulator striked us as the most interesting. Hence, we made it the center of this thesis and have introduced its theoretical background in-depth. After figuring out that the delay bound, which is based on the IR, proposed by Mohammadpour et al. [10] could not be tight, we went through several established modeling approaches in the search for one that would accurately represent what the IR does. This led us to introducing a new service curve, which would have the same functionality for a flow aggregate consisting of two flows. We first derived specific curves for constant-rate servers, then were able to define the general cases based on the previously found ones. The jump to rate-latency servers was very easy thanks to the utilized property of convoluting a CR curve with the impulse function.

There are several things that are left to be worked on in regards to the proposed min-plus SC. First, we need to formally prove that the SC is correct. The same applies to several convolutions that we have done graphically due to their complicated nature. We also need to revise the proofs for the correctness of the point clouds in [12], since we have only assumed them to be correct in this thesis. Once this is done, we need to make the jump to arbitrarily large flow aggregates that can also change throughout the network. Since our modeling is based on flow aggregates, splitting off or adding flows will not be easily possible. A possible workaround for this might be the approach proposed in [13], where flows that enter the network at an arbitrary node will be extended to enter at the first node instead, allowing us to keep the flow aggregate approach.

We finally derived and compared the delay bounds of our approach with the bound given in [10]. While the additive bound comparison produced good results, the bounds using the server concatenation were very small in comparison. It is left to analyze as to why these bounds are so optimistic. A possible relativization in this context is the move to more complex network topologies, where the modeling approach using flow aggregates quickly becomes very restricted.



# Acronyms

**AC:** Arrival Curve.

**ATS:** Asynchronous Traffic Shaping.

**BE:** Best Effort.

**CBFS:** Class-Based FIFO System.

**CBS:** Credit-Based Shaper.

**CDT:** Control-Data Traffic.

**CR:** Constant-Rate.

**e2e:** end-to-end.

**foi:** flow of interest.

**GCL:** Gate Control List.

**IR:** Interleaved Regulator.

**LO:** Left-Over.

**LRQ:** Length Rate Quotient.

**NC:** Network Calculus.

**PBOO:** Pay Bursts Only Once.

**RL:** Rate-Latency.

**SC:** Service Curve.

**TAS:** Time-Aware Scheduling.

**TB:** Token-Bucket.

**TSN:** Time-Sensitive Networking.

**TTG:** Time-Triggered Gate.

**UBS:** Urgency-Based Scheduler.





# Bibliography

- [1] M. D. J. Teener, A. N. Fredette, C. Boiger, P. Klein, C. Gunther, D. Olsen, and K. Stanton, "Heterogeneous networks for audio and video: Using iee 802.1 audio video bridging," *Proceedings of the IEEE*, vol. 101, no. 11, pp. 2339–2354, 2013.
- [2] L. L. Bello, "Novel trends in automotive networks: A perspective on ethernet and the iee 802.1 audio video bridging," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, 2014, pp. 1–8.
- [3] D. Bruckner, M.-P. Stănică, R. Blair, S. Schriegel, S. Kehrer, M. Seewald, and T. Sauter, "An introduction to opc ua ts n for industrial communication systems," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1121–1131, 2019.
- [4] J. A. R. De Azua and M. Boyer, "Complete modelling of avb in network calculus framework," in *Proceedings of the 22nd International Conference on Real-Time Networks and Systems*, 2014, pp. 55–64.
- [5] J. Liebeherr, "Duality of the max-plus and min-plus network calculus," *Foundations and Trends in Networking*, vol. 11(3-4), pp. 139–282, 2017.
- [6] J. Schmitt and P. Nikolaus, "Lecture notes in worst-case analysis of distributed systems," 2021.
- [7] J.-Y. L. Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer-Verlag, 2001.
- [8] J. Specht and S. Samii, "Urgency-based scheduler for time-sensitive switched ethernet networks," in *28th Euromicro Conference on Real-Time Systems (ECRTS)*, Jul. 2016, pp. 75–85.
- [9] J.-Y. Le Boudec, "A theory of traffic regulators for deterministic networks with application to interleaved regulators," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2721–2733, 2018.
- [10] E. Mohammadpour, E. Stai, M. Mohiuddin, and J.-Y. L. Boudec, "End-to-end latency and backlog bounds in time-sensitive networking with credit based shapers and asynchronous traffic shaping," in *Proceedings of the ITC Network Calculus Workshop*, Sep. 2018, pp. 1–6.
- [11] C. S. Chang and Y. H. Lin, "A general framework for deterministic service guarantees in telecommunication networks with variable length packets," in *Proc. 6th International Workshop on Quality of Service (IWQoS)*, May 1998, pp. 49–58.

- [12] K. Pandit, “Quality of service performance analysis based on network calculus,” Ph.D. dissertation, Technische Universität Darmstadt, 2006.
- [13] S. Bondorf and F. Geyer, “Virtual cross-flow detouring in the deterministic network calculus analysis,” in *2020 IFIP Networking Conference (Networking)*, 2020, pp. 554–558.