Worst-Case Performance Analysis of Feed-Forward Networks – An Efficient and Accurate Network Calculus

Thesis approved

by the Department of Computer Science

of the University of Kaiserslautern (TU Kaiserslautern)

for the award of the Doctoral Degree

Doctor of Engineering (Dr.-Ing.)

to

Steffen Bondorf

Date of the viva	:	12 July 2016
Dean:	:	Prof. Dr. Klaus Schneider
PhD committee		
Chairperson	:	Prof. Dr. Heike Leitte
Reviewers	:	Prof. Dr. Jens B. Schmitt Prof. Dr. Roland Meyer Dr. Marc Boyer

D 386

Abstract

Distributed systems are omnipresent nowadays and networking them is fundamental for the continuous dissemination and thus availability of data. Provision of data in real-time is one of the most important non-functional aspects that safety-critical networks must guarantee. Formal verification of data communication against worst-case deadline requirements is key to certification of emerging x-by-wire systems. Verification allows aircraft to take off, cars to steer by wire, and safety-critical industrial facilities to operate. Therefore, different methodologies for worst-case modeling and analysis of real-time systems have been established. Among them is deterministic Network Calculus (NC), a versatile technique that is applicable across multiple domains such as packet switching, task scheduling, system on chip, software-defined networking, data center networking and network virtualization. NC is a methodology to derive deterministic bounds on two crucial performance metrics of communication systems:

(a) the end-to-end delay data flows experience and

(b) the buffer space required by a server to queue all incoming data.

NC has already seen application in the industry, for instance, basic results have been used to certify the backbone network of the Airbus A380 aircraft.

The NC methodology for worst-case performance analysis of distributed real-time systems consists of two branches. Both share the NC network model but diverge regarding their respective derivation of performance bounds, i.e., their analysis principle. NC was created as a deterministic system theory for queueing analysis and its operations were later cast in a (min,+)-algebraic framework. This branch is known as algebraic Network Calculus (algNC). While algNC can efficiently compute bounds on delay and backlog, the algebraic manipulations do not allow NC to attain the most accurate bounds achievable for the given network model. These tight performance bounds can only be attained with the other, newly established branch of NC, the optimization-based analysis (optNC). However, the only optNC analysis that can currently derive tight bounds was proven to be computationally infeasible even for the analysis of moderately sized networks other than simple sequences of servers.

This thesis makes various contributions in the area of algNC: accuracy within the existing framework is improved, distributivity of the sensor network calculus analysis is established, and most significantly the algNC is extended with optimization principles. They allow algNC to derive performance bounds that are competitive with optNC. Moreover, the computational efficiency of the new NC approach is improved such that this thesis presents the first NC analysis that is both accurate and computationally feasible at the same time. It allows NC to scale to larger, more complex systems that require formal verification of their real-time capabilities.

Acknowledgements

I would like to thank my parents, Rita and Hans-Georg, for their endless support that allowed me to achieve my goals. My heartfelt gratitude inevitably goes to my significant other Elena, who supported me during this entire time. You have continuously encouraged me throughout this research effort and without you this dissertation would not have been possible.

I am thankful to Prof. Jens Schmitt for being my mentor, my friend. Your decision to push me to network calculus when our paths crossed in 2008 shaped my academic life ever since. Thank you for guiding me through the endeavors of a bachelor's thesis, a master's thesis and finally this PhD thesis. Thank you for believing in me.

I would also like to thank Prof. Roland Meyer for helping me as a co-supervisor since the start of my doctorate studies and Marc Boyer for all the discussions we had at the various network calculus events. Thank you both for providing valuable suggestions.

My sincere appreciation belongs to everyone I met at DISCO and who made my time worthwhile. I want to thank all the former and current team members, namely Matthias Schäfer, Daniel Berger, Michael Beck, Hao Wang, Matthias Wilhelm, Wint Yi Poe, Adam Bachorek, Nicos Gollan, Ivan Martinovic and Paul Nikolaus. You certainly had an impact on me and my research. My special gratitude is also extended to our associated staff Markus Fuchs, Steffen Reithermann and Barbara Erlewein.

Thank you all,
Steffen

Contents

Abstract iii

Acknowledgements v

1 Introduction 1
- 1.1 Worst-Case Performance Analysis 1
- 1.2 Network Calculus 2
- 1.3 Thesis Contribution 3
- 1.4 Thesis Organization 4

2 State of the Art in Network Calculus 7
- 2.1 Network Description 7
  - 2.1.1 Data Arrivals, Forwarding Service and Performance Characteristics 7
  - 2.1.2 The Network Model 9
  - 2.1.3 (min,+)-algebraic Operations and Performance Bounds 10
- 2.2 Analysis Principles 13
- 2.3 Algebraic Analyses 14
  - 2.3.1 Total Flow Analysis 14
  - 2.3.2 Separate Flow Analysis 16
  - 2.3.3 Pay Multiplexing Only Once Analysis 17
- 2.4 Optimization-based Analysis 20
  - 2.4.1 The Linear Programming Analysis 21

3 Accuracy of Algebraic Network Calculus 23
- 3.1 Compositional Analysis of Feed-forward Networks 23
- 3.2 Problems 25
  - 3.2.1 Problems Caused by Compositionality 25
  - 3.2.2 Problems Caused by Tandem Analyses 26
- 3.3 Derivation of Cross-traffic Arrival Bounds 28
  - 3.3.1 Segregated Cross-flow Arrival Bounding 29
  - 3.3.2 Aggregate Cross-traffic Arrival Bounding 29
  - 3.3.3 Deconvolution for $\alpha \in \mathcal{F}_{\mathrm{mTB}}$ with $\beta \in \mathcal{F}_{\mathrm{mRL}}$ 31
  - 3.3.4 Distributivity of $\oslash$ over $+$ 32
  - 3.3.5 Ranking of PBOO Arrival Bounding Alternatives 37
  - 3.3.6 Accuracy Evaluation 40
- 3.4 Cross-traffic Burstiness Reduction 42
  - 3.4.1 An Alternative Output Bound 43

3.4.2 TFA-assisted aggrPBOOAB 46
3.4.3 Accuracy Evaluation 51

4 Distributed Sensor Network Calculus Analysis 55
4.1 Arrivals, Service, (min,+)-Operations and Performance Bounds 55
4.2 The Network Model 57
4.3 Compositional Sink-tree Analysis 57
4.3.1 Flow of Interest Analysis 57
4.3.2 Cross-traffic Arrival Bounding 60
4.4 Improving SensorNC Analysis 61
4.4.1 The Generalized Concatenation for SensorNC 61
4.4.2 The New SensorNC Cross-traffic Arrival Bounding 65
4.4.3 Accuracy Evaluation 67
4.4.4 Effort Evaluation 68
4.5 Distribution of the SensorNC Analysis Procedure 70

5 Limitations of Linear Programming Analysis 75
5.1 Scalability Issues 75
5.2 A Tradeoff between Accuracy and Efficiency 76
5.2.1 The Unique Linear Programming Analysis 77
5.3 Evaluation of Accuracy and Effort 78
5.3.1 The Non-Nested Tandem 78
5.3.2 The Square Network 81
5.3.3 Feed-forward Networks 84

6 Optimization Principles in Algebraic Network Calculus 89
6.1 Exhaustive Aggregate Cross-traffic Arrival Bounding 89
6.1.1 Accuracy Evaluation 93
6.1.2 Effort Considerations 97
6.2 The Tandem Matching Analysis 97
6.2.1 Matching Tandems 98
6.2.2 Related Work 101
6.2.3 Accuracy Evaluation 103
6.2.4 Effort Considerations 105
6.3 Combining Search Spaces of Optimization Principles 106

7 Efficiency of Algebraic Network Calculus Analysis 109
7.1 Analysis Execution Efficiency 109
7.1.1 Convolution of Intermediate Arrival Bounds 109

7.1.2 Caching of Intermediate Arrival Bounds 110
7.2 The Efficient TMA Algorithm 111
7.2.1 Effort Evaluation 114
7.3 Network Calculus on Compact Domains 116
7.3.1 Related Work 118
7.3.2 Deriving Compact Curve Domains 119
7.3.3 Accuracy and Effort Evaluation 125

8 The New State of the Art in Network Calculus 129
8.1 Numerical Experiments I: Erdős-Rényi Networks 130
8.2 Numerical Experiments II: General Linear Preference Networks 132
8.3 Case Study: Avionics Full-Duplex Ethernet 137

9 Conclusion 141
9.1 Concluding Remarks 141
9.2 Future Research Directions 142

Nomenclature 147

List of Figures 148

List of Tables 149

References 151

List of Publications 159

CV 161

1 Introduction

1.1 Worst-Case Performance Analysis

A fundamental problem in the analysis of real-time networks is the derivation of deterministic performance bounds. The most important measures are the delay data flows experience when they traverse a network from their respective source to their sink and the backlog they experience at the individual servers they cross. The latter is required to dimension buffers sufficiently large to avoid overflows and thus the loss of data. Lost or dropped data needs to be retransmitted and increases the delay between a flow's initial creation and its eventual delivery. Depending on the payload, this delay might be decisive for a safety-critical real-time system. Therefore, worst-case performance bounds play a crucial role in different areas, ranging from verification of hard real-time communication capabilities to quality of experience assurance for end users. In case of safety-critical systems, formally verified delay and backlog bounds are even required for certification. I.e., for larger systems such as aircraft that rely on an embedded real-time communication network, they are crucial for the entrance into service.

Networks that are required to guarantee for real-time communication have been deployed in safety-critical environments for decades. These networks are verified to keep certain deadlines when transmitting information. However, they are oftentimes specifically built for a single purpose and to operate under unalterable conditions only – and so is their evaluation customized to this setting. Nowadays, we observe the trend to employ commercial off-the-shelf technologies. For instance, cable harnesses with their dedicated connections have been replaced by shared-access to a vehicle bus systems like CAN (ISO 11898). For multiple reasons such as complexity, development time and costs, the industry developed communication network standards based on Ethernet (IEEE 802.3), a widely used standard for local area networks. Airbus, a manufacturer of aircraft, based its Avionics Full-Duplex Switched Ethernet (AFDX: ARINC 664) on this standard, the automotive industry developed Audio Video Bridging (AVB: IEEE 802.1BA, 802.1AS, 802.1Qat, 802.1Qav) from the IEEE 802 family and in industrial facilities, usage of Time-Triggered Ethernet (TTEthernet: SAE AS6802) emerges. CAN has also be amended by a time-triggered variant to increase its real-time capabilities (TTCAN: ISO 11898-4:2004) [33, 27, 39, 24, 28, 63].

Embedded into a safety-critical system such as a vehicle, networks based on one of these standards must warrant hard real-time guarantees. Formal verification is key to obtain deterministic worst-case bounds on delay and backlog and thus to the certification of the entire system. Simulation is not sufficient. Changing requirements and emerging technologies employed in such systems demand extensibility and flexibility from a formal performance evaluation methodology. There are various approaches for specification and verification of real-time systems, yet, only few are powerful w.r.t. their modeling capabilities such that they can be applied to diverse environments, e.g., switched Ethernets [32], wireless sensor networks [71, 43, 8, 65], network on chip [56, 67, 61], Internet of Things [88, 5, 62], software-defined networking and network virtualization [39, 62], data centers [89] as well as network simulation [47]. Moreover, the analysis based on the model must be able to scale to large, complex systems that require formal verification of their real-time capabilities. One of most versatile techniques that promises both of these properties is Network Calculus (NC) [25, 26, 23, 50], a methodology for the derivation of worst-case bounds on flow delay and server backlog.

1.2 Network Calculus

NC provides a mathematical framework to derive deterministic bounds on a flow's end-to-end delay when crossing a network and the buffer size requirement at the traversed servers, i.e., the bound on their maximum backlog at any time. NC has been widely adopted in research as well as industry – for example, the switched Ethernet backbone network used in the Airbus A380 has been certified using NC.

The evolution of the NC methodology resulted in two branches that only share the NC system description: On the one hand, there is the algebraic NC (algNC) branch. Its analysis of networks free of cyclic dependencies (feed-forward property) is based on the extension of results for sequences of servers, the so-called tandems. AlgNC established three distinct analyses to compute performance bounds, TFA, SFA and PMOOA. Each of these analyses defines, among other aspects, a specific way to decompose the network into tandems, their order of analysis and the applied operations. Unfortunately, these compositional algNC analyses are not able to derive tight bounds, i.e., the most accurate bounds are known to be out of their reach. In terms of bound accuracy, algNC has been superseded by the optimization-based NC (optNC) branch. While this work started with a compositional analysis as well, the most recent approach abandoned the decomposition procedure. It aims at a network-wide, global optimization: The entire network description is transformed into a set of linear programs to solve. This constitutes the only analysis currently known to achieve tight bounds on end-to-end flow delays and server backlog in general feed-forward networks. However, the amount of linear programs to solve grows super-exponentially with the network size. The analysis is shown to be NP-hard with no algorithm known to solve the underlying problem efficiently (extension of a partial order to the set of all compatible total orders). Therefore, the optNC branch is commonly represented by an analysis variant circumventing this problem. Tightness is traded against computational feasibility by reducing the amount of linear programs to a single one. Naturally, this optNC analysis does not capture interference patterns of flows as exhaustively as the tight one. It is, however, believed to be both, efficient and accurate. The former is concluded from decades of research on optimization and its tool support, the latter from the attained global view of the optimization. Yet, neither of these assumptions was profoundly evaluated.

Therefore, only algNC has seen application in the industry. It was applied to analyze networks designed from scratch for delay sensitive employment. Especially the avionics industry has embraced the algNC methodology. Examples can be found in the analysis of HCS (Heterogeneous Communication System) [80] and AFDX (Avionics Full-Duplex Switched Ethernet). The latter has continuously seen attention in order to verify against timing constraints for these networks [32, 20, 38, 54, 19]. Current Airbus aircraft's AFDX backbone was certified using the algebraic techniques provided by NC.

The application of algNC has also resulted in diverse tool support. It ranges from open-source tools provided by academia [18, 2, 7], to a freely available closed-source toolbox [86], to a commercial offering [21] and others [57, 17]. Moreover, several companies are known to employ NC and develop internal tool support – for example Rockwell Collins (ConfGen [36]), Hirschmann Automation and Control (DelayLyzer [69]) and SIEMENS (NC Engine [44]) to name a few.

Theoretical and practical enhancements of algNC are in the focus of this thesis.

1.3 Thesis Contribution

This thesis makes several contributions in the area of NC feed-forward analysis, under the assumption of arbitrary multiplexing: First, the accuracy of current state-of-the-art algNC analysis is improved with two distinct enhancements. In addition to the impact on performance bounds, this contribution also highlights that algNC's potential has not been fully exploited before. Moreover, the algebraic properties accessible during the algNC analysis also allow for features not attained by current optNC analyses: assessment of intermediate results and distributed execution. This thesis contributes an extension to a procedure for distributed algNC analysis that enables for in-network performance bounding. It can be employed for distributed admission control as well as monitoring tasks within self-modeling sensor networks. OptNC's reliance on optimization software introduces a black-box view on the analysis that prohibits such features. Indeed, we also show that every optNC analysis imposes computational effort on this software that renders its application nearly impossible – even for moderately sized networks as found embedded into systems such as aircraft that are currently in operation. These observations reveal that existing NC analyses can be distinguished regarding their fundamental tradeoff between efficiency and accuracy:

1. AlgNC is computationally efficient and possesses the potential to provide useful

features like distributed in-network execution. However, accuracy of attained performance bounds is not competitive with optNC.

2. OptNC, on the other hand, theoretically allows for the derivation of tight bounds. Yet, this thesis shows that in practice neither tight nor accurate bounds are computationally feasible to derive with an optNC analysis.

Therefore, we contribute a novel NC analysis that is based on algNC but incorporates optimization principles in order to achieve highly accurate bounds. In particular, it exploits the idea to consider the entire feed-forward network in the search for performance bounds instead of focussing on individual tandem solutions only. The analysis can make use of a vast search space but this optimization principle at the core of the new approach results in computational effort similar to optNC. Yet, in contrast to the back-box view imposed by optimization software, algNC grants access to the entire inner workings of the analysis procedure. We show how this knowledge can be used for multiple efficiency improvements, both fundamentally and implementation-wise. They allow for a fast search for performance bounds in the search space defined by our new NC analysis approach. In an extensive evaluation, we demonstrate that this is the first NC analysis to achieve the high degree of accuracy previously exclusively attributed to optNC while demanding computation times that are several orders of magnitude below optNC. This thesis therefore provides the first efficient and accurate NC analysis applicable to large-scale feed-forward networks.

1.4 Thesis Organization

In Section 2, we provide an in-depth depiction of NC with a focus on the properties required for the later contributions. We start with the curves bounding worst-case flow arrivals and worst-case forwarding service that are common to all NC analyses (Section 2.1). Then, we present the network model that NC analyses are applied to and justify the focus on arbitrary multiplexing in server queues, i.e., we assume that flows can be reordered arbitrarily when they share a queue. The foundations of NC are concluded by the (min,+)-algebra algNC is based on. We cover operations that manipulate curves such that the worst-case model is retained as well as the derivation of deterministic bounds on flow delays and server backlog. We then depict the known NC analysis principles (Section 2.2) followed by the algNC tandem analyses and the optNC feed-forward analysis that implement these principles (Sections 2.3 and 2.4).

Section 3 is dedicated to the accuracy of current algNC. After contributing a generic composition scheme that allows algNC tandem analyses to derive performance bounds in feed-forward networks, we identify the problems caused by compositionality (Sections 3.1 and 3.2). Then, we focus on improvements to algNC's accuracy within this approach. In

Section 3.3, we improve the overall procedure that derives bounds on cross-traffic arrivals and Section 3.4 is dedicated to the specific aspect of cross-traffic burstiness.

Non-functional properties of the algNC analysis are addressed in Section 4. For sensor network calculus (SensorNC), a specialized variant of algNC (Sections 4.1 and 4.2), we augment the sink-tree analysis it applies (Section 4.3) with a tailored sink-tree procedure to bound cross-traffic arrivals that can also be executed in a distributed fashion (Section 4.4). In Section 4.5, we conclude this work on non-functional aspects with a scheme that allows algNC performance bounds to be computed in the sensor network itself.

After we improved several aspects of algNC, we turn to optNC in Section 5. An analysis of its limited scalability is provided in Section 5.1 and Section 5.2 presents the tradeoff between accuracy and computational efficiency suggested in the literature. We comprehensively benchmark this tradeoff against algNC analyses in Section 5.3 in order to gain insight on the gap in accuracy that algNC has to close.

In Section 6, we incorporate optimization principles into the algNC analysis. We contribute two approaches to establish a hybrid NC analysis: First, we depict an extension of the search space for cross-traffic arrival bounding (Section 6.1). Secondly, we identify previously not applied but potentially beneficial sequences of algebraic operations that extend the search space of the entire algNC analysis (Section 6.2). Finally, we combine both search spaces in Section 6.3 to obtain accurate performance bounds.

Section 7 is dedicated to the computational efficiency of algNC. We contribute two improvements that reduce computational effort and thus allow the analysis – including our extensively searching new approach – to scale to network sizes previously infeasible to analyze with NC (Sections 7.1 and 7.2). In Section 7.3, we investigate performance bound improvements caused by a more accurate network modeling and derive a solution to the vast increase in effort such an accurate model imposes on the algNC analysis.

Section 8 provides an extensive comparison of the previous state of the art in NC and the new one established by the contributions of this thesis. Section 9 concludes the thesis and provides directions for future research on the topic of efficient and accurate feed-forward network analysis with NC.

2 State of the Art in Network Calculus

This section provides an introduction to the modeling and analysis capabilities of NC. Within the comprehensive possibilities NC offers, we are concerned with analyzing systems that retain the order of data within a flow. This property is known as FIFO per µFlow [25, 64, 70]. When multiplexing multiple flows, however, we do not assume any knowledge about the resulting order among the different flows. I.e., in a subsequent queue their data can be arranged in any order. This behavior is known as arbitrary multiplexing or blind multiplexing. Other behaviors that are analyzed in the NC literature are First In, First Out (FIFO) multiplexing, Strict Priority (SP) multiplexing or the lack of FIFO per µFlow, i.e., even the order of data within an individual flow may change arbitrarily. FIFO multiplexing is a special case of arbitrary multiplexing and non-FIFO per µFlow is a generalization of our assumed setting. We will discuss the case of differing assumptions where it is helpful for further reading.

2.1 Network Description

2.1.1 Data Arrivals, Forwarding Service and Performance Characteristics

Flows are characterized by functions cumulatively counting their data. They belong to the set $\mathcal{F}_0$ of non-negative, wide-sense increasing functions that pass through the origin:

$$\begin{aligned} \mathcal{F}_0 \;&=\; \left\{ f : \mathbb{R} \to \mathbb{R}^+_\infty \;\middle|\; f(0) = 0,\ \forall \underline{t} \leq t \,:\, f(\underline{t}) \leq f(t) \right\}, \\ &\mathbb{R}^+_\infty := [0, +\infty) \cup \{+\infty\} \;. \end{aligned}$$

We are particularly interested in the functions $A(t)$ and $A'(t)$ cumulatively counting a flow's data put into a server s and put out from s, both from the start of operation up until time t. We further demand servers and flows to preserve causality by fulfilling the flow constraint, i.e., $\forall t \in \mathbb{R}^+ : \; A(t) \geq A'(t)$. Then, these functions allow performance characteristics of a queuing system to be defined in a straight-forward manner.

Definition 2.1. (Backlog and Delay) Assume a flow with input function A traverses a server s and results in the output function A'.

- The backlog of the flow at time t is defined as

$$B(t) = A(t) - A'(t).$$

- The *(virtual)* delay for a data unit arriving at s at time t is defined as

$$D(t) = \inf \left\{ \tau \geq 0 \;\middle|\; A(t) \leq A'(t + \tau) \right\}.$$

Note, that the FIFO per µFlow assumption is crucial for the virtual delay definition. It argues that the expected delay is caused by data that entered s before the data unit under analysis and is therefore served before it. In the defined setting, this data can only belong to the analyzed flow. The backlog bound, in contrast, does not relate A and A' at different times. Thus, it is not influenced by the server behavior between these time instances, i.e., the backlog derivation is independent of the FIFO per µFlow assumption.

NC models data arrivals with curves (from $\mathcal{F}_0$) that bound behavior in the interval time domain, i.e., whereas the function value $A(t)$ returns the data cumulated in the interval $[0, t]$, the NC arrival curve for $A(t)$, $\alpha(d)$, returns an upper bound on data arrivals for any duration of length d, e.g., $d = t - 0$.

Definition 2.2. (Arrival Curve) Let a flow have input function $A \in \mathcal{F}_0$, then $\alpha \in \mathcal{F}_0$ is an arrival curve for A iff it bounds A in any time interval of duration d, i.e.,

$$\forall t\, \forall d,\ 0 \leq d \leq t\ :\ A(t) - A(t-d) \leq \alpha(d).$$

We additionally demand that arrival curves fulfill $\alpha(0) = 0$, i.e., there are no instantaneous arrivals.

A useful basic shape for arrival curves is the so-called token bucket. These curves are from the set $\mathcal{F}_{\mathrm{TB}} \subseteq \mathcal{F}_0$,

$$\mathcal{F}_{\mathrm{TB}} = \left\{ \gamma_{r,b} \mid \gamma_{r,b}(d) = \begin{cases} 0 & \text{if } d = 0 \\ b + r \cdot d & \text{otherwise} \end{cases},\ r, b \in \mathbb{R}_{\infty}^{+} \right\},$$

where r denotes the maximum arrival rate and b is the maximum burstiness (bucket size). A common generalization if $\mathcal{F}_{\mathrm{TB}}$ is the set of multi-token-bucket curves $\mathcal{F}_{\mathrm{mTB}}$

$$\mathcal{F}_{\mathrm{mTB}} = \left\{ \bigwedge_{i=1}^{n} \gamma_{r_i,b_i} \mid \gamma_{r_i,b_i} \in \mathcal{F}_{\mathrm{TB}} \right\} \subseteq \mathcal{F}_0.$$

They are able to represent different traffic constraints for different time scales [87], each defined by a token bucket.

Scheduling and buffering at a server result in the output function $A'(t)$. NC captures the minimum forwarding capabilities that lead to A' in interval time as well.

Definition 2.3. (Service Curve) If the service provided by a server s for a given input A results in an output A', then s is said to offer a (simple) service curve $\beta \in \mathcal{F}_0$ iff

$$\forall t\ :\ A'(t) \geq \inf_{0 \leq d \leq t} \{A(t-d) + \beta(d)\}.$$

A number of servers fulfill a stricter definition of service curves by considering their internal state in addition to their input A.

Definition 2.4. (Backlogged Period) A server s is backlogged at time t if $A(t)-A'(t)>0$ and it is backlogged during period $(\underline{t},\overline{t})$ if $\forall t,\ \underline{t}<t<\overline{t} \mid A(t)-A'(t)>0$.

Servers offering strict service curve guarantees have a higher output during backlogged periods.

Definition 2.5. (Strict Service Curve) If, during any backlogged period of duration $d=\overline{t}-\underline{t}$, a server s with input A guarantees an output of at least $\beta(d)$, it is said to offer a strict service curve $\beta \in \mathcal{F}_0$.

A basic shape for service curves is the rate-latency curve defined by the set $\mathcal{F}_{\mathrm{RL}} \subseteq \mathcal{F}_0$,

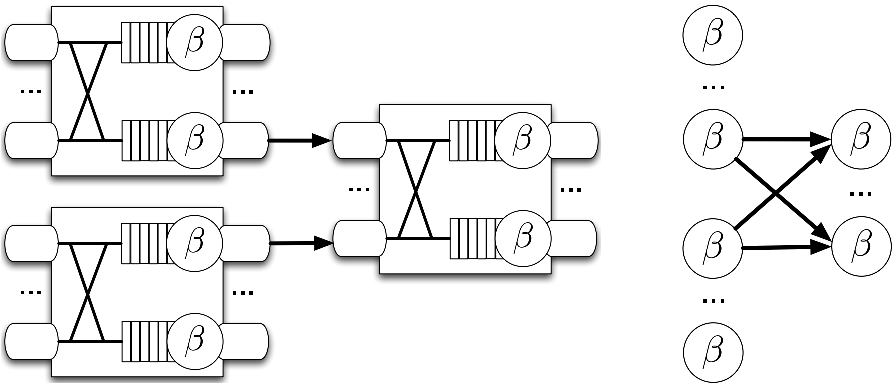
$$\mathcal{F}_{\mathrm{RL}} = \left\{\beta_{R,T} \;\middle|\; \beta_{R,T}(d) = \max\left\{0,\ R\cdot(d-T)\right\},\ R,T \in \mathbb{R}^+_\infty \right\},$$

where R denotes the minimum service rate and T is the maximum latency. Multi-rate-latency curves are defined by the pointwise maximum over a set of rate latencies

$$\mathcal{F}_{\mathrm{mRL}} = \left\{ \bigvee_{j=1}^{m} \beta_{R_j,T_j} \;\middle|\; \beta_{R_j,T_j} \in \mathcal{F}_{\mathrm{RL}} \right\} \subseteq \mathcal{F}_0.$$

2.1.2 The Network Model

Topology Data communication networks are commonly modeled as graphs where nodes represent individual devices like a router or a switch. These devices can have multiple outputs to connect to other devices (Figure 2.1a). This common depiction does, however, not suit NC's queueing analysis well. NC therefore transforms such a device graph to its so-called server graph representation. Assuming that a device's input buffer is served at line speed, queueing effects manifest at the output buffers. These are modeled by the server graph's nodes [6, 8] (see Figure 2.1b). Figure 2.1 also illustrates that information about the device's sub-components is lost during the transformation from a device graph to a server graph. Especially the highly optimized switching fabric inside network devices that connects inputs with outputs is, however, crucial for our considerations. Although the servers most likely work off queued data in a FIFO manner, this fabric interconnecting input ports with output ports can rearrange flows arbitrarily [29]. In the server graph, however, device ports (i.e., servers) do not share a common switching fabric component; they are directly connected to each other. The impact of switching fabrics can thus only be captured by departing from FIFO multiplexing towards the more general arbitrary multiplexing assumption. Data units of individual flows do not overtake each other in



(a) Device Graph. (b) Server Graph.

Figure 2.1: A graph of network devices with output buffering (a) and its transformation to directly connected servers (b).

switching fabrics, i.e., the FIFO per µFlow property is retained. We assume directed links in both graph representations, i.e., full duplex links need to be split up into two directed links before the device graph can be transformed [80].

Data Flows and The Feed-forward Property NC also has some restricting assumptions concerning flows routed through the server graph. Firstly, the end-to-end analyses of NC assume unicast data flows. Secondly, these advanced NC analyses require the absence of cyclic dependencies between flows. Work departing from this assumption can be found in [68, 41] but is not covered by this thesis. Instead, we focus on networks that guarantee the feed-forward property by design. We use the Turn Prohibition (TP) algorithm [79] to break potential cycles when transforming the device graph into the server graph [34][1].

In this thesis, we will use the term *network* to refer to a server graph and the data flows traversing it.

2.1.3 (min,+)-algebraic Operations and Performance Bounds

Network calculus [25, 26] was cast in a $(\min,+)$-algebraic framework in [50, 23]. The following operations manipulate arrival and service curves while retaining their worst-case semantic.

---

[1] A "turn" is a device's input/output connection, i.e., it corresponds to a link in the server graph representation [34]. Prohibiting turns results in a cycle-free server graph. TP does not consider the route of flows in the device graph. Therefore, flows must be routed in the turn-prohibited server graph in order not to break their paths. Note, that shortest paths in the turn-prohibited server graph might not coincide with the shortest path in the device graph.

Definition 2.6. ((min,+)-Operations) The (min,+) aggregation, convolution and deconvolution of two functions $f, g \in \mathcal{F}_0$ are defined as follows:

$$\begin{aligned}
\text{Aggregation:} \quad (f+g)(t) &= f(t)+g(t) \\
\text{Convolution:} \quad (f \otimes g)(t) &= \inf_{0 \le \underline{t} \le t} \{f(t-\underline{t}) + g(\underline{t})\} \\
\text{Deconvolution:} \quad (f \oslash g)(t) &= \sup_{u \ge 0} \{f(t+u) - g(u)\}
\end{aligned}$$

A flow aggregate is thus a combination of flows that does not preserve information about the aggregated flow individually. The service curve definition translates to $A' \geq A \otimes \beta$, the arrival curve definition to $A \otimes \alpha \geq A$, and performance characteristics can be bounded using the deconvolution $\alpha \oslash \beta$. The algebraic properties of these operations can be found in [50]. Table 2.1 provides the more comprehensive notation scheme for NC performance bounding we use in this thesis.

Theorem 2.1. *(Performance Bounds) Consider a server s that offers a service curve β. Assume a flow f with arrival curve α^f traverses the server. Then, we obtain the following performance bounds for f:*

$$\begin{aligned}
\text{(Flow) Delay Bound:} \quad \forall t \in \mathbb{R}^+ : \quad D^f(t) &\le \sup_{u \ge 0} \left\{ \inf \left\{ \tau \ge 0 \mid \alpha^f(u) \le \beta(u+\tau) \right\} \right\} \\
&= \inf \left\{ \tau \ge 0 \,\middle|\, \left(\alpha^f \oslash \beta\right)(-\tau) \le 0 \right\} \\
&= h\left(\alpha^f, \beta\right) =: D^f
\end{aligned}$$

$$\begin{aligned}
\text{Backlog Bound:} \quad \forall t \in \mathbb{R}^+ : \quad B^f(t) &\le \sup_{r \ge 0} \left\{ \alpha^f(r) - \beta(r) \right\} \\
&= \left(\alpha^f \oslash \beta\right)(0) \\
&= v\left(\alpha^f, \beta\right) =: B^f
\end{aligned}$$

$$\text{Output Bound:} \quad \forall d \in \mathbb{R}^+ : \quad \left(\alpha^f\right)'(d) = \begin{cases} 0 & \text{if } d = 0 \\ \left(\alpha^f \oslash \beta\right)(d) & \text{otherwise} \end{cases}$$

$h(\alpha^f, \beta)$ denotes the (maximum) horizontal deviation between α and β and $v(\alpha^f, \beta)$ denotes their (maximum) vertical deviation. We abbreviate delay and backlog bounds with D^f and B^f, respectively, as they are valid independent of parameter t. Note, that α' is an arrival curve for A' and thus it is required to pass through the origin[2].

Consider an aggregate of flows $\mathbb{F}$ with arrival curve $\alpha^{\mathbb{F}}$. The aggregate's delay when

---

[2] In a slight abuse of notation, we will use the symbol $\oslash$ for both, deconvolution and output bounding.

Quantifier	Definition
foi	Flow of interest, the flow under analysis
$\mathbb{F}$	Aggregate of flows
$[f_n, ..., f_m]$	Flow aggregate containing flows $f_n, ..., f_m$
$F(s)$	Set of flows at server s
$F_{\text{src}}(s)$	Set of flows entering the network at server s
$x(f)$, $x(\mathbb{F})$	Cross-traffic of flow f, aggregate $\mathbb{F}$
$\langle s_x, \ldots, s_y \rangle$	Tandem of consecutive servers s_x to s_y
$P(f)$	Path of flow f
α^f, $\alpha^{\mathbb{F}}$	Arrival curve of flow f, set of flows $\mathbb{F}$ at their (common) source
α_s^f, $\alpha_s^{\mathbb{F}}$	Arrival bound at server s
α_s	Abbreviation for $\alpha_s^{F(s)}$, i.e., arrivals of all flows at s
β_s	Service curve of server s
$\beta^{\text{l.o.}f}$, $\beta^{\text{l.o.}\mathbb{F}}$	Left-over service curve

Table 2.1: Network calculus notation for flows, arrivals and service.

crossing a server s with strict *service curve β is bounded by:*

(Flow Aggregate) Delay Bound:

$$\begin{aligned}\forall t \in \mathbb{R}^+ : \quad D^{\mathbb{F}}(t) \quad &\leq \quad \max\left\{0, \sup\left\{ \underline{t} - \bar{t} \,\middle|\, \forall t \in (\underline{t}, \bar{t}).\, \alpha(t) - \beta(t) > 0 \right\}\right\} \\ &= \quad bp(\alpha^{\mathbb{F}}, \beta) \;=:\; D^{\mathbb{F}}\end{aligned}$$

where $bp\left(\alpha^{\mathbb{F}}, \beta\right)$ is maximum backlogged period of β w.r.t. $\alpha^{\mathbb{F}}$.

Under arbitrary multiplexing assumptions, the lack of knowledge about ordering of flows within the aggregate requires a different delay bounding for $\mathbb{F}$. Whereas FIFO assumptions (per µFlow and when aggregating flows) allow NC to bound the delay with $h\left(\alpha^{\mathbb{F}}, \beta\right)$, potential reordering of flows may lead to a worse delay bound. Therefore, we bound the delay of flow aggregates with $bp\left(\alpha^{\mathbb{F}}, \beta\right)$. Note, that the maximum backlogged period derivation requires a strict service curve.

NC also offers operations for compound analysis of servers as well as the separate analysis of individual flows.

Theorem 2.2. *(Concatenation of Servers) Consider a flow (aggregate) $\mathbb{F}$ crossing a tandem of servers $\mathcal{T} = \langle s_1, \ldots, s_n \rangle$ and assume that each s_i, $i \in \{1, \ldots, n\}$, offers a* *service curve* β_{s_i}*. The overall service curve offered to* $\mathbb{F}$ *is their concatenation*

$$\beta_{s_1} \otimes \ldots \otimes \beta_{s_n} = \bigotimes_{i=1}^{n} \beta_{s_i} = \beta_{\mathcal{T}}.$$

The service resulting from the concatenation of strict service curves, $\beta_{\mathcal{T}}$, is not necessarily strict. It cannot be used to derive a backlogged period bound of the compound system [11, 31]. This imposes challenges if the FIFO per µFlow property cannot be assumed [64, 70]. Also note, that convolution is commutative, i.e., the order of servers in $\langle s_1, \ldots, s_n \rangle$ cannot be reconstructed from $\beta_{\mathcal{T}}$.

The worst-case share of data a flow (aggregate) receives from a server is lower bounded by the left-over service curve:

Theorem 2.3. *(Left-Over Service Curve) Consider a server* s *that offers a strict service curve* β_s*. Let* s *be crossed by flow (aggregate)* $\mathbb{F}_0$ *and flow (aggregate)* $\mathbb{F}_1$ *with arrival curves* $\alpha^{\mathbb{F}_0}$ *and* $\alpha^{\mathbb{F}_1}$*, respectively. Then* $\mathbb{F}_1$*'s worst-case residual service share under arbitrary multiplexing at* s*, i.e., its left-over (simple) service curve at* s*, is*

$$\beta_s^{\text{l.o.}\mathbb{F}_1} = \beta_s \ominus \alpha^{\mathbb{F}_0}$$

where $(\beta \ominus \alpha)(d) := \sup_{0 \leq u \leq d} \{(\beta - \alpha)(u)\}$ *denotes the non-decreasing upper closure of* $(\beta - \alpha)(d)$*.*

For arbitrary multiplexing servers, the result of this subtraction is not necessarily strict, i.e., consecutive left-over operations are not permitted. However, aggregating arrival curves before subtraction from service is backed by the NC theory. In the analysis of different multiplexing disciplines, namely SP or FIFO, consecutive application of their respective left-over service curve operation is permitted. For SP, the left-over service curve derivation is $\beta_s \ominus \alpha^{\mathbb{F}_0}$ as well [11]. Theorem 2.3 can also be used for FIFO multiplexing servers as arbitrary multiplexing is more general. However, there is a left-over service curve derivation for FIFO servers: $\beta_{s,\theta}^{\text{l.o.}\mathbb{F}_1}(d) = \left[\beta_s(d) - \alpha_s^{x(\mathbb{F}_1)}(d - \theta)\right]^+ \cdot 1_{\{d > \theta\}}$, $[\cdot]^+ = \max\{\cdot, 0\}$, $\theta \geq 0$ [50].

2.2 Analysis Principles

Application of NC evolved from the analysis of single servers crossed by a single flow to the analysis of feed-forward networks of servers crossed by multiple flows that can be entangled arbitrarily. During this evolution, two fundamental principles have been identified. They express behavior observable in a realistic network and thus should be implemented in the NC analysis in order to improve the quality of bounds, i.e., accuracy of delay and backlog bounds.

Pay Burst Only Once An analyzed flow of interest (foi) possess a certain worst-case burstiness. In the NC description, this manifests in its arrival curve's burst term $\lim_{d \to 0^+} \alpha(d)$. Due to the FIFO per µFlow property, data units within the foi cannot overtake each other. I.e., the burst cannot be delayed by other data of the foi and occur again in an multiplexing-enforced store-and-forward behavior. Burstiness levels off at the first server and thus the foi's burst term should only appear once in the end-to-end performance bound derivation for its entire path. This principle is called Pay Bursts Only Once (PBOO).

Pay Multiplexing Only Once The service available to the foi depends on its cross-traffic. Deriving the left-over service thus requires arrival curves for cross-flows, so called cross-traffic arrival bounds. In case cross-flows share multiple consecutive hops with the foi, their arrival bound burstiness should not impact the delay bound derivation too often – the same reasoning as with the PBOO principle applies. I.e., from the foi's perspective, multiplexing should only be paid for once and therefore this principle is called Pay Multiplexing Only Once (PMOO).

2.3 Algebraic Analyses

An algebraic NC (algNC) analysis takes the network and compiles it into an equation – either for bounding of a specific foi's performance characteristics or those of a server. These equations consist of (min,+)-algebraic operations and they must retain the worst case established by the network description in order to derive valid bounds. In this section, we provide the foremost NC analyses. They are the result of effort towards implementing the above principles. Each analysis defines a different order of operations for its respective equation. The following analyses all assume lossless transmissions and infinite buffer space.

Figure 2.2 will serve as a running example in this section. We chose a tree network to circumvent a comprehensive transformation of the device graph to the server graph. In trees, there is only one potential next hop and thus each network device has only a single output buffer. Note, that Figure 2.2 in fact depicts a sink-tree network. However s_6 is not the sink of the device graph but corresponds to the device that is directly connected to it. At the sink-device, there is no output buffer to cross and thus it does not appear in the network as a sink-server after s_6.

2.3.1 Total Flow Analysis

The Total Flow Analysis (TFA) [26] directly applies the basic results from Definition 2.6, aggregation and deconvolution, as well as Theorem 2.1: It takes the totality of flows

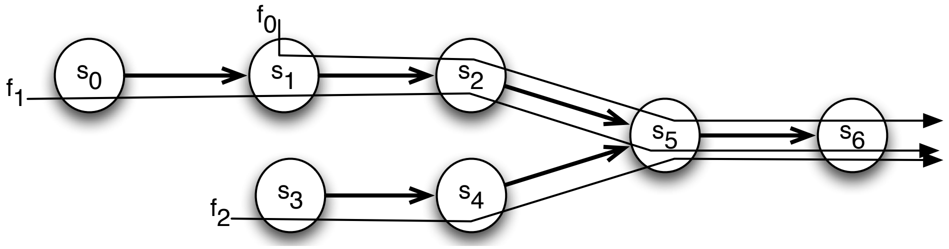


Figure 2.2: Tree network, i.e., server graph with flows, serving as a running example.

(a flow aggregate) present at a server and the server's service curve to bound delay and backlog. I.e., the TFA operates from a server's point of a view by aggregating all incoming traffic. The delay bound is thus valid for all flows crossing the server and the backlog bound denotes the server's buffer requirement for serving them.

Assume server s_i is crossed by the flow aggregate $F(s_i)$ and let α_{s_i} be the aggregated arrival curve of all flows. Then, we get the server-local TFA bounds:

$$D_{s_i} = \begin{cases} h\left(\alpha_{s_i}, \beta_{s_i}\right) & \text{if } |F(s_i)| = 1 \text{ (FIFO per µFlow)} \\ bp\left(\alpha_{s_i}, \beta_{s_i}\right) & \text{otherwise} \end{cases}, \qquad B_{s_i} = v\left(\alpha_{s_i}, \beta_{s_i}\right).$$

The delay bounds on the foi's path $P(\text{foi})$, from its source server to its destination server, can be used to derive a flow delay bound:

$$D^{\text{TFA}}_{P(\text{foi})} = \sum_{s_i \in P(\text{foi})} D_{s_i}$$

For instance, the TFA delay bound derivation for flow f_1 in Figure 2.2 proceeds as follows:

$$\begin{aligned}
D^{\text{TFA}}_{P(f_1)} &= D_{s_0} + D_{s_1} + D_{s_2} + D_{s_5} + D_{s_6} \\
&= h\Big(\alpha^{f_1}_{s_0}, \beta_{s_0}\Big) + bp\Big(\alpha^{[f_1,f_0]}_{s_1}, \beta_{s_1}\Big) + bp\Big(\alpha^{[f_1,f_0]}_{s_2}, \beta_{s_2}\Big) \\
&\quad + bp\Big(\alpha^{[f_1,f_0,f_2]}_{s_5}, \beta_{s_5}\Big) + bp\Big(\alpha^{[f_1,f_0,f_2]}_{s_6}, \beta_{s_6}\Big) \\
&= \ldots \\
&= h\Big(\alpha^{f_1}, \beta_{s_0}\Big) \\
&\quad + bp\Big(\Big(\alpha^{f_1} \oslash \beta_{s_0}\Big) + \alpha^{f_0}, \beta_{s_1}\Big) \\
&\quad + bp\Big(\Big(\Big(\alpha^{f_1} \oslash \beta_{s_0}\Big) + \alpha^{f_0}\Big) \oslash \beta_{s_1}, \beta_{s_2}\Big) \\
&\quad + bp\Big(\Big(\Big(\Big(\alpha^{f_1} \oslash \beta_{s_0}\Big) + \alpha^{f_0}\Big) \oslash \beta_{s_1}\Big) \oslash \beta_{s_2}\Big) + \Big(\Big(\alpha^{f_2} \oslash \beta_{s_3}\Big) \oslash \beta_{s_4}\Big), \beta_{s_5}\Big) \\
&\quad + bp\Big(\Big(\Big(\Big(\Big(\Big(\alpha^{f_1} \oslash \beta_{s_0}\Big) + \alpha^{f_0}\Big) \oslash \beta_{s_1}\Big) \oslash \beta_{s_2}\Big) + \Big(\Big(\alpha^{f_2} \oslash \beta_{s_3}\Big) \oslash \beta_{s_4}\Big)\Big) \oslash \beta_{s_5}, \beta_{s_6}\Big)
\end{aligned}$$

Although the TFA is based on per-server bounds, replacing a single server yields large recomputation effort. The output boundings (deconvolution operations $\oslash$) required to derive all arrivals at the servers create dependencies between them. For instance, replacing β_{s_0} results in a complete recomputation of the above $D^{\mathrm{TFA}}_{P(f_1)}$. Moreover, accuracy is compromised by the occurrences of (entire) arrival curves α^{f_0}, α^{f_1} and α^{f_2} at every server they cross. Each arrival curve contributes a burst term to the server-local delay derivation – neither the PBOO nor the PMOO principle are implemented in the TFA.

2.3.2 Separate Flow Analysis

The Separate Flow Analysis (SFA) departs from computing per-server bounds that are valid for the totality of flows. Instead, SFA derives an *end-to-end* left-over service curve for the foi and uses it to bound only the foi's performance characteristics. For NC, this step from server analysis to end-to-end flow analysis constitutes a big evolutional leap forward. Building the delay analysis around the foi shifted its view directly to the tandem of servers on its path $P(\mathrm{foi})$ – the SFA is a *tandem analysis*. Separating the foi allows NC to make use of its FIFO per µFlow property, i.e., its end-to-end delay is always bounded with the horizontal deviation $h\left(\alpha,\beta\right)$. In [74], it was shown that this method strictly outperforms TFA with the $bp\left(\alpha,\beta\right)$. Note, however, that the backlog bound, $v\left(\alpha,\beta\right)$, has a different semantic in the SFA. It does not bound the backlog at a server but the foi's data in transit.

The SFA is a straight-forward, server-by-server application of Theorems 2.1, 2.3 and 2.2 [50]: it bounds cross-traffic arrivals (cf. TFA, not depicted), subtracts them from the servers' service curves, concatenates the resulting left-over service curves and finally bounds the foi's performance characteristics.

$$\beta^{\mathrm{l.o.foi}}_{s_i} = \beta_{s_i} \ominus \alpha^{x(\mathrm{foi})}_{s_i}, \qquad \beta^{\mathrm{l.o.}^{\mathrm{SFA}}\mathrm{foi}}_{P(\mathrm{foi})} = \bigotimes_{s_i \in P(\mathrm{foi})} \beta^{\mathrm{l.o.foi}}_{s_i}.$$

$$D^{\mathrm{foi}} = h\left(\alpha^{\mathrm{foi}}, \beta^{\mathrm{l.o.}^{\mathrm{SFA}}\mathrm{foi}}_{P(\mathrm{foi})}\right), \qquad B^{\mathrm{foi}} = v\left(\alpha^{\mathrm{foi}}, \beta^{\mathrm{l.o.}^{\mathrm{SFA}}\mathrm{foi}}_{P(\mathrm{foi})}\right).$$

Executing the operations in this order can be directly proven to result in a valid end-to-end left-over service curve. Bounding the delay with the horizontal deviation instead of the maximum backlogged period size relaxes the strict service curve requirement.

Using a single, end-to-end left-over service curve, the foi's arrival curve α^{foi} appears only once in the equation for delay bound computation. I.e., the SFA implements the PBOO principle. In case cross-flows are present at multiple consecutive hops, their arrivals appear multiple times. We illustrate this problem with the $\beta^{\mathrm{l.o.}^{\mathrm{SFA}}f_1}_{P(f_1)}$-computation for Figure 2.2:

$$
\begin{aligned}
\beta_{P(f_1)}^{\mathrm{l.o.}^{\mathrm{SFA}} f_1} &= \beta_{s_0}^{\mathrm{l.o.} f_1} \otimes \beta_{s_1}^{\mathrm{l.o.} f_1} \otimes \beta_{s_2}^{\mathrm{l.o.} f_1} \otimes \beta_{s_5}^{\mathrm{l.o.} f_1} \otimes \beta_{s_6}^{\mathrm{l.o.} f_1} \\
&= \beta_{s_0} \otimes \Big(\beta_{s_1} \ominus \alpha^{f_0}\Big) \otimes \Big(\beta_{s_2} \ominus \alpha_{s_2}^{f_0}\Big) \otimes \Big(\beta_{s_5} \ominus \alpha_{s_5}^{[f_0,f_2]}\Big) \otimes \Big(\beta_{s_6} \ominus \alpha_{s_6}^{[f_0,f_2]}\Big) \\
&= \ldots \\
&= \beta_{s_0} \otimes \Big(\beta_{s_1} \ominus \alpha^{f_0}\Big) \otimes \Big(\beta_{s_2} \ominus \Big(\alpha^{f_0} \oslash \beta_{s_1}\Big)\Big) \\
&\quad \otimes \Big(\beta_{s_5} \ominus \Big(\Big(\Big(\alpha^{f_0} \oslash \beta_{s_1}\Big) \oslash \beta_{s_2}\Big) + \Big(\Big(\alpha^{f_2} \oslash \beta_{s_3}\Big) \oslash \beta_{s_4}\Big)\Big)\Big) \\
&\quad \otimes \Big(\beta_{s_6} \ominus \Big(\Big(\Big(\Big(\alpha^{f_0} \oslash \beta_{s_1}\Big) \oslash \beta_{s_2}\Big) + \Big(\Big(\alpha^{f_2} \oslash \beta_{s_3}\Big) \oslash \beta_{s_4}\Big)\Big)\Big) \oslash \beta_{s_5}\Big)
\end{aligned}
$$

In this SFA equation, α^{f_1} does not appear, yet, α^{f_0} and α^{f_2} are found multiple times. Every occurrence adds the respective cross-flow's burst term; the derivation is not able to capture how these level off in a realistic system. From f_1's point of view, multiplexing with cross-traffic is paid for multiple times.

The presented $\beta_{P(f_1)}^{\mathrm{l.o.}^{\mathrm{SFA}} f_1}$ is also a bound on the left-over service for SP multiplexing and FIFO multiplexing tandems of servers because the foi is assumed to have lowest priority among all flows. A more accurate SP left-over service curve can be derived with a simple adaptation: Flows of lower priority than f_1 need not be subtracted from the service curve. For FIFO multiplexing, the FIFO left-over service curve can be used instead of the arbitrary multiplexing one. The equation will have multiple parameters θ_i, one for each server s_i, and a subsequent optimization step is required. In contrast to the arbitrary multiplexing SFA, per-server knowledge is not sufficient to derive the end-to-end left-over service curve [51]. For non-FIFO per µFlow analysis, implementing the PBOO principle requires a different notion of service curve that guarantees an upper bound on the dwell period of a data unit traversing a tandem of servers [70].

2.3.3 Pay Multiplexing Only Once Analysis

The SFA's $\beta_{P(f_1)}^{\mathrm{l.o.}^{\mathrm{SFA}} f_1}$ derivation reveals that cross-traffic arrival curves appear at every server they need to be subtracted. Therefore, implementing the PMOO principle follows the idea to convolve the analyzed tandem of servers before subtracting the cross-traffic arrivals[3]. This order of operations allows the NC analysis to capture the leveling off of cross-flow bursts on the subpath shared with the analyzed foi [30]. The PMOO analysis (PMOOA) implements this order of operations. In this thesis, we use two different variants of the PMOOA, depending on the interference pattern of cross-traffic.

---

[3] Application of algebraic operations in this order is not backed by NC theory. Service curve strictness is required for subtraction but the result of convolution is not provably strict [31]. Yet, [74] provides a proof that the resulting curve is indeed a valid lower bound for the tight left-over service curve.

Definition 2.7. (Flow Nesting) Let $\mathcal{T}$ be a tandem of servers and f_a and f_b two flows crossing some servers of this tandem. Then, flows are related in one of these ways:

- Flow f_a is nested into f_b on $\mathcal{T}$, $f_a \preceq_{\mathcal{T}} f_b$, iff all servers f_a consecutively crosses on $\mathcal{T}$ are also consecutively crossed by f_b.
- Flow f_b is nested into f_a on $\mathcal{T}$, $f_b \preceq_{\mathcal{T}} f_a$, iff all servers f_b consecutively crosses on $\mathcal{T}$ are also consecutively crossed by f_a.
- Flows f_a and f_b are not related on $\mathcal{T}$ iff they do not share any servers on this tandem.
- Flows f_a and f_b overlap on $\mathcal{T}$ if neither of the above is true.

Note, that servers can only be crossed in one direction due to the directed links in the server graph and that, by definition, all cross-flows of an analyzed foi are nested into its path, i.e., $\mathcal{T} = P(\text{foi})$.

Definition 2.8. (Interference Pattern [52]) Let foi be the flow of interest and let $x(\text{foi})$ be the set of its cross-flows. The foi's *cross-traffic interference pattern* is called nested if, on the foi's path $P(\text{foi})$, there is either a nesting relation between all pairs of flows in $x(\text{foi})$ or no relation at all. If there are at least two cross-flows that overlap on $P(\text{foi})$, the entire cross-traffic interference pattern is called non-nested.

The first PMOOA variant is exclusively applicable to tandems with a nested cross-traffic interference pattern. The second one defines a new operation, a left-over service curve computation for an entire tandem, that is applicable to tandems with non-nested interference patterns. However, it requires that arrival curves are from $\mathcal{F}_{\text{mTB}}$ and service curves are from $\mathcal{F}_{\text{mRL}}$.

The PMOOA for Nested Interference This interference pattern guarantees a hierarchical nesting relation between all flows on $P(\text{foi})$. This relation can be expressed with a nesting tree with the foi at its root [52, 53]. The tree defines the order of algebraic NC operations to be applied: Starting from the tree's leaf nodes, i.e., the flows without others nested into them, flow paths are convolved and the according flow traversing it is subtracted. These two steps result in (not necessarily end-to-end) left-over service curves for flows on the next level in the nesting tree. Convolution and subtraction are repeated, level for level in the nesting tree, until only the foi and its end-to-end left-over service curve remain.

The nesting tree for the network of Figure 2.2 is $f_2 \preceq_{P(f_1)} f_0 \preceq_{P(f_1)} f_1$. It results in

$$\begin{aligned}
\beta_{P(f_1)}^{\text{l.o.}^{\text{PMOO}} f_1} &= \beta_{\langle s_0, s_1, s_2, s_5, s_6 \rangle}^{\text{l.o.}^{\text{PMOO}} f_1} \\
&= \left(\left(\left((\beta_{s_5} \otimes \beta_{s_6}) \ominus \alpha_{s_5}^{f_2} \right) \otimes (\beta_{s_1} \otimes \beta_{s_2}) \right) \ominus \alpha^{f_0} \right) \otimes \beta_{s_0} \\
&= \left(\left(\left((\beta_{s_5} \otimes \beta_{s_6}) \ominus (\alpha^{f_2} \oslash \beta_{s_3} \oslash \beta_{s_4}) \right) \otimes (\beta_{s_1} \otimes \beta_{s_2}) \right) \ominus \alpha^{f_0} \right) \otimes \beta_{s_0}
\end{aligned}$$

and f_1's performance bounds are, similar to the SFA,

$$D^{f_1} = h\left(\alpha^{f_1}, \beta_{P(f_1)}^{\text{l.o.}^{\text{PMOO}} f_1} \right), \qquad B^{f_1} = v\left(\alpha^{f_1}, \beta_{P(f_1)}^{\text{l.o.}^{\text{PMOO}} f_1} \right).$$

For the analysis of nested FIFO-multiplexing tandems, this procedure constitutes the fundamental aspect of the least upper delay bound (LUDB) analysis [3].

The PMOOA Left-over Service Curve for non-Nested Interference The second variant of the PMOOA addresses the problem of non-nested interference patterns [75]. It provides a single-step left-over service curve derivation $\beta_{P(\text{foi})}^{\text{l.o.}^{\text{PMOO}} \text{foi}}$. This non-nested-PMOO operation has an additional requirement on the shape of curves: The strict service curves must belong to the class of multi-rate-latency curves $\mathcal{F}_{\text{mRL}}$ and arrival curves must be from the class of multi-token-bucket curves $\mathcal{F}_{\text{mTB}}$. Moreover, for the concrete derivation of $\beta_{P(f_1)}^{\text{l.o.}^{\text{PMOO}} f_1}$, we need to maintain distinguishable partial curves after the decomposition into partial curves from $\mathcal{F}_{\text{mRL}}$ and $\mathcal{F}_{\text{mTB}}$:

$$\beta_{s_0} = \bigvee_{h=1}^{n_h} \beta_{R_{s_0,h}, T_{s_0,h}} \quad , \quad \beta_{s_1} = \bigvee_{i=1}^{n_i} \beta_{R_{s_1,i}, T_{s_1,i}} \quad ,$$

$$\beta_{s_2} = \bigvee_{j=1}^{n_j} \beta_{R_{s_2,j}, T_{s_2,j}} \quad , \quad \beta_{s_5} = \bigvee_{k=1}^{n_k} \beta_{R_{s_5,k}, T_{s_5,k}} \quad ,$$

$$\beta_{s_6} = \bigvee_{l=1}^{n_l} \beta_{R_{s_6,l}, T_{s_6,l}} \quad ,$$

$$\alpha_{s_1}^{f_0} = \alpha^{f_0} = \bigwedge_{p=1}^{n_p} \gamma_{r_{s_1,p}^{f_0}, b_{s_1,p}^{f_0}} \quad , \quad \alpha_{s_5}^{f_2} = \left(\alpha^{f_2} \oslash \beta_{s_3} \right) \oslash \beta_{s_4} = \bigwedge_{q=1}^{n_q} \gamma_{r_{s_5,q}^{f_2}, b_{s_5,q}^{f_2}} \quad .$$

Then we get the following left-over service curve:

$$\begin{aligned}
\beta_{P(f_1)}^{\text{l.o.}^{\text{PMOO}} f_1} &= \beta_{\langle s_0, s_1, s_2, s_5, s_6 \rangle}^{\text{l.o.}^{\text{PMOO}} f_1} \\
&= \bigvee_{h=1}^{n_h} \bigvee_{i=1}^{n_i} \bigvee_{j=1}^{n_j} \bigvee_{k=1}^{n_k} \bigvee_{l=1}^{n_l} \bigvee_{p=1}^{n_p} \bigvee_{q=1}^{n_q} \beta_{R_{h,i,j,k,l,p,q}^{\text{l.o.}}, T_{h,i,j,k,l,p,q}^{\text{l.o.}}}
\end{aligned}$$

with

$$\begin{aligned} R^{\text{l.o.}}_{h,i,j,k,l,p,q} &= R_{s_0,h} \wedge \left(R_{s_1,i} - r^{f_0}_{s_1,p}\right) \wedge \left(R_{s_2,j} - r^{f_0}_{s_1,p}\right) \\ &\quad \wedge \left(R_{s_5,k} - r^{f_0}_{s_1,p} - r^{f_2}_{s_5,q}\right) \wedge \left(R_{s_2,l} - r^{f_0}_{s_1,p} - r^{f_2}_{s_5,q}\right) \end{aligned}$$

and

$$\begin{aligned} T^{\text{l.o.}}_{h,i,j,k,l,p,q} &= T_{s_0,h} + T_{s_1,i} + T_{s_2,j} + T_{s_5,k} + T_{s_6,l} && (2.1) \\ &\quad + \frac{b^{f_0}_{s_1,p} + b^{f_2}_{s_5,q}}{R^{\text{l.o.}}_{h,i,j,k,l,p,q}} && (2.2) \\ &\quad + \frac{r^{f_0}_{s_1,p} \cdot (T_{s_1,i} + T_{s_2,j} + T_{s_5,k} + T_{s_6,l}) + r^{f_2}_{s_5,q} \cdot (T_{s_5,k} + T_{s_6,l})}{R^{\text{l.o.}}_{h,i,j,k,l,p,q}} && (2.3) \end{aligned}$$

The three parts of the left-over service curve's latency are distinguishable on the right-hand side of $T^{\text{l.o.}}_{h,i,j,k,l,p,q}$:

- The original latencies of the crossed servers (Term 2.1),
- the additional latency due to first working off bursts of cross-traffic (Term 2.2), and
- the added latency for working off data that was queued while waiting for the servers' latency (Term 2.3).

Only the second part holds the cross-flows' burst terms and by taking the minimum over the combinations of token buckets and rate latencies, cross-traffic burstiness is paid for only once in the overall derivation. The analyzed flow's performance bounds are, again, derived with this service curve (see nested PMOOA).

For FIFO multiplexing, a single-step derivation akin to the non-nested PMOOA does not exist. Instead, the analyzed tandem is cut into sub-tandems with nested interference patterns. Each sub-tandem's left-over service curve is computed with the nested PMOOA variant and they are convolved to an end-to-end left-over service curve. In case there are multiple alternatives to cut the tandem, all are tested and the best result, i.e., the least delay bound, is returned by the LUDB analysis [3]. This approach does not necessarily implement the PMOO principle to its fullest extent.

2.4 Optimization-based Analysis

The PMOOA was considered strictly superior to the SFA w.r.t. delay bounding as it implements the eponymous principle for end-to-end analysis. However, [74] shows that the SFA can arbitrarily outperform the PMOOA. Due to the convolution's commutativity, PMOOA cannot benefit from large residual service at the end of the analyzed (convolved) tandem. This is expressed on the right-hand side the above equation, Terms 2.2 and 2.3, where the minimum end-to-end left-over rate is in the denominator. I.e., the per-server service rate is not available to the analysis.

Additionally, an optimization-based bounding method that implements the PMOO principle is provided in [74]. It transforms the NC network description to an optimization problem; departing from the algebraic methodology to overcome the problematic property of convolution. This method shares the PMOOA's requirement on arrival and service curves to be from $\mathcal{F}_{\text{mTB}}$ and $\mathcal{F}_{\text{mRL}}$, respectively, and derives a superior left-over service curve from these. Hence, [74] also proves that the PMOOA's $\beta_{P(\text{foi})}^{\text{l.o.}^{\text{PMOO}}\text{foi}}$ is valid despite its order of operations that cannot be directly proven to be resulting in a valid left-over service curve.

2.4.1 The Linear Programming Analysis

The seminal work on optimization-based NC inherited some crucial aspects from algNC. Most importantly, it composes tandem-local results when analyzing a feed-forward network. This approach is known to enforce worst-case assumptions in the analysis of feed-forward networks that may lead to a loss of result accuracy [53]. Optimization does, however, not require a compositional analysis. Based on these insights, an optimization-based, tight feed-forward analysis was proposed in [12]: the Linear Programming Analysis (LPA). Intuitively, LPA delay analysis proceeds as follows:

1. Starting from the foi's sink server, flows as well as their respective cross-flows are recursively traced towards their sources. For every link traversed backwards, the start of backlogged periods at the link's source and its destination are related. This step terminates as soon as all flows are traced to their sources. The result is a partial order where, for example, there is no given order between the starts of backlogged periods for servers in different branches of a tree (e.g., s_2 and s_4 in Figure 2.2).

2. The second step is to extend the partial order to the set of all compatible total orders. This procedure enumerates all potential entanglements of the start of backlogged periods at the network's servers. In the above example, the entanglement on distinct branches of the tree network are enumerated such that all outcomes are considered at the later servers s_5 and s_6. Special care must be taken of relations caused by rejoining flows [53].

3. This step transforms each total order to one linear program. The order between the start of backlogged periods as well as the NC network description (strict service

curves, arrival curves, non-decreasing curves, non-negativity, flow constraint) are used to derive the constraints of each linear program. Like most analyses, the LPA requires arrival curves from $\mathcal{F}_{\mathrm{mTB}}$ and service curves from $\mathcal{F}_{\mathrm{mRL}}$. They are decomposed as shown above; each token bucket and each rate latency is converted into one constraint.

4. The LPA's set of linear programs models all potential entanglements of flows as they gradually progress on their path through the network; not only the worst-case one(s). Therefore, all linear programs must be solved in the final step. The maximum among their solutions is a valid worst-case bound for the foi's delay. I.e., the LPA is an all-or-nothing analysis approach.

Step 2 constitutes the main drawback of the LPA: The amount of different linear programs and thus the computational complexity of the LPA grows possibly (super-)exponentially with the network size. The authors prove that their approach to obtain tight bounds is in fact NP-hard.

For other multiplexing disciplines, similar optimization-based NC analyses have been provided in the literature: SP feed-forward-networks [13], FIFO tandems [14] and FIFO feed-forward networks [15]. The latter one is most similar to the LPA, yet, it replaces step 2 with a different technique to encode multiplexing and demultiplexing of flows that lead to parallel paths. It introduces integer variables, i.e., it transforms the NC network description into a single mixed-integer linear program instead of a set of ordinary linear programs. There is no optimization-based analysis for tandems of servers that do not retain the FIFO per µFlow property.

3 Accuracy of Algebraic Network Calculus

This section presents the problem of flow segregation arising in algNC analysis of feed-forward networks. We provide two countermeasures to it: aggregate arrival bounding and arrival bound burstiness reduction. AlgNC can derive delay, backlog and output bounds for tandems of servers in a straight-forward fashion. For the analysis of feed-forward networks, we present the generic concepts to decompose the network into tandems and recompose tandem-local analysis results to the flow of interest's performance bounds.

Section 2.2 presented the state of the art in NC; work that was mainly dedicated to the step from a single server analysis to a tandem of servers. We have established the framework that allows a tandem analysis to compute performance bounds in general feed-forward networks. This framework, called the compositional feed-forward analysis (compFFA), is presented in Section 3.1. It defines how the tandem-local results are composed to a network-global analysis – a step not required in the LPA. Investigating the compFFA procedure allows us to identify the causes for inaccurate results in algNC's feed-forward analysis (Section 3.2). Improvements to the algNC are presented in Sections 3.3 and 3.4.

3.1 Compositional Analysis of Feed-forward Networks

The LPA's approach to the feed-forward analysis imposes considerable effort to attain a network-global view on flow scheduling and cross-traffic (de-)multiplexing, rendering it NP-hard (Section 2.4.1). The tandem analyses of algNC cannot follow this approach. For the analysis of a feed-forward network, they must follow a compositional divide-and-conquer approach, the compFFA. It consists of two steps [7, 8].

1. *Cross-traffic Arrival Bounding:*
   The first compFFA step abstracts from the feed-forward network to the foi's path – a tandem of servers that can be analyzed with one of the existing procedures. After this step, a bound on the worst-case shape of cross-flows is known at the locations of interference with the foi. Then, the following step need not consider the part of the network traversed by these flows nor the potentially complex interference patterns they are subject to (see Figure 3.1). In detail, this step proceeds as follows:
   a) Starting at the locations of interference with the foi, cross-flows are backtracked to their sources. This procedure derives the dependencies between the foi, its cross-flows, their cross-flows, etc., in a recursive fashion. A new instance of this sub-step is started for any cross-flow of the current cross-flow under consideration. Due to the network's feed-forward property, the recursion is guaranteed to terminate.

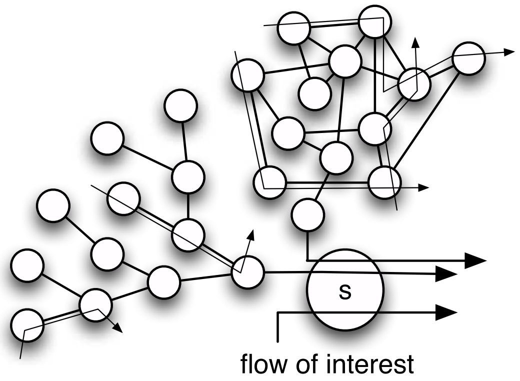


Figure 3.1: CompFFA: abstraction from a feed-forward network, possibly a tree, to the servers on the flow of interest's path.

   b) Next, the dependencies are converted into an equation, i.e., a sequence of algebraic operations, that capture the transformation of worst-case flow arrivals.
   c) Finally, the equations are solved to obtain the bounds on cross-traffic arrivals.

2. *Flow of Interest Performance Bounding:*
   The foi's end-to-end delay bound in the feed-forward network is derived with a less complex tandem analysis.

The second step of the compFFA procedure has seen much treatment in the literature (see Section 2.3 as well as [10, 31] for recent overviews). The first step of the feed-forward network analysis, bounding the cross-traffic, has so far been largely neglected. Most work starts directly with the tandem analysis or suggests to use straightforward techniques from basic NC results (more details are given in Section 3.3). An exception can be found in [29], where, for a single node under arbitrary multiplexing of several flows, tight output descriptions are derived for a single flow. However, when targeting a feed-forward network, we need to bound cross-flows that may have traversed several servers with potentially many other flows joining and leaving it. The literature neither provides an in-depth analysis of compFFA's shortcomings nor work to improve analysis accuracy through step 1. This thesis will focus on the first step of the compFFA as well as the interdependencies between both steps in order to improve the accuracy of algNC performance bounds.

3.2 Problems

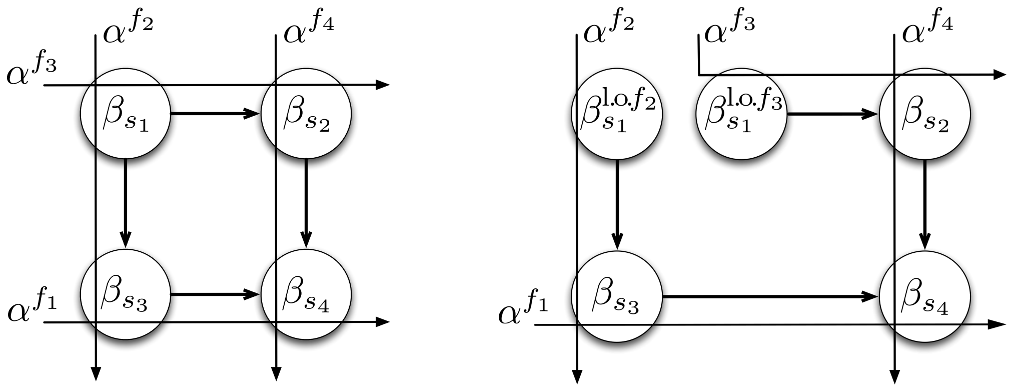
3.2.1 Problems Caused by Compositionality

The compFFA's compositionality causes a fundamental problem: Cross-flows arrivals at different locations of interference are bounded independently from each other with separate instances of compFFA step 1. In every instance, the flow under consideration is bounded with flow-local worst-case assumptions; the compositional analysis suffers from a *cross-flow segregation effect*. Each of the segregated flows computes a left-over service curve by considering any other traffic to interfere with it in the worst case. The cross-flow segregation negatively impacts compFFA if, during independent backtrackings of step 1a, servers are traversed multiple times. I.e., they appear in the equation of step 1b more than once, with different settings of flows to subtract (left-over service curve operation) and different flows for arrival bounding. Rejoining interference with the foi is an instance of this problem [53][4] that also motivates the LPA [12]. A similar instance of the problem is used for evaluating the LPA: the square network (see Figure 3.2a). It imposes segregation in a transitive fashion when analyzing flow f_1: The flows at server s_1 (service β_{s_1}) are in two separate backtracking instances, one started for at server s_3 and one started at server s_4 that, in turn, requires to bound f_2 at server s_2. I.e., at s_1, both flows assume worst-case mutual interference by a left-over service curve derivation with their respective flow-local worst-case assumptions. Service β_{s_1} is segregated into $\beta_{s_1}^{\text{l.o.}f_2} = \beta_{s_1} \ominus \alpha^{f_3}$ and $\beta_{s_1}^{\text{l.o.}f_3} = \beta_{s_1} \ominus \alpha^{f_2}$ (Figure 3.2b) – two left-over service curves that cannot be attained simultaneously in a realistic network. As a consequence, the result of compFFA step 1a, i.e., the internal model it creates, does not necessarily equal the server graph. We formulate this problem in a novel principle to be implemented by feed-forward NC analyses, similar to PBOO and PMOO (Section 2.2).

The Pay Segregation Only Once (PSOO) principle If the arrivals of two flows have to be bounded segregately in the compositional feed-forward analysis and these flows both cross the same server before interfering with the foi, then they should not be segregated in a way that imposes the worst-case mutual interference assumptions on both. In the algebraic analysis equation, segregated flows should not have to consider each other fully in their respective arrival bounding. Although this leads to a valid upper bound, the according behavior is not attainable by a realistic system and thus the eventual performance bound cannot be tight. Segregation of cross-flows should only be paid for once by the ensemble of the two flows.

---

[4] In the earlier work of [73] "eliminate rejoining interfering flows" means that the analysis internally creates a new flow for every location of interference with the flow of interest. The arrivals of these flows are bounded with compFFA step 1, i.e., the cross-flow segregation problem persists.



(a) The Square Network.

(b) CompFFA-internal model of the network.

Figure 3.2: Cross-flow segregation in the square network.

The LPA is not compositional, it is the only NC analysis implementing the PSOO principle in addition to PBOO and PMOO.

3.2.2 Problems Caused by Tandem Analyses

We found an interdependency between compFFA step 2 and compFFA step 1 that can enforce segregation of cross-flows. We derived the minimal network setting that exhibits the problem (Figure 3.3a). Interestingly, it is the non-nested tandem as already used in [74] to motivate the use of optimization, yet, flows take different roles. The foi does not cross the entire tandem, instead, we analyze the flow crossing only the two servers at the end of the tandem.

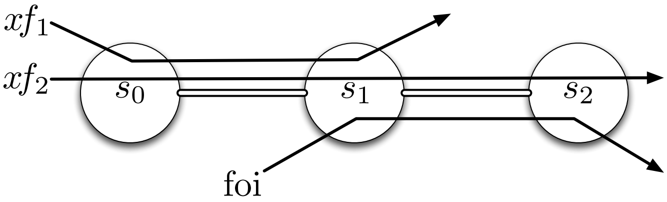
Bounding the foi's end-to-end delay depends on the choice of tandem analysis applied to it[5]. I.e., the actual decomposition into left-over service curve and output bound operations in step 1 depends on the analysis chosen for in step 2.

- The SFA first decomposes the foi's paths into single servers to apply Theorems 2.3 and 2.2 (Figure 3.3c). Thus, it starts an cross-traffic arrival bounding instance for every server.
- The PMOOA does not decompose the foi's path into smaller units of operation than this tandem itself. I.e., it tries to operate on longest tandems possible (Figure 3.3b).

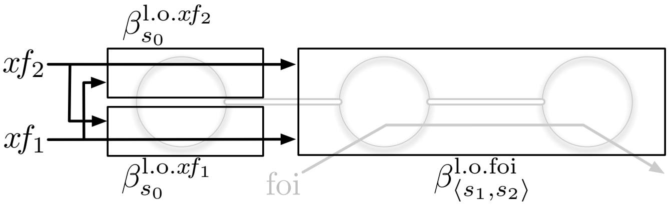
We have identified crucial problems that arise from either of these two alternatives of algNC. They enforce a segregation of cross-flows and with their distinct cause, they can occur in addition to the segregation problem illustrated with the square network.

---

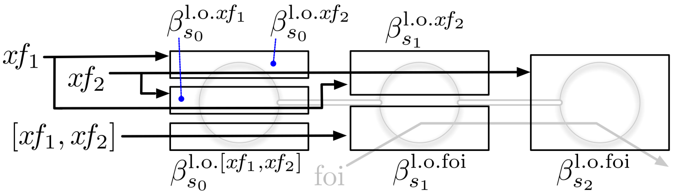
[5] We will not consider the TFA for delay analysis as it will result in strictly inferior bounds.



(a) Minimal network suffering from composition penalty.


$\beta_{s_0}^{\text{l.o.}xf_2}$
xf_2
xf_1
$\beta_{s_0}^{\text{l.o.}xf_1}$
foi
$\beta_{\langle s_1,s_2\rangle}^{\text{l.o.foi}}$

(b) Figure 3.3a's PMOOA-internal model.


$\beta_{s_0}^{\text{l.o.}xf_1}$
$\beta_{s_0}^{\text{l.o.}xf_2}$
$\beta_{s_1}^{\text{l.o.}xf_2}$
xf_1
xf_2
$[xf_1, xf_2]$
$\beta_{s_0}^{\text{l.o.}[xf_1,xf_2]}$
foi
$\beta_{s_1}^{\text{l.o.foi}}$
$\beta_{s_2}^{\text{l.o.foi}}$

(c) Figure 3.3a's SFA-internal model.

Figure 3.3: The minimal network imposing a PSOO principle violation due to the tandem analysis applied in compFFA step 2.
Interpretation: Boxes depict tandems for $\beta^{\text{l.o.}}$ derivation and arrows depict flows. Flows pointing at a box are subtracted from β and crossing a box means using the $\beta^{\text{l.o.}}$ for output bounding.

Cross-flow Segregation Enforced by Subpath Sharing Considerations The PMOOA was constructed to consider shared subpaths between the foi and its cross-flows. It demands a distinct arrival curve for each cross-flow (aggregate) sharing a specific subpath. I.e., if two cross-flows xf_1 and xf_2 interfere on two different subpaths, they require segregate arrival boundings [9]. This procedure causes problems in case these cross-flows share hops apart from the foi's path – similar to the known problem cause by the compFFA procedure. There, they cannot be aggregated; they must be considered mutual interference. Figure 3.3b illustrates this problem. Cross-flows xf_1 and xf_2 fulfill the criteria for segregated arrival bounding and their respective left-over service curves at s_0 are $\beta_{s_0}^{\text{l.o.}xf_1} = \beta_{s_0} \ominus \alpha^{xf_2}$ and $\beta_{s_0}^{\text{l.o.}xf_2} = \beta_{s_0} \ominus \alpha^{xf_1}$. The cross-traffic arrival bound derivation will have more than one cross-traffic burstiness for each of the interfering flows xf_1 and xf_2, the PSOO principle is violated. Both of the PMOOA variants presented in

Section 2.3.3 suffer from this problem.

Cross-flow Segregation Enforced by Tandem Decomposition The SFA can have various degrees of aggregation and separation within its equation, i.e, its internal model of the network. The SFA proceeds server-by-server; in terms of the PMOOA's subpath sharing point of view, it can aggregate all cross-flows sharing a single hop on the foi's path. Consider Figure 3.3a's network again: On the foi's first hop, s_1, SFA allows to aggregately bound the arrivals of xf_1 and xf_2 in a single instance of compFFA step 1. Then, the left-over service curve of the cross-flow aggregate $[xf_1, xf_1]$ crossing server s_0 is $\beta_{s_0}^{\text{l.o.}[xf_1,xf_1]} = \beta_{s_0}$. In this case, aggregation supersedes segregation and thus no explicit implementation of the PSOO principle is required. However, the SFA's decomposition into individual servers also requires an arrival bounding instance for xf_2 at s_2. Here, mutual interference between xf_1 and xf_2 has to be assumed. At s_0, we get the same $\beta_{s_0}^{\text{l.o.}xf_1}$ and $\beta_{s_0}^{\text{l.o.}xf_2}$ as in the PMOOA – the PSOO principle must be violated to retain the worst case. At s_1, we additionally get $\beta_{s_1}^{\text{l.o.}xf_2} = \beta_{s_1} \ominus \alpha_{s_1}^{xf_1}$, i.e., another burst term appears in the derivation due to violating the PMOO principle.

Decisions on Incomplete Knowledge In [74], the authors show that knowledge of the crossed servers' sequence is lost in the PMOOA's tandem analysis. The SFA, in contrast, retains this knowledge due to its server-by-server procedure. This allows the SFA to outperform the PMOOA on tandems where $\beta^{\text{l.o.}}$ curves get considerably faster towards the end. In feed-forward networks, either applying the SFA or the PMOOA is the first decision to take. That means, the decision does not have the $\beta^{\text{l.o.}}$ curves at its disposal – they remain unknown until the first step of the compFFA resulted in the cross-traffic arrival bounds. Thus, eventually concluding the analysis can exploit the sequence of servers triggers another feed-forward analysis of the network.

We summarize all the above problems in the *composition penalty* of algNC.

3.3 Derivation of Cross-traffic Arrival Bounds

We identified segregation of flows to cause inaccuracy in the algNC analysis. In this section, we present a new, recursive algorithm implementing an arrival bounding for compFFA step 1 that maximizes aggregation of flows. Thus, we reduce the composition penalty of compFFA's step 1. Unaware of the drawbacks of segregation, the literature suggest to execute SFA with a cross-traffic arrival bounding that applies maximum segregation [10]. It operates on longest possible tandems. We prove that our approach outperforms the one of the literature if (min,+)-deconvolution distributes over addition.

3.3.1 Segregated Cross-flow Arrival Bounding

NC analyses previously focused on improved end-to-end view on the tandem in the second compFFA step in order to tighten the delay bound. The generic way to compute performance bounds in feed-forward networks [10] is in line with this approach. It derives the SFA-like cross-traffic arrival bound for each flow at each system. Cross-flows are segregated from each other such that every cross-flow can be analyzed with a SFA on a maximum length tandem, i.e., source-to-interference location. This cross-flow bounding implements the PBOO principle for the segregated flows. Therefore, we call this procedure to decompose a feed-forward network into a sequence of tandem analyses the segregated PBOO arrival bounding, segrPBOOAB.

Figure 3.4 illustrates this method: At both servers crossed by the foi, s_1 and s_2, a new instance of compFFA step 1 is started for every cross-flow. The enforced segregation results in a violation of the PSOO principle:

$$\beta_{s_1}^{\text{l.o.foi}} = \beta_{s_1} \ominus \left(\alpha_{s_1}^{xf_1} + \alpha_{s_1}^{xf_2}\right) = \beta_{s_1} \ominus \left(\left(\alpha^{xf_1} \oslash \left(\beta_{s_0} \ominus \alpha^{xf_2}\right)\right) + \left(\alpha^{xf_2} \oslash \left(\beta_{s_0} \ominus \alpha^{xf_1}\right)\right)\right)$$

The arrival bounding at s_1 is equal to Figure 3.3b's derivation; the segrPBOOAB suffers from the same problem as the analysis considering subpath sharing. Moreover, at s_2 it suffers from the decomposition of the analyzed tandem into server-by-server left-over derivations and another cross-flow segregation at s_0 (cf. Figure 3.3c). Next, we present our alternative cross-traffic arrival bounding for the compFFA.

3.3.2 Aggregate Cross-traffic Arrival Bounding

In our alternative arrival bounding for compFFA step 1, we prioritize maximum aggregation of cross-flows over the analysis of maximum length tandems. We apply Theorem 2.1, output bound, to cross-traffic aggregates – moving along the topology, starting from the location of interference with the foi and terminating at the sources of cross-flows. Thus, aggregation results in a different decomposition of the network into tandems than the procedure of Section 3.3.1. It also reduces the PSOO violations by circumventing segregation as much as possible. In order to benchmark this change against the segrPBOOAB, we implement the PBOO principle on the tandems shared by all flows in a cross-traffic aggregate. We call this procedure the aggregate PBOO arrival bounding, aggrPBOOAB.

The derivation of the left-over service curve for s_1 in Figure 3.3a's network changes to (see Figure 3.3c):

$$\beta_{s_1}^{\text{l.o.foi}} = \beta_{s_1} \ominus \alpha_{s_1}^{[xf_1,xf_2]} = \beta_{s_1} \ominus \left(\left(\alpha^{xf_1} + \alpha^{xf_2}\right) \oslash \beta_{s_0}\right)$$

The crucial difference is located at s_0. As both of the cross-flows, xf_1 and xf_2, share this

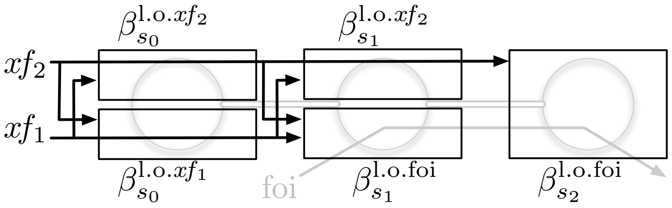

$\beta_{s_0}^{\text{l.o.}xf_2}$
$\beta_{s_1}^{\text{l.o.}xf_2}$
xf_2
xf_1
$\beta_{s_0}^{\text{l.o.}xf_1}$
foi
$\beta_{s_1}^{\text{l.o.foi}}$
$\beta_{s_2}^{\text{l.o.foi}}$

Figure 3.4: Figure 3.3a's SFA-internal model as presented in [10]. In contrast to Figure 3.3c, it does not aggregate cross-traffic arrivals at server s_1.

server, their output can be bounded aggregately. Information of per-cross-flow arrivals, i.e., a segregated bounding, is not necessary and thus the PSOO principle need not be implemented.

The aggrPBOOAB is not as straight-forward as the segrPBOOAB. Therefore, we developed the principle to maximize aggression into an algorithm for compFFA step 1. It integrates the three distinguishable substeps, backtracking, derivation of the analysis equation and solving it, into a recursive solution for efficient computation. This algorithm was implemented in the DiscoDNC tool [7].

Algorithm 3.1. (Recursive aggrPBOOAB) The foi's cross-traffic arrival bound at server s is derived as follows. Starting from s, paths of cross-flows are backtracked via links; each link l connects a source server l^{src} and a destination l^{dest}. The function $dest(s)$ returns the set of links whose destination is s, $x\left(\text{foi}, l\right) = F\left(l^{\text{src}}\right) \cap F\left(l^{\text{dest}}\right) \cap x\left(\text{foi}\right)$ returns the cross-flows of foi on l and for a set of links $\mathbb{L}$ we define $x\left(\text{foi}, \mathbb{L}\right) = \bigcap_{l \in \mathbb{L}} x\left(\text{foi}, l\right)$. We get, for instance, $x\left(x\left(\text{foi}, l\right)\right) = \left(F\left(l^{\text{src}}\right) \cap F\left(l^{\text{dest}}\right)\right) \setminus x\left(\text{foi}, l\right)$. Note, that flow sets might be restricted by location, e.g., $\alpha_{s_x}^{x(\text{foi})} = \alpha_{s_x}^{x(\text{foi}) \cap F(s_x)}$, and that in an expression $x\left(\text{foi}, l\right)$, the foi need not cross l.

$$\begin{aligned}
\alpha_s^{x(\text{foi})} &= \sum_{l_1 \in dest(s)} \left(\alpha_{l_1^{\text{src}}}^{x(\text{foi},l_1)} \oslash \beta_{l_1^{\text{src}}}^{\text{l.o.}x(\text{foi},l_1)} \right) + \alpha^{F_{\text{src}}(s) \cap x(\text{foi})} \\
&= \sum_{l_1 \in dest(s)} \left(\alpha_{l_1^{\text{src}}}^{x(\text{foi},l_1)} \oslash \left(\beta_{l_1^{\text{src}}} \ominus \alpha_{l_1^{\text{src}}}^{x(x(\text{foi},l_1))} \right) \right) + \alpha^{F_{\text{src}}(s) \cap x(\text{foi})} \\
&= \sum_{l_1 \in dest(s)} (\left(\sum_{l_2 \in dest\left(l_1^{\text{src}}\right)} \left(\alpha_{l_2^{\text{src}}}^{x(\text{foi},\{l_2,l_1\})} \oslash \beta_{l_2^{\text{src}}}^{\text{l.o.}x(\text{foi},\{l_2,l_1\})} \right) + \alpha^{F_{\text{src}}(l_1^{\text{src}}) \cap x(\text{foi},l_1)} \right) \\
&\qquad\qquad \oslash \left(\beta_{l_1^{\text{src}}} \ominus \alpha_{l_1^{\text{src}}}^{x(x(\text{foi},l_1))} \right)) \\
&\quad + \alpha^{F_{\text{src}}(s) \cap x(\text{foi})}
\end{aligned}$$

$$\begin{aligned} = \quad & \sum_{l_1 \in dest(s)} \Bigg(\left(\sum_{l_2 \in dest\left(l_1^{\mathrm{src}}\right)} \left(\alpha_{l_2^{\mathrm{src}}}^{x(\mathrm{foi},\{l_2,l_1\})} \oslash \beta_{l_2^{\mathrm{src}}}^{\mathrm{l.o.}x(\mathrm{foi},\{l_2,l_1\})} \right) + \alpha^{F_{\mathrm{src}}(l_1^{\mathrm{src}}) \cap x(\mathrm{foi},,l_1)} \right) \\ & \qquad \oslash \left(\beta_{l_1^{\mathrm{src}}} \ominus \sum_{l_2 \in dest\left(l_1^{\mathrm{src}}\right)} \left(\alpha_{l_2^{\mathrm{src}}}^{x(x(\mathrm{foi},\{l_2,l_1\}))} \oslash \beta_{l_2^{\mathrm{src}}}^{\mathrm{l.o.}x(x(\mathrm{foi},\{l_2,l_1\}))} \right) \right) \Bigg) \\ & + \alpha^{F_{\mathrm{src}}(s) \cap x(\mathrm{foi})} \\ = \quad & \ldots \end{aligned}$$

The recursion of Algorithm 3.1 is exemplarily unfolded to illustrate the backtracking of interference in a feed-forward network. Initially, the foi's cross-traffic $\alpha_s^{x(\mathrm{foi})}$ is split into the sum of flow arrivals from incoming links and those originating at s itself, $\alpha^{F_{\mathrm{src}}(s) \cap x(\mathrm{foi})}$. Next, the left-over service curves at the sources of incoming links are derived in order to separate the foi's cross-traffic that has to be bounded, $x(\mathrm{foi}, l_1)$, from the cross-traffic it experiences, $x(x(\mathrm{foi}, l_1))$, i.e., there are two directions to further backtrack flows by unfolding the term. Both directions require a new instance of compFFA step 1 in the following steps and will thus operate with independent worst-case assumptions. The PSOO violation depicted in Section 3.2.1 can still occur. Step 3 of Algorithm 3.1 proceeds towards the sources of $x(\mathrm{foi}, l_1)$ and step 4 proceeds towards $x(x(\mathrm{foi}, l_1))$. The backtracking will eventually terminate at the flows' sources.

3.3.3 Deconvolution for $\alpha \in \mathcal{F}_{\mathrm{mTB}}$ with $\beta \in \mathcal{F}_{\mathrm{mRL}}$

AggrPBOOAB heavily relies on the output bounding operation of algNC, i.e., the deconvolution $\oslash$. In [17], an algorithm for the deconvolution of two piecewise linear functions is provided. It can be applied to curves of a class more general than $\mathcal{F}_0$. In this thesis, we are concerned with analyses that are restricted to multi-token-bucket arrival curves $\alpha \in \mathcal{F}_{\mathrm{mTB}}$ and multi-rate-latency service curves $\beta \in \mathcal{F}_{\mathrm{mRL}}$. We provide an algorithm for deconvolution that is tailored to these curves.

Recall the definition of deconvolution (Definition 2.6):

$$(\alpha \oslash \beta)(d) \quad = \quad \sup_{u \geq 0} \{\alpha(d+u) - \beta(u)\}$$

Our explicit solution for the deconvolution of $\alpha \in \mathcal{F}_{\mathrm{mTB}}$ with $\beta \in \mathcal{F}_{\mathrm{mRL}}$ relies on two observations:

1. The inflection points of arrival and service curves are crucial for the result [75].
2. We can split up the supremum into several suprema applied in sequence [17]. We first derive the set of potential solution curves by computing a *result candidate* for

each inflection point α and β. Then, we take the pointwise supremum over the all result candidates.

Lemma 3.1. (Deconvolution of $\alpha \in \mathcal{F}_{\mathrm{mTB}}$ with $\beta \in \mathcal{F}_{\mathrm{mRL}}$) *Let* $\alpha \in \mathcal{F}_{\mathrm{mTB}}$ *and* $\beta \in \mathcal{F}_{\mathrm{mRL}}$. *Further, let* I_α *be the set of locations of inflection points of* α *and* I_β *be the set of locations of inflection points of* β. *By definition* $t = 0$ *is in the set* I_β. *Then, the deconvolution of* α with β *is*

$$(\alpha \oslash \beta)(d) \quad = \quad \sup \left\{ \sup_{i_\beta \in I_\beta} \{\alpha(d + i_\beta) - \beta(i_\beta)\}\,,\; \sup_{i_\alpha \in I_\alpha} \{\alpha(i_\alpha) - \beta(i_\alpha - d)\} \right\}.$$

As mentioned above, the supremum has been split up [17]. Derivations tailored to inflections points of α and β are inside the outer supremum and each inflection point creates a result candidate curve. I.e., the amount of candidates the outer supremum is applied to is equal to the amount of α's inflection points $|I_\alpha|$ plus β's inflection points $|I_\beta|$.

3.3.4 Distributivity of $\oslash$ over $+$

In the aggrPBOOAB, aggregation of cross-traffic is followed by a deconvolution with this aggregate's left-over service curve on the subsequently traversed tandem. See Algorithm 3.1, step 3, where the term inside the outer sum follows the pattern $\sum_i(\alpha_i) \oslash \beta$. This pattern will occur with every recursion level required for aggrPBOOAB. Based on Lemma 3.1, we derive conditions for the distributivity of $\oslash$ over $+$.

Lemma 3.2. (Deconvolution of $\alpha \in \mathcal{F}_{\mathrm{mTB}}$ with $\beta \in \mathcal{F}_{\mathrm{mRL}}$) *Let* $\alpha^{f_1}, \alpha^{f_2} \in \mathcal{F}_{mTB}$ *be arrival curves and let* β *be a service curve. Then, the deconvolution distributes over the aggregation,*

$$\left(\left(\alpha^{f_1} + \alpha^{f_2}\right) \oslash \beta\right)(d) \quad = \quad \left(\left(\alpha^{f_1} \oslash \beta\right) + \left(\alpha^{f_2} \oslash \beta\right)\right)(d),$$

if, for the result candidates of Lemma 3.1, it holds that result candidates for inflection points $i_{\alpha^{\mathbb{F}}} \in I_{\alpha^{\mathbb{F}}}$, $i_{\alpha^{\mathbb{F}}} \leq T$ *and* $i_\beta \in I_\beta$, $i_\beta \leq T$ *are greater equal to the candidates for* $i_{\alpha^{\mathbb{F}}} \in I_{\alpha^{\mathbb{F}}}$, $i_{\alpha^{\mathbb{F}}} > T$ *and* $i_\beta \in I_\beta$, $i_\beta > T$, *where* T *denotes the service curve's latency.*
The condition especially holds for $\beta = \beta_{R,T} \in \mathcal{F}_{\mathrm{RL}}$ *and* $\max\{I_{\alpha^{\mathbb{F}}}\} \leq T$ *(e.g.,* $\alpha \in \mathcal{F}_{\mathrm{TB}}$*).*

Proof. We apply Lemma 3.1 and get

$$\left(\left(\alpha^{f_1} + \alpha^{f_2}\right) \oslash \beta\right)(d) =$$
$$\sup\left\{\sup_{i_\beta \in I_\beta}\left\{\left(\alpha^{f_1} + \alpha^{f_2}\right)(d + i_\beta) - \beta(i_\beta)\right\},\; \sup_{i_{\alpha^{\mathbb{F}}} \in I_{\alpha^{\mathbb{F}}}}\left\{\left(\alpha^{f_1} + \alpha^{f_2}\right)(i_{\alpha^{\mathbb{F}}}) - \beta(i_{\alpha^{\mathbb{F}}} - d)\right\}\right\} \tag{3.1}$$

where $\mathbb{F} = [f_1, f_2]$ and $\alpha^{\mathbb{F}} = \alpha^{f_1} + \alpha^{f_2}$ such that $I_{\alpha^{\mathbb{F}}}$ denotes the set of inflection points of the curve $\alpha^{f_1} + \alpha^{f_2}$, $I_{\alpha^{\mathbb{F}}} = I_{\alpha^{f_1}} \cup I_{\alpha^{f_2}}$.

We are interested in the conditions that allow deconvolution $\oslash$ to distribute over the pointwise addition of curves, i.e.,

$$\Big(\big(\alpha^{f_1} + \alpha^{f_2}\big) \oslash \beta\Big)(d) \quad \overset{\text{conditions?}}{=} \quad \Big(\big(\alpha^{f_1} \oslash \beta\big) + \big(\alpha^{f_2} \oslash \beta\big)\Big)(d).$$

The right side of the equation translates to:

$$\begin{aligned}
&\Big(\big(\alpha^{f_1} \oslash \beta\big) + \big(\alpha^{f_2} \oslash \beta\big)\Big)(d) = \\
&\qquad \sup\left\{ \sup_{i_\beta \in I_\beta} \Big\{\alpha^{f_1}(d + i_\beta) - \beta(i_\beta)\Big\}, \sup_{i_{\alpha^{f_1}} \in I_{\alpha^{f_1}}} \Big\{\alpha^{f_1}(i_{\alpha^{f_1}}) - \beta(i_{\alpha^{f_1}} - d)\Big\} \right\} \\
&\qquad + \sup\left\{ \sup_{i_\beta \in I_\beta} \Big\{\alpha^{f_2}(d + i_\beta) - \beta(i_\beta)\Big\}, \sup_{i_{\alpha^{f_2}} \in I_{\alpha^{f_2}}} \Big\{\alpha^{f_2}(i_{\alpha^{f_2}}) - \beta(i_{\alpha^{f_2}} - d)\Big\} \right\} \qquad (3.2)
\end{aligned}$$

I) Valid Result Candidates: Comparing the first sub-term of equations 3.1 and 3.2's outer suprema (deriving result candidates for β's inflection points), we see that $-\beta(i_\beta)$ has to be considered twice in $\big(\big(\alpha^{f_1} \oslash \beta\big) + \big(\alpha^{f_1} \oslash \beta\big)\big)(d)$. Therefore, we need to ensure that $\forall i_\beta \in I_\beta . \, \beta(i_\beta) = 0$ which implies that $\forall 0 \leq d \leq \max\{I_\beta\} . \, \beta(d) = 0$, i.e., $\beta = \beta_{R,T} \in \mathcal{F}_{\mathrm{RL}}$.

Similarly, in the second terms (deriving result candidates for $\alpha^{\mathbb{F}}$'s, α^{f_1}'s and α^{f_2}'s inflection points), we need to ensure $\beta\left(i_{\alpha^{\mathbb{F}}} - d\right) = \beta\left(i_{\alpha^{f_1}} - d\right) + \beta\left(i_{\alpha^{f_2}} - d\right)$. We know that $I_{\alpha^{\mathbb{F}}} = I_{\alpha^{f_1}} \cup I_{\alpha^{f_2}}$. Thus, we get the condition $\forall i_{\alpha^{\mathbb{F}}} \in I_{\alpha^{\mathbb{F}}} . \, \beta\left(i_{\alpha^{\mathbb{F}}} - d\right) = \beta\left(i_{\alpha^{\mathbb{F}}} - d\right) + \beta\left(i_{\alpha^{\mathbb{F}}} - d\right)$ from equations 3.1 and 3.2. This equation is fulfilled iff $\forall i_{\alpha^{\mathbb{F}}} \in I_{\alpha^{\mathbb{F}}} . \, \beta\left(i_{\alpha^{\mathbb{F}}} - d\right) = 0$. It translates to the condition $i_{\alpha^{\mathbb{F}}}^{\max} := \max\{I_{\alpha^{\mathbb{F}}}\} \leq T = i_\beta$, i.e., $\alpha^{\mathbb{F}}$ must be a multi-token-bucket curve whose rightmost inflection point does not exceed β's latency.

If these conditions are met, we get:

$$\begin{aligned}
&\Big(\big(\alpha^{f_1} \oslash \beta\big) + \big(\alpha^{f_2} \oslash \beta\big)\Big)(d) = \\
&\qquad \sup\Big\{\alpha^{f_1}(d + i_\beta),\, \alpha^{f_1}(i_{\alpha^{\mathbb{F}}}^{\max})\Big\} + \sup\Big\{\alpha^{f_2}(d + i_\beta),\, \alpha^{f_2}(i_{\alpha^{\mathbb{F}}}^{\max})\Big\}
\end{aligned}$$

From $i^{\max}_{\alpha^{\mathbb{F}}} \leq i_\beta$ and $t \geq 0$ we know that $\alpha^{f_i}\left(d+i_\beta\right) \geq \alpha^{f_i}\left(i^{\max}_{\alpha^{\mathbb{F}}}\right)$, $i \in \{1,2\}$ such that

$$
\begin{aligned}
&\left(\left(\alpha^{f_1} \oslash \beta\right) + \left(\alpha^{f_2} \oslash \beta\right)\right)(d) = \\
&\qquad \sup\left\{\sup_{i_\beta \in I_\beta}\left\{\alpha^{f_1}\left(d+i_\beta\right) - \beta\left(i_\beta\right)\right\}, \sup_{i_{\alpha^{f_1}} \in I_{\alpha^{f_1}}}\left\{\alpha^{f_1}\left(i_{\alpha^{f_1}}\right) - \beta\left(i_{\alpha^{f_1}} - d\right)\right\}\right\} \\
&\quad + \sup\left\{\sup_{i_\beta \in I_\beta}\left\{\alpha^{f_2}\left(d+i_\beta\right) - \beta\left(i_\beta\right)\right\}, \sup_{i_{\alpha^{f_2}} \in I_{\alpha^{f_2}}}\left\{\alpha^{f_2}\left(i_{\alpha^{f_2}}\right) - \beta\left(i_{\alpha^{f_2}} - d\right)\right\}\right\}
\end{aligned}
$$

$$
\begin{aligned}
&\left(\text{Condition } \beta = \beta_{R,T} \in \mathcal{F}_{\text{RL}}\right) \\
&= \sup\left\{\alpha^{f_1}\left(d+T\right), \sup_{i_{\alpha^{f_1}} \in I_{\alpha^{f_1}}}\left\{\alpha^{f_1}\left(i_{\alpha^{f_1}}\right) - \beta\left(i_{\alpha^{f_1}} - d\right)\right\}\right\} \\
&+ \sup\left\{\alpha^{f_2}\left(d+T\right), \sup_{i_{\alpha^{f_2}} \in I_{\alpha^{f_2}}}\left\{\alpha^{f_2}\left(i_{\alpha^{f_2}}\right) - \beta\left(i_{\alpha^{f_2}} - d\right)\right\}\right\}
\end{aligned}
$$

$$
\begin{aligned}
&\left(\text{Condition } i_{\alpha^{f_i}} \leq i^{\max}_{\alpha^{\mathbb{F}}} \leq T\right) \\
&= \sup\left\{\alpha^{f_1}\left(d+T\right), \sup_{i_{\alpha^{f_1}} \in I_{\alpha^{f_1}}}\left\{\alpha^{f_1}\left(i^{\max}_{\alpha^{\mathbb{F}}}\right)\right\}\right\} \\
&+ \sup\left\{\alpha^{f_2}\left(d+T\right), \sup_{i_{\alpha^{f_2}} \in I_{\alpha^{f_2}}}\left\{\alpha^{f_2}\left(i^{\max}_{\alpha^{\mathbb{F}}}\right)\right\}\right\}
\end{aligned}
$$

$$
\begin{aligned}
&\left(\text{Condition } i^{\max}_{\alpha^{\mathbb{F}}} \leq T\right) \\
&= \alpha^{f_1}\left(d+T\right) + \alpha^{f_2}\left(d+T\right)
\end{aligned}
$$

$$
\begin{aligned}
&\left(\text{Condition } \beta = \beta_{R,T} \in \mathcal{F}_{\text{RL}}\right) \\
&= \alpha^{f_1}\left(d+i_\beta\right) + \alpha^{f_2}\left(d+i_\beta\right) \\
&= \left(\alpha^{f_1} + \alpha^{f_2}\right)\left(d+i_\beta\right) \\
&= \left(\alpha^{f_1} + \alpha^{f_2}\right)\left(d+i_\beta\right) + 0 \\
&= \sup_{i_\beta}\left\{\left(\alpha^{f_1} + \alpha^{f_2}\right)\left(d+i_\beta\right) - \beta\left(i_\beta\right)\right\}
\end{aligned}
$$

$$
\begin{aligned}
&\left(\text{Condition } i^{\max}_{\alpha^{\mathbb{F}}} \leq i_\beta\right) \\
&= \sup\left\{\sup_{i_\beta \in I_\beta}\left\{\left(\alpha^{f_1} + \alpha^{f_2}\right)\left(d+i_\beta\right) - \beta\left(i_\beta\right)\right\}, \left(\alpha^{f_1} + \alpha^{f_2}\right)\left(i^{\max}_{\alpha^{\mathbb{F}}}\right) - \beta\left(i^{\max}_{\alpha^{\mathbb{F}}} - d\right)\right\}
\end{aligned}
$$

$$\left(\text{Condition } I_{\alpha^{\mathbb{F}}} \ni i_{\alpha^{\mathbb{F}}} \leq i_{\alpha^{\mathbb{F}}}^{\max}\right)$$

$$= \sup\left\{\sup_{i_\beta \in I_\beta}\left\{\left(\alpha^{f_1}+\alpha^{f_2}\right)(d+i_\beta)-\beta(i_\beta)\right\}, \sup_{i_{\alpha^{\mathbb{F}}} \in I_{\alpha^{\mathbb{F}}}}\left\{\left(\alpha^{f_1}+\alpha^{f_2}\right)(i_{\alpha^{\mathbb{F}}})-\beta(i_{\alpha^{\mathbb{F}}}-d)\right\}\right\}$$

$$= \left(\left(\alpha^{f_1}+\alpha^{f_2}\right) \oslash \beta\right)(d)$$

II) Invalid Result Candidates without Impact: The above conditions were derived by matching the outcome of equivalent parts of equation 3.1 and equation 3.2. I.e., if the conditions on α^{f_1}, α^{f_2}, $\alpha^{\mathbb{F}}$, and β are met, the candidate curves derived by the inner terms are equal and so is the result of the outer supremum.

This approach is, however, deriving too strong conditions. The outer supremum's impact needs to be considered in more detail. This operation has the ability to "remove" invalid result candidates where the distributivity does not apply, i.e., where $\left(\left(\alpha^{f_1}+\alpha^{f_2}\right) \oslash \beta\right)(d) \neq \left(\left(\alpha^{f_1} \oslash \beta\right)+\left(\alpha^{f_2} \oslash \beta\right)\right)(d)$. From the conditions above we know that invalid candidates are derived for the following inflection points of curves:

- $\forall i_{\alpha^{\mathbb{F}}} \in I_{\alpha^{\mathbb{F}}}. i_{\alpha^{\mathbb{F}}} > T$
- $\forall i_\beta \in I_\beta. i_\beta > T$, i.e., $\beta \in \mathcal{F}_{\mathrm{mRL}}$ with latency T

If these invalid result candidate curves are smaller than the supremum of valid candidates, the outer supremum in equation 3.2 removes them such that they do not impact the final result. To check if we can distribute a deconvolution over an addition of two multi-token-bucket curves, we need to split the derivation into two parts, according to the location of inflection points:

1. The part deriving valid candidates for inflection points on $[0, T]$ and

2. the part deriving invalid candidates for inflection points on $(T, +\infty)$.

Then, we need to check if the supremum of invalid result candidates is smaller than the supremum of valid result candidates. We either require $\left(\left(\alpha^{f_1} \oslash \beta\right)+\left(\alpha^{f_2} \oslash \beta\right)\right)(d)$ with

$$\left(\left(\alpha^{f_1} \oslash \beta\right)+\left(\alpha^{f_2} \oslash \beta\right)\right)(d)\Big|_{i \leq T} :=$$

$$\sup\left\{\sup_{I_\beta \ni i_\beta \leq T}\left\{\alpha^{f_1}(d+i_\beta)-\beta(i_\beta)\right\}, \sup_{I_{\alpha^{f_1}} \ni i_{\alpha^{f_1}} \leq T}\left\{\alpha^{f_1}(i_{\alpha^{f_1}})-\beta(i_{\alpha^{f_1}}-d)\right\}\right\}$$

$$+\sup\left\{\sup_{I_\beta \ni i_\beta \leq T}\left\{\alpha^{f_2}(d+i_\beta)-\beta(i_\beta)\right\}, \sup_{I_{\alpha^{f_2}} \ni i_{\alpha^{f_2}} \leq T}\left\{\alpha^{f_2}(i_{\alpha^{f_2}})-\beta(i_{\alpha^{f_2}}-d)\right\}\right\}$$

and

$$\left(\left(\alpha^{f_1} \oslash \beta\right) + \left(\alpha^{f_2} \oslash \beta\right)\right)(d)\Big|_{i>T} \coloneqq$$
$$\sup\left\{\sup_{I_\beta \ni i_\beta > T}\left\{\alpha^{f_1}(d+i_\beta) - \beta(i_\beta)\right\}, \sup_{I_{\alpha^{f_1}} \ni i_{\alpha^{f_1}} > T}\left\{\alpha^{f_1}(i_{\alpha^{f_1}}) - \beta(i_{\alpha^{f_1}} - d)\right\}\right\}$$
$$+ \sup\left\{\sup_{I_\beta \ni i_\beta > T}\left\{\alpha^{f_2}(d+i_\beta) - \beta(i_\beta)\right\}, \sup_{I_{\alpha^{f_2}} \ni i_{\alpha^{f_2}} > T}\left\{\alpha^{f_2}(i_{\alpha^{f_2}}) - \beta(i_{\alpha^{f_2}} - d)\right\}\right\},$$

leading to

$$\forall t \geq 0 : \left(\left(\alpha^{f_1} \oslash \beta\right) + \left(\alpha^{f_2} \oslash \beta\right)\right)(d)\Big|_{i\leq T} \geq \left(\left(\alpha^{f_1} \oslash \beta\right) + \left(\alpha^{f_2} \oslash \beta\right)\right)(d)\Big|_{i>T}$$

or we require $\left(\left(\alpha^{f_1} + \alpha^{f_2}\right) \oslash \beta\right)(d)$ with

$$\left(\left(\alpha^{f_1} + \alpha^{f_2}\right) \oslash \beta\right)(d)\Big|_{i\leq T} \coloneqq$$
$$\sup\left\{\sup_{I_{\alpha^{\mathbb{F}}} \ni i_{\alpha^{\mathbb{F}}} \leq T}\left\{\left(\alpha^{f_1} + \alpha^{f_2}\right)(d+i_\beta) - \beta(i_\beta)\right\}, \sup_{I_{\alpha^{\mathbb{F}}} \ni i_{\alpha^{\mathbb{F}}} \leq T}\left\{\left(\alpha^{f_1} + \alpha^{f_2}\right)(i_{\alpha^{\mathbb{F}}}) - \beta(i_{\alpha^{\mathbb{F}}} - d)\right\}\right\}$$

and

$$\left(\left(\alpha^{f_1} + \alpha^{f_2}\right) \oslash \beta\right)(d)\Big|_{i>T} \coloneqq$$
$$\sup\left\{\sup_{I_{\alpha^{\mathbb{F}}} \ni i_{\alpha^{\mathbb{F}}} > T}\left\{\left(\alpha^{f_1} + \alpha^{f_2}\right)(d+i_\beta) - \beta(i_\beta)\right\}, \sup_{I_{\alpha^{\mathbb{F}}} \ni i_{\alpha^{\mathbb{F}}} > T}\left\{\left(\alpha^{f_1} + \alpha^{f_2}\right)(i_{\alpha^{\mathbb{F}}}) - \beta(i_{\alpha^{\mathbb{F}}} - d)\right\}\right\}$$

leading to

$$\forall t \geq 0 \ : \ \left(\left(\alpha^{f_1} + \alpha^{f_2}\right) \oslash \beta\right)(d)\Big|_{i\leq T} \quad \geq \quad \left(\left(\alpha^{f_1} + \alpha^{f_2}\right) \oslash \beta\right)(d)\Big|_{i>T}.$$

□

Lemma 3.2 can be extended to flow aggregates of arbitrary sizes > 2 by iterative application.

3.3.5 Ranking of PBOO Arrival Bounding Alternatives

Lemma 3.2 allows us to rank both PBOO cross-traffic arrival bounding methods, segrPBOOAB and aggrPBOOAB.

Theorem 3.1. (Accuracy of PBOO Cross-traffic Arrival Bounding) *Cross-traffic arrival bounding with aggrPBOOAB is more accurate than segrPBOOAB if the conditions of Lemma 3.2 are fulfilled.*

Proof. Without loss of generality (detailed explanation follows after the proof), we prove this statement by showing that there are no beneficial effects of bounding long tandems (Figure 3.5b) that are more advantageous than bounding both flows aggregately on shorter tandems (Figure 3.5c).

First, we derive the segregated cross-flow arrival bounds at s_1, i.e., $\alpha_{s_1}^{xf_n}$, $n \in \{1,2\}$, and aggregate the results to $\alpha_{s_1}^{[xf_1,xf_2]} = \alpha_{s_1}^{xf_1} + \alpha_{s_1}^{xf_2}$.

$$\begin{aligned}
\alpha_{s_1}^{xf_n} &= \alpha_{s_{0n}}^{xf_n} \oslash \beta^{\text{l.o.}xf_n}_{\langle s_{0n}, s_0\rangle} &&= \alpha_{s_{0n}}^{xf_n} \oslash \left(\beta_{s_{0n}} \otimes \left(\beta_{s_0} \ominus \alpha_{s_0}^{xf_{\overline{n}}}\right)\right)\\
& &&= \alpha_{s_{0n}}^{xf_n} \oslash \left(\beta_{s_{0n}} \otimes \left(\beta_{s_0} \ominus \left(\alpha_{s_{0\overline{n}}}^{xf_{\overline{n}}} \oslash \beta^{\text{l.o.}xf_{\overline{n}}}_{s_{0\overline{n}}}\right)\right)\right)\\
& &&= \alpha_{s_{0n}}^{xf_n} \oslash \left(\beta_{s_{0n}} \otimes \left(\beta_{s_0} \ominus \left(\alpha_{s_{0\overline{n}}}^{xf_{\overline{n}}} \oslash \beta_{s_{0\overline{n}}}\right)\right)\right)
\end{aligned}$$

where $\overline{n}$ denotes the opposite cross-flow's index, i.e., $\overline{1} = 2$ and $\overline{2} = 1$. Therefore,

$$\begin{aligned}
\alpha_{s_0}^{[xf_1,xf_2]} &= \alpha_{s_{01}}^{xf_1} \oslash \left(\beta_{s_{01}} \otimes \left(\beta_{s_0} \ominus \left(\alpha_{s_{02}}^{xf_2} \oslash \beta_{s_{02}}\right)\right)\right)\\
&\quad + \alpha_{s_{02}}^{xf_2} \oslash \left(\beta_{s_{02}} \otimes \left(\beta_{s_0} \ominus \left(\alpha_{s_{01}}^{xf_1} \oslash \beta_{s_{01}}\right)\right)\right) \qquad (3.3)
\end{aligned}$$

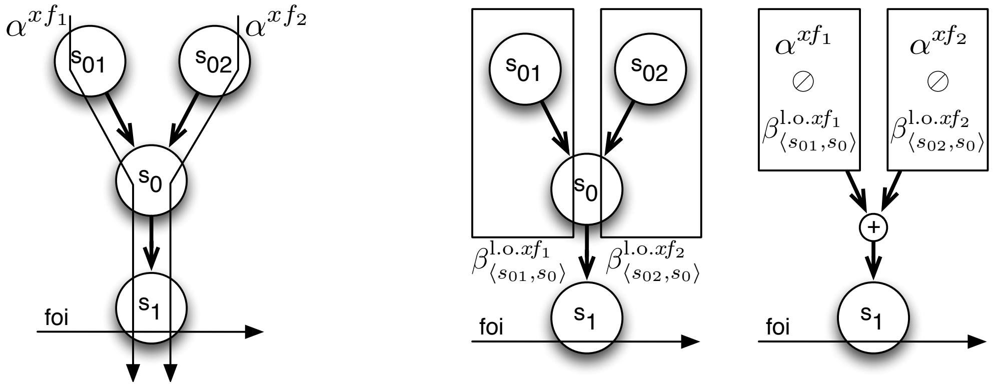
Next, we derive the aggregate cross-traffic arrival bound $\alpha_{s_0}^{[xf_1,xf_2]}$ according to Algorithm 3.1.

$$\begin{aligned}
\alpha_{s_0}^{[xf_1,xf_2]} &= \left(\alpha_{s_0}^{xf_1} + \alpha_{s_0}^{xf_2}\right) \oslash \beta_{s_0}^{\text{l.o.}[xf_1,xf_2]}\\
&= \left(\left(\alpha_{s_{01}}^{xf_1} \oslash \beta_{s_{01}}\right) + \left(\alpha_{s_{02}}^{xf_2} \oslash \beta_{s_{02}}\right)\right) \oslash \beta_{s_0} \qquad (3.4)
\end{aligned}$$

Last, we show that the former arrival bound cannot be smaller than the latter one, i.e., (3.4) $\leq$ (3.3) holds. This step relies on Lemma 3.2 and thus we inherit its conditions in this theorem. For (3.4), distributivity of $\oslash$ over $+$ results in:

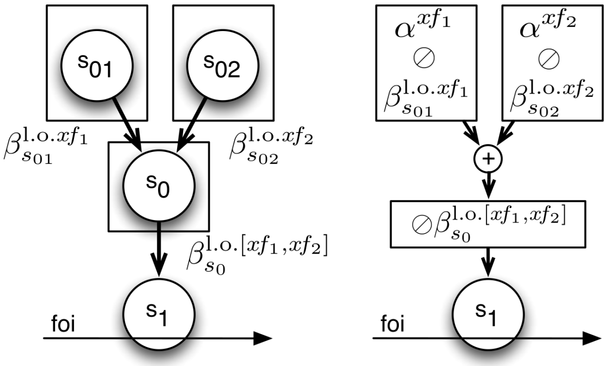
$$\left(\left(\alpha_{s_{01}}^{xf_1} \oslash \beta_{s_{01}}\right) + \left(\alpha_{s_{02}}^{xf_2} \oslash \beta_{s_{02}}\right)\right) \oslash \beta_{s_0} \;=\; \left(\alpha_{s_{01}}^{xf_1} \oslash \beta_{s_{01}}\right) \oslash \beta_{s_0} + \left(\alpha_{s_{02}}^{xf_2} \oslash \beta_{s_{02}}\right) \oslash \beta_{s_0}$$

and (3.4) $\leq$ (3.3) translates to


α^{xf_1}
α^{xf_2}
s01
s02
s0
s1
foi
$\beta^{\text{l.o.}xf_1}_{\langle s_{01},s_0\rangle}$
$\beta^{\text{l.o.}xf_2}_{\langle s_{02},s_0\rangle}$
$\alpha^{xf_1} \oslash \beta^{\text{l.o.}xf_1}_{\langle s_{01},s_0\rangle}$
$\alpha^{xf_2} \oslash \beta^{\text{l.o.}xf_2}_{\langle s_{02},s_0\rangle}$
+

(a) Sample network.

(b) Segregated cross-flow arrival bounding segrPBOOAB.


s01
s02
s0
s1
foi
$\beta^{\text{l.o.}xf_1}_{s_{01}}$
$\beta^{\text{l.o.}xf_2}_{s_{02}}$
$\beta^{\text{l.o.}[xf_1,xf_2]}_{s_0}$
$\alpha^{xf_1} \oslash \beta^{\text{l.o.}xf_1}_{s_{01}}$
$\alpha^{xf_2} \oslash \beta^{\text{l.o.}xf_2}_{s_{02}}$
+
$\oslash\beta^{\text{l.o.}[xf_1,xf_2]}_{s_0}$

(c) Aggregate cross-traffic arrival bounding aggrPBOOAB.

Figure 3.5: Decomposition of a network for PBOO cross-traffic arrival bounding: (b) depicts the segregation of cross-flows (segrPBOOAB) and (c) illustrates our aggregation approach (aggrPBOOAB) resulting in shorter tandems.

$$\begin{aligned}
&\left(\alpha^{xf_1}_{s_{01}} \oslash \beta_{s_{01}}\right) \oslash \beta_{s_0} + \left(\alpha^{xf_2}_{s_{02}} \oslash \beta_{s_{02}}\right) \oslash \beta_{s_0} \\
&\quad \leq \quad \alpha^{xf_1}_{s_{01}} \oslash \left(\beta_{s_{01}} \otimes \left(\beta_{s_0} \ominus \left(\alpha^{xf_2}_{s_{02}} \oslash \beta_{s_{02}}\right)\right)\right) + \alpha^{xf_2}_{s_{02}} \oslash \left(\beta_{s_{02}} \otimes \left(\beta_{s_0} \ominus \left(\alpha^{xf_1}_{s_{01}} \oslash \beta_{s_{01}}\right)\right)\right).
\end{aligned}$$

We now compare each segregated cross-flow's impact on the final arrival bound. This results in two sub-terms:

$$\left(\alpha^{xf_n}_{s_{0n}} \oslash \beta_{s_{0n}}\right) \oslash \beta_{s_0} \quad \leq \quad \alpha^{xf_n}_{s_{0n}} \oslash \left(\beta_{s_{0n}} \otimes \left(\beta_{s_0} \ominus \left(\alpha^{xf_{\overline{n}}}_{s_{0\overline{n}}} \oslash \beta_{s_{0\overline{n}}}\right)\right)\right)$$

Next, we can apply the composition rule for $\oslash$ ([50], Theorem 3.1.12) and use $\otimes$'s commutativity to reformulate the terms to

$$\alpha^{xf_n}_{s_{0n}} \oslash \beta_{s_{0n}} \oslash \beta_{s_0} \quad \leq \quad \alpha^{xf_n}_{s_{0n}} \oslash \beta_{s_{0n}} \oslash \left(\beta_{s_0} \ominus \left(\alpha^{xf_{\overline{n}}}_{s_{0\overline{n}}} \oslash \beta_{s_{0\overline{n}}}\right)\right).$$

These equations reveal that the crucial difference between both arrival bounding alternatives is the respective (left-over) service curve at server s_0. It defines the difference between both arrival bounds; potential end-to-end effects are superseded by the negative impact of PSOO violation.

Finally, as all curves are from $\mathcal{F}_0$, we know that

$$\beta_{s_0} \quad \geq \quad \beta_{s_0} \ominus \left(\alpha_{s_{0n}}^{xf_n} \oslash \beta_{s_{0n}} \right),$$

i.e., aggrPBOOAB's service curve is larger, and for $\forall \beta_{s_x}, \beta_{s_y} \in \mathcal{F}_{\mathrm{mRL}}$ and $\forall \alpha \in \mathcal{F}_{\mathrm{mTB}}$ in particular, we know that

$$\beta_{s_x} \geq \beta_{s_y} \quad \Rightarrow \quad \left(\alpha \oslash \beta_{s_x} \right) \leq \left(\alpha \oslash \beta_{s_y} \right),$$

i.e., a larger service curve leads to a smaller output bound.
Therefore, aggrPBOOAB outperforms segrPBOOAB under the conditions of Lemma 3.2.
□

Note, that equality between both arrival bounding alternatives only holds for the trivial cases where the service β_{s_0} is infinitely large, i.e., $\beta_{+\infty,0}$, or $\alpha_{s_{0n}}^{xf_n}$ (α^{xf_n}) is zero.

Last, let us clarify why the seemingly simple scenario of Figure 3.5a allows for our ranking of aggrPBOOAB over segrPBOOAB in general feed-forward (subject to Lemma 3.2's preconditions). Any more involved feed-forward network can be decomposed into a combination of variants of the network shown in Figure 3.5a, retaining Theorem 3.1's validity:

Intermediate tandems instead of single servers s_0, s_{01}, or s_{02} can be assumed to consist of multiple servers in tandem that were convolved into single servers for analysis (see Theorem 2.2). Then, the above proof as well as Algorithm 3.1 (aggrPBOOAB) virtually move across the network tandem-by-tandem instead of server-by-server.

Cross-traffic of cross-traffic AggrPBOOAB compromises on the source-to-interference view of segregated cross-flow arrival bounding. Tandems are restricted to sequences of servers shared by all flows in the respective cross-traffic aggregate. If an aggregate had its own cross-traffic (see $x(x(\mathrm{foi}, l_1))$ above), another PBOO left-over service curve derivation (Theorem 2.3) and thus an arrival bounding (of $x(x(\mathrm{foi}, l_1))$) would be needed. This new instance of compFFA step 1 operates with its own worst-case assumptions for $x(x(\mathrm{foi}, l_1))$. In the best case for segregated arrival bounding, when it is able to derive the same $\alpha^{x(x(\mathrm{foi}, l_1))}$ as aggregate arrival bounding, Theorem 3.1's proof remains unchanged. In any other case, segrPBOOAB even operates on worse left-over service curves than aggrPBOOAB.

Further cross-traffic with the same interference pattern Assume another cross-flow (aggregate) xf_3 merges with the existing cross-traffic $[xf_1, xf_2]$ at server s_0, entering from a different server (or convolved tandem of left-over service curves) s_{03}. Then, equations containing the segregated flows' impact on arrival bounds are expanded with xf_3's influence

$$\begin{aligned}
\alpha^{xf_1}_{s_{01}} \oslash \beta_{s_{01}} \oslash \beta_{s_0} &\geq \alpha^{xf_1}_{s_{01}} \oslash \beta_{s_{01}} \oslash \left(\beta_{s_0} \ominus \left(\alpha^{xf_2}_{s_{02}} \oslash \beta_{s_{02}} + \alpha^{xf_3}_{s_{03}} \oslash \beta_{s_{03}}\right)\right), \\
\alpha^{xf_2}_{s_{02}} \oslash \beta_{s_{02}} \oslash \beta_{s_0} &\geq \alpha^{xf_2}_{s_{02}} \oslash \beta_{s_{02}} \oslash \left(\beta_{s_0} \ominus \left(\alpha^{xf_1}_{s_{01}} \oslash \beta_{s_{01}} + \alpha^{xf_3}_{s_{03}} \oslash \beta_{s_{03}}\right)\right)
\end{aligned}$$

and a similar equation for xf_3's impact on the aggregate arrival bound is added

$$\alpha^{xf_3}_{s_{03}} \oslash \beta_{s_{03}} \oslash \beta_{s_0} \geq \alpha^{xf_3}_{s_{03}} \oslash \beta_{s_{03}} \oslash \left(\beta_{s_0} \ominus \left(\alpha^{xf_1}_{s_{01}} \oslash \beta_{s_{01}} + \alpha^{xf_2}_{s_{02}} \oslash \beta_{s_{02}}\right)\right).$$

Neither adaptation impacts the proof's core statement; s_0 still constitutes the crucial bottleneck server where aggregation outperforms segregation.

Further cross-flows with different interference patterns Assume s_0, s_{01}, and s_{02} actually consisted of tandems of servers and there were further cross-flows of the foi, i.e., in $x(\text{foi})$, merging and demultiplexing somewhere on these tandems. In this case, the above reasoning would have to be repeated recursively for every backtracking required – similar to *cross-traffic of cross-traffic,* yet, with a SFA tandem analysis for the derivation of the left-over service instead of Theorem 2.3. Again, each recursion level constitutes an independent instance of arrival bounding, with its own bottleneck server where aggregation outperforms segregation due to the former's violation of the PSOO principle. The effect causing aggregation's superiority is thus amplified in more involved feed-forward networks.

3.3.6 Accuracy Evaluation

In our numerical experiments, we use the aSHIIP topology generator [84] to randomly create Erdős-Rényi device graphs following the $G(n, p)$-model with $p = 0.1$. We evaluate the impact of improved arrival bounding in flat and hierarchical network topologies with 32 devices resulting in 114 servers (flat) as well as 73 servers (hierarchical). The amount of flows is continuously increased in steps of 50 randomly routed unit size flows ($\alpha = \gamma_{1,1}$) until reaching the network's capacity limit[6]. Each resulting network is analyzed with an SFA with segrPBOOAB as suggested in [10] as well as the aggrPBOOAB we contributed in this section.

---

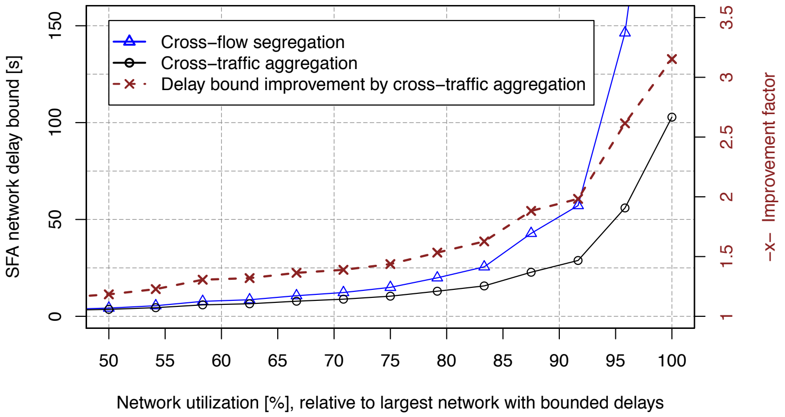
[6] The flat network could route up to 1200 flows, the hierarchical network could route at most 900 flows.

In this accuracy evaluation, we aim to gain insight on the impact of our aggrPBOOAB on the compFFA, more precisely, Section 3.3.5's crucial insight: amplification of aggregate bounding's superiority when the number of bottlenecks as well as their utilization increases. Our measure is the so-called *network delay bound*. It is the worst-case end-to-end delay bound occurring in a network, i.e., it is globally valid for all flows in the analyzed network. Our results are shown in Figure 3.6. Both figures depict the network delay bounds for network utilizations between 50% and 100%, both relative to the largest network in terms of flows and bounded delays. We express the improvement from the new result's point of view by the measure suggested for the LPA in [12]. This improvement factor is defined as $\frac{D(\text{segrPBOOAB})}{D(\text{aggrPBOOAB})}$. A value of 1.0 denotes parity, a value in $(0, 1)$ indicates a worsened network delay bound and values > 1.0 measure the improvement. Note, that the actual impact of an improvement depends on the context. Even large improvement factors might not suffice to reach a required level of delay guarantee whereas a relatively small one in a different network could be decisive for schedulability of a critical task. Therefore, we focus our presentation of improvement factors or delay bound reductions in % instead of the comparison of absolute values such as the derived delay bounds themselves.

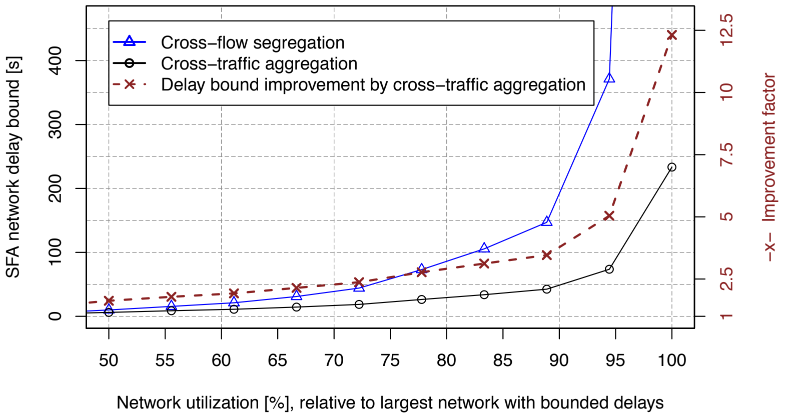
In both networks, our aggrPBOOAB achieves a considerable network delay bound improvement. All improvement factors are > 1.0 (see Figure 3.6, dashed line with crosses, right y-axis of each plot). In the flat ER network with the medium utilization of 50%, we achieve an improvement of factor 1.18, which already reduces the delay bound by about 15%. With growing utilization and fast growing delay bounds, the factor increases to 3.15 (reduction of 68%, Figure 3.6a). The hierarchical network possesses a set of predefined bottleneck links – those links connecting the levels of the hierarchy. The aSHIIP generator creates a relatively static amount of levels, 5 to 6 in our experiments. Randomly routed flows are likely to cross levels and we can observe considerable gains. At 50% network utilization, the factor is at 1.62 (38% reduction) and it grows to a factor of 12.3 (91% reduction) at 100% due to the asymptotic growth of delay bounds. At the evaluated hierarchical networks with a utilization of more than 66%, the delay bound could always be at least halved with aggrPBOOAB (see Figure 3.6b).

Our experiments reveal these general trends:

- The aggrPBOOAB method for compFFA allows algNC to compute more accurate network delay bounds than segrPBOOAB, independent of the network utilization. The highest gains are achieved when the network reaches its capacity limit.
- The predefined bottleneck links of the hierarchical ER topology lead to a higher impact of aggregate arrival bounding than in the flat network topologies.
- The cross-traffic arrival bounding method itself does not affect the network delay



(a) Flat ER network.



(b) Hierarchical ER network.

Figure 3.6: Accuracy evaluation of ER networks with 32 devices each: SFA with two different cross-traffic arrival bounding alternatives, segrPBOOAB and aggrPBOOAB.

bound's asymptotic growth pattern when utilization is increased. However, the growth rate is slowed down considerably and in cases of a large bottleneck utilization, our aggrPBOOAB achieves network delay bounds that are multiple times more accurate than the segrPBOOAB ones.

3.4 Cross-traffic Burstiness Reduction

In the following, we prove a counterintuitive property of algNC analysis: By assuming a worse network setting, better delay bounds can be derived. To be precise, we assume worse interference with the foi by aggregating its cross-traffic with other flows in the network. The analysis does not only bound the burstiness of actual cross-flows but additionally that of flows not directly interfering with the foi. In return, the analysis can aggregate the totality of flows like the TFA. We prove that this overly pessimistic aggregation can be beneficial to the analysis nonetheless. The pessimism can become less impactful than the aggregation's positive effects and therefore we are able to reduce the derived cross-traffic burstiness. Worst-case burstiness is the crucial characteristic of cross-traffic arrivals. In the foi analysis of compFFA step 2, it defines the latency increase in both left-over service curve derivations of algNC, i.e., SFA and PMOOA. Whereas the previous section improved the overall cross-traffic arrival bounding method by circumventing the segregation problem, this section presents an improvement explicitly targeting the worst-case cross-traffic burstiness.

3.4.1 An Alternative Output Bound

The output bound operation, i.e., the deconvolution, is applied to derive the burstiness and the rate of flow arrivals (see Algorithm 3.1). Therefore, we first derive a novel output bound that is based on the input/output-relation of a server. It does not rely on the deconvolution.

Let A, A' be input and output to/from a server. We assume to have an arrival curve α for the arrivals A and a service curve β offered by the server. Let us further assume that the arrival curve α is such that for $d > 0$ it can be written as

$$\alpha(d) = \tilde{\alpha}(d) + \alpha(0^+),$$

with $\tilde{\alpha}$ being a sub-additive curve (defined for $d > 0$ by the above equation and with $\tilde{\alpha}(0) = 0$), and $\alpha(0^+) = \lim_{d \to 0^+} \alpha(d)$. Clearly, this means that α is also a sub-additive function. Further note that, for instance, any concave piecewise-linear arrival curve meets this condition, hence it is not restrictive in practice.

Noting that we can bound the backlog for any given arrival process A by

$$B(t) = A(t) - A'(t) \leq A(t) - (A \otimes \beta)(t) = \sup_{0 \leq u \leq t} \{A(t) - A(u) - \beta(t-u)\},$$

we provide the alternative output bound in the following theorem.

Theorem 3.2. *Under the above assumptions and notations, an output bound on the departure flow (aggregate) A' can be calculated as*

$$\alpha'(d) = \alpha(d) + (v(\alpha, \beta) - \alpha(0^+)) \cdot 1_{\{d>0\}}.$$

Proof. Let $s < t$:

$$\begin{aligned} A'(t) - A'(s) &= A(t) - A(s) + B(s) - B(t) \\ &\leq A(t) - A(s) + B(s) \end{aligned}$$

$$\begin{aligned} &\leq A(t) - A(s) + \sup_{0 \leq u \leq s} \{A(s) - A(u) - \beta(s-u)\} \\ &= \sup_{0 \leq u \leq s} \{A(t) - A(u) - \beta(s-u)\} \\ &\leq \sup_{0 \leq u \leq s} \{\alpha(t-u) - \beta(s-u)\} \\ &= \sup_{0 \leq u \leq s} \{\tilde{\alpha}(t-u) + \alpha(0^+) - \beta(s-u)\} \\ &\leq \sup_{0 \leq u \leq s} \{\tilde{\alpha}(t-s) + \tilde{\alpha}(s-u) + \alpha(0^+) - \beta(s-u)\} \\ &= \tilde{\alpha}(t-s) + \sup_{0 \leq u \leq s} \{\alpha(s-u) - \beta(s-u)\} \\ &\leq \tilde{\alpha}(t-s) + v(\alpha, \beta) \\ &= \alpha(t-s) + v(\alpha, \beta) - \alpha(0^+) = \alpha'(t-s). \end{aligned}$$

For $s = t : A'(t) - A'(s) = 0 = \alpha'(t-s)$. □

Note, that this result resembles a known basic result that can be found in Chang's textbook in Lemma 1.4.2 [23]. This lemma states that for a server with a bound on the queue $\bar{q}$ and a $\gamma_{r,b}$-constrained input, an output bound can be given as $\gamma_{r,b+\bar{q}}$.

Lemma. *([23], Lemma 1.4.2) Let A, A' be input and output to/from a server s. If α is an arrival curve for A and B_s a bound on the backlog of s, then the output of s, A', is bounded by $\alpha + B_s$.*

Proof. Let $u \leq t$, then we know that $A(u) - A'(u) \leq B_s$ and that $A'(t) \leq A(t)$ (flow constraint). This leads to

$$\begin{aligned} A'(t) - A'(u) &\leq A(t) - A(u) + B_s \\ &\leq \alpha(t-u) + B_s \end{aligned}$$

□

Besides generalizing this lemma, we point out that we actually improve it, as we basically get rid of the burst term and would obtain $\gamma_{r,\bar{q}}$ as an output bound under Chang's assumptions.

Next, we show that our alternative output bound can be used to bound the output of individual flows of an aggregate that crosses a server.

Corollary 3.1. *(Application of Theorem 3.2 to individual flows of an aggregate) Let* s *be a server crossed by two flows* f_1 *and* f_2*. Then, the output of each individual flow* f_i*,* $i \in \{1, 2\}$*, is bounded by* $\tilde{\alpha}^{f_1} + v\left(\alpha^{[f_1,f_2]}, \beta\right)$.

Proof. Let the flows f_1, f_2 have input functions A_1, A_2 and output functions A'_1, A'_2. Assume that α^{f_i} is an arrival curve for flow f_i, $i \in \{1, 2\}$, and that $\alpha^{[f_1,f_2]}$ is an arrival curve for the flow aggregate $[f_1, f_2]$, i.e., for $A = A_1 + A_2$. Further, let α^{f_i} meet the above condition $\alpha^{f_i}(d) = \tilde{\alpha}^{f_i}(d) + \alpha^{f_i}(0^+)$ with $\tilde{\alpha}^{f_i}$ being a sub-additive curve. $B_i : \mathbb{R} \to \mathbb{R}^+$ is the backlog function of flow f_i, i.e., $B_i(t) = A_i(t) - A'_i(t)$. Then, we get for $u \leq t$ the following output of the flow aggregate $[f_1, f_2]$ from s:

$$\begin{aligned}
&\left(A'_1(t) - A'_1(u)\right) + \left(A'_2(t) - A'_2(u)\right) \\
&\quad = ((A_1(t) - B_1(t)) - (A_1(u) - B_1(u))) + \left(A'_2(t) - (A_2(u) - B_2(u))\right) \\
&\quad = A_1(t) - B_1(t) - A_1(u) + B_1(u) + A'_2(t) - A_2(u) + B_2(u) \\
&\quad \text{(remove } -B_1(t)) \\
&\quad \leq A_1(t) - A_1(u) + A'_2(t) - A_2(u) + (B_1(u) + B_2(u))
\end{aligned}$$

Now, let $B_1 + B_2$ be the backlog function of the flow aggregate $[f_1, f_2]$, i.e.,

$$(B_1 + B_2)(u) \leq \sup_{0 \leq r \leq u} \{A(u) - A(r) - \beta(u - r)\},$$

to get

$$\begin{aligned}
&A_1(t) - A_1(u) + A'_2(t) - A_2(u) + (B_1(u) + B_2(u)) \\
&\quad \leq A_1(t) - A_1(u) + A'_2(t) - A_2(u) \\
&\quad\quad + \sup_{0 \leq r \leq u} \{A(u) - A(r) - \beta(u - r)\} \\
&\quad \text{(expand } A \text{ to } A_1 + A_2 \text{)} \\
&\quad \leq A_1(t) - A_1(u) + A'_2(t) - A_2(u) \\
&\quad\quad + \sup_{0 \leq r \leq u} \{A_1(u) + A_2(u) - A_1(r) - A_2(r) - \beta(u - r)\} \\
&\quad = A_1(t) + A'_2(t) + \sup_{0 \leq r \leq u} \{-A_1(r) - A_2(r) - \beta(u - r)\} \\
&\quad = A'_2(t) + \sup_{0 \leq r \leq u} \{A_1(t) - A_1(r) - A_2(r) - \beta(u - r)\} \\
&\quad = A'_2(t) + \sup_{0 \leq r \leq u} \left\{\alpha^{f_1}(t - r) - A_2(r) - \beta(u - r)\right\}
\end{aligned}$$

We now have

$$\begin{aligned}&A'_1(t) - A'_1(u) + A'_2(t) - A'_2(u)\\ &\qquad\leq\ A'_2(t) + \sup_{0\leq r\leq u}\Big\{\alpha^{f_1}(t-r) - A_2(r) - \beta(u-r)\Big\}\end{aligned}$$

Shifting $A'_2(t) - A'_2(u)$ to the right yields

$$A'_1(t) - A'_1(u) \ \leq\ A'_2(u) + \sup_{0\leq r\leq u}\Big\{\alpha^{f_1}(t-r) - A_2(r) - \beta(u-r)\Big\}.$$

From the decomposition of α^{f_i} and sub-additivity of $\tilde{\alpha}^{f_i}$, we get

$$\begin{aligned}\alpha^{f_1}(t-r) &= \alpha^{f_1}(0^+) + \tilde{\alpha}^{f_1}(t-r)\\ &\leq \alpha^{f_1}(0^+) + \tilde{\alpha}^{f_1}(t-u) + \tilde{\alpha}^{f_1}(u-r)\\ &= \tilde{\alpha}^{f_1}(t-u) + \alpha^{f_1}(u-r)\end{aligned}$$

and

$$A'_1(t) - A'_1(u) \ \leq\ A'_2(u) + \tilde{\alpha}^{f_1}(t-u) + \sup_{0\leq r\leq u}\Big\{\alpha^{f_1}(u-r) - A_2(r) - \beta(u-r)\Big\}.$$

Since α^{f_2} is an arrival curve, we know that $-A_2(r) \leq \alpha^{f_2}(u-r) - A_2(u)$ and finally get

$$\begin{aligned}&A'_2(u) + \tilde{\alpha}^{f_1}(t-u) + \sup_{0\leq r\leq u}\Big\{\alpha^{f_1}(u-r) - A_2(r) - \beta(u-r)\Big\}\\ &\leq\ A'_2(u) + \tilde{\alpha}^{f_1}(t-u) + \sup_{0\leq r\leq u}\Big\{\alpha^{f_1}(u-r) + \alpha^{f_2}(u-r) - A_2(u) - \beta(u-r)\Big\}\\ &=\ A'_2(u) - A_2(u) + \tilde{\alpha}^{f_1}(t-u) + \sup_{0\leq r\leq u}\Big\{\alpha^{f_1}(u-r) + \alpha^{f_2}(u-r) - \beta(u-r)\Big\}\\ &\leq\ A'_2(u) - A_2(u) + \tilde{\alpha}^{f_1}(t-u) + v\left(\alpha^{f_1} + \alpha^{f_2},\, \beta\right)\\ &\leq\ \tilde{\alpha}^{f_1}(t-u) + v\left(\alpha^{f_1} + \alpha^{f_2},\, \beta\right)\end{aligned}$$

$\square$

3.4.2 TFA-assisted aggrPBOOAB

Next, we demonstrate how to exploit the basic insight about the alternative output characterization from the previous section. It gives us the choice between the existing PBOO arrival bounding (aggrPBOOAB), which applies the conventional output bound, and an approach where we use a backlog bound for the cross-traffic and apply Theorem 3.2. This backlog bound is obtained from TFA, i.e., it actually considers flows that demultiplex

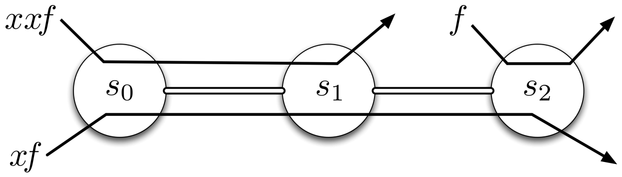


Figure 3.7: Sample network.

from cross-traffic and do not interfere with the foi. In the following we discuss why and when this can actually lead to an improvement.

Consider the network configuration of Figure 3.7 where f is the analyzed foi, xf is its cross-flow and xxf is the cross-traffic of xf. Although the network is depicted as a tandem, we cannot apply a simple tandem analysis because the foi f does not cross all servers, i.e., cross-traffic arrival bounding is necessary in this network: Deriving f's performance bounds with the SFA requires bounding xf's arrival at s_2, $\alpha_{s_2}^{xf}$, with compFFA step 1, aggrPBOOAB, first. Its result is used to separate f by computing f's left-over service curve at s_2 that is then used to derive f's delay bound (compFFA step 2).

AggrPBOOAB retains the worst-case when arbitrarily multiplexing flows, i.e., in contrast to FIFO multiplexing, data of xf may always be served after xxf's data – independent of their relative arrival times. Thus, burstiness of $\alpha_{s_2}^{xf}$, denoted by $b_{s_2}^{xf} := \alpha_{s_2}^{xf}(0^+)$, increases when more data of xxf arrives in shorter intervals, i.e., its arrival curve α^{xxf} increases. In our illustrative numerical evaluation of this section, service curves are chosen to be rate latency functions $\beta_{R,T} = \beta_{20,20}$ and arrival curves are token buckets $\alpha = \gamma_{r,10}$ where the rate r is variable. In this parameterized homogeneous setting, α^{xxf} increases with parameter r that we use to illustrate xf's worst-case burstiness increase with a growing network utilization.

Figure 3.8 shows the utilization's impact on the aggrPBOOAB burstiness of f's cross-traffic, $b_{s_2}^{xf}$, and on the TFA backlog bound at server s_1, $B_{s_1}^{\text{TFA}}$. TFA considers all flows at s_1 and derives the backlog bound based on their aggregate arrival curve. Being the backlog of all incoming traffic at the server, i.e., a superset of f's cross-traffic xf, $B_{s_1}^{\text{TFA}}$ is also a backlog bound for xf. In Figure 3.8, $B_{s_1}^{\text{TFA}}$ scales linearly whereas $b_{s_2}^{xf}$ scales super-linearly with the utilization. Consequently, both curves intersect and $b_{s_2}^{xf}$ exceeds $B_{s_1}^{\text{TFA}}$, such that using the TFA backlog bound and Theorem 3.2 indeed achieves an improvement over aggrPBOOAB. This can be explained by the derivation of the two

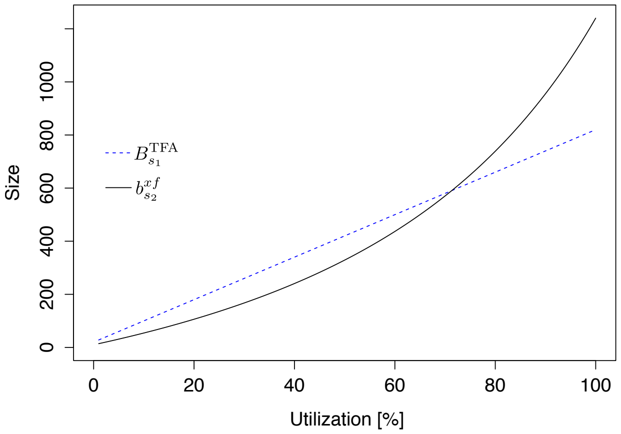


Figure 3.8: Different scaling behaviors of $B_{s_1}^{\mathrm{TFA}}$ and $b_{s_2}^{xf}$ with respect to the network utilization.

values, $B_{s_1}^{\mathrm{TFA}}$ and $b_{s_2}^{xf}$:

$$\begin{aligned}
b_{s_2}^{xf} &= \left(\alpha^{xf} \oslash \beta_{\langle s_0,s_1\rangle}^{\mathrm{l.o.}xf}\right)(0) \\
&= \left(\alpha^{xf} \oslash \left(\beta_{s_0}^{\mathrm{l.o.}xf} \otimes \beta_{s_1}^{\mathrm{l.o.}xf}\right)\right)(0) \\
&= \left(\alpha^{xf} \oslash \left(\left(\beta_{s_0} \ominus \alpha_{s_0}^{xxf}\right) \otimes \left(\beta_{s_1} \ominus \alpha_{s_1}^{xxf}\right)\right)\right)(0) \\
&= \left(\alpha^{xf} \oslash \left(\left(\beta_{s_0} \ominus \alpha^{xxf}\right) \otimes \left(\beta_{s_1} \ominus \left(\alpha_{s_0}^{xxf} \oslash \beta_{s_0}^{\mathrm{l.o.}xxf}\right)\right)\right)\right)(0) \\
&= \left(\alpha^{xf} \oslash \left(\left(\beta_{s_0} \ominus \alpha^{xxf}\right) \otimes \left(\beta_{s_1} \ominus \left(\alpha^{xxf} \oslash \left(\beta_{s_0} \ominus \alpha_{s_0}^{xf}\right)\right)\right)\right)\right)(0) \\
&= (\gamma_{r,10} \oslash ((\beta_{20,20} \ominus \gamma_{r,10}) \otimes (\beta_{20,20} \ominus (\gamma_{r,10} \oslash (\beta_{20,20} \ominus \gamma_{r,10})))))(0) \\
&= \left(\gamma_{r,10} \oslash \left(\beta_{20-r,\frac{410}{20-r}} \otimes \left(\beta_{20,20} \ominus \left(\gamma_{r,10} \oslash \beta_{20-r,\frac{410}{20-r}}\right)\right)\right)\right)(0) \tag{3.5} \\
&= \left(\gamma_{r,10} \oslash \left(\beta_{20-r,\frac{410}{20-r}} \otimes \left(\beta_{20,20} \ominus \gamma_{r,\frac{410r}{20-r}+10}\right)\right)\right)(0) \tag{3.6} \\
&= \left(\gamma_{r,10} \oslash \left(\beta_{20-r,\frac{410}{20-r}} \otimes \left(\beta_{20,20} \ominus \gamma_{r,\frac{400r+200}{20-r}}\right)\right)\right)(0) \tag{3.7} \\
&= \left(\gamma_{r,10} \oslash \left(\beta_{20-r,\frac{410}{20-r}} \otimes \beta_{20-r,\frac{8200}{(20-r)^2}}\right)\right)(0) \tag{3.8} \\
&= \left(\gamma_{r,10} \oslash \beta_{20-r,\frac{410}{20-r}+\frac{8200}{(20-r)^2}}\right)(0) \\
&= \left(\gamma_{r,10} \oslash \beta_{20-r,\frac{16400-410r}{(20-r)^2}}\right)(0) \tag{3.9} \\
&= \frac{9200+410r}{(20-r)^2} r + 10 \\
&= \frac{4000+16000r-400r^2}{400-40r+r^2} \tag{3.10}
\end{aligned}$$

We can see that $b_{s_2}^{xf}$ monotonically increases because the numerator is larger as well as faster growing than the denominator and the stability condition for bounded performance characteristics, $r \leq 10$, leads to an always positive denominator.

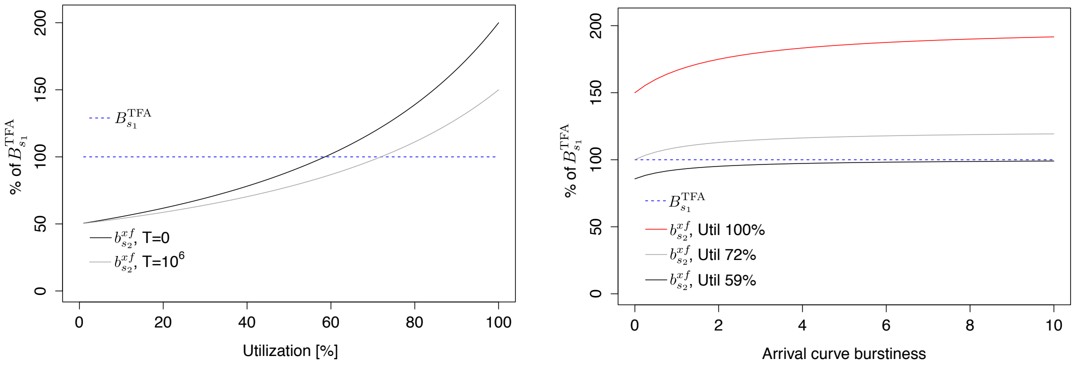
Next, let us see how the polynomial expression's degree builds up during the above derivation. Multiplication by the arrival rate is required to compute the burstiness of an output arrival curve, i.e., every time we deconvolve – see steps from (3.5) to (3.6) and from (3.9) to (3.10). Subsequent left-over service curve operations, e.g., from (3.7) to (3.8), retain the rate in the latency term's denominator, as does the convolution of service curves in the step from (3.8) to (3.9). Deconvolution is required for output bounding and thus occurs at every level of the recursive arrival bounding procedure. In this example, xf is bounded in the first recursion level and it requires bounding xxf in a second level; hence, we obtain a rational function of degree 2 (with a pole at $r = 20$).

The TFA backlog bound derivation for server s_1 proceeds as follows:

$$\begin{aligned}
B_{s_1}^{\text{TFA}} &= v\left(\alpha_{s_1}^{xf} + \alpha_{s_1}^{xxf},\, \beta_{s_1}\right) && (3.11)\\
&= v\left(\left(\alpha_{s_0}^{xf} + \alpha_{s_0}^{xxf}\right) \oslash \beta_{s_0}^{\text{l.o.}[xf,xxf]},\, \beta_{s_1}\right) \\
&= v\left(\alpha_{s_0}^{[xf,xxf]} \oslash \beta_{s_0},\, \beta_{s_1}\right) && (3.12)\\
&= v\left(\left(\gamma_{r,10} + \gamma_{r,10}\right) \oslash \beta_{20,20},\, \beta_{20,20}\right) && (3.13)\\
&= v\left(\gamma_{2r,20} \oslash \beta_{20,20},\, \beta_{20,20}\right) && (3.14)\\
&= v\left(\gamma_{2r,20+2r\cdot 20},\, \beta_{20,20}\right) && (3.15)\\
&= 80r + 20
\end{aligned}$$

The derivation takes advantage of aggregation in (3.11) and (3.13), which prevents recursive cross-traffic arrival bounding in our example – in fact, the TFA implements the PSOO principle, yet, by assuming too much cross-traffic of f. However, xxf is not considered cross-traffic of xf as both belong to the same flow aggregate and therefore no action has to be taken to derive the left-over service curve at s_0 in (3.12). The only relevant deconvolution in $B_{s_1}^{\text{TFA}}$'s derivation is found in the computation of the aggregate's output bound after crossing s_0. The deconvolution in the backlog bounding operation $v\left(\alpha_{s_1}^{xf} + \alpha_{s_1}^{xxf},\, \beta_{s_1}\right)$ executed in the step from (3.14) to (3.15) is, in contrast to the $b_{s_2}^{xf}$-derivation, not affecting the polynomial expression's degree because its latency is not depending on r. Thus, the entire term grows linearly with the flow arrival rate.

Remark 3.1. It is not possible to improve xf's output bound by using the backlog bound for flow xf at server s_1, i.e., $B_{s_1}^{xf}$, because $B_{s_1}^{xf}$ and $b_{s_2}^{xf}$ are equal due to [50], Theorem



(a) Influence of the service curve latency T.

(b) Influence of the arrival curve burstiness b.

Figure 3.9: Relative difference between the TFA backlog bound at s_1, $B_{s_1}^{\text{TFA}}$, and xf's worst-case burstiness derived at s_2, $b_{s_2}^{xf}$.

3.1.12, Rule 12:

$$\begin{aligned} B_{s_1}^{xf} &= \left(\left(\alpha^{xf} \oslash \beta_{s_0}^{\text{l.o.}xf}\right) \oslash \beta_{s_1}^{\text{l.o.}xf}\right)(0) \\ &= \left(\alpha^{xf} \oslash \left(\beta_{s_0}^{\text{l.o.}xf} \otimes \beta_{s_1}^{\text{l.o.}xf}\right)\right)(0) \;=\; b_{s_2}^{xf} \end{aligned}$$

From this reformulated derivation of $b_{s_2}^{xf}$ we obtain another explanation for its function being of degree 1 in the above example: there is only one deconvolution.

Remark 3.2. Theorem 1.4.5 in Le Boudec and Thiran's text book [50] presents conditions for tight output arrival bounds. These are satisfied in both our derivations above, yet, we improve xf's output bound by incorporating $B_{s_1}^{\text{TFA}}$. At first glance, this may seem like a contradiction, however, we gain tightness from additional considerations of a feed-forward analysis that are not addressed in [50], Theorem 1.4.5. It remains valid, yet only with respect to the given service curves that, in turn, might be tightness-compromising left-overs like in Remark 3.1.

In a more involved feed-forward network, we often have multi-level recursions for cross-traffic of cross-traffic in the arrival bounding phase of the derivations [9] – also for the backlog bound at a server – and therefore polynomial expressions of higher degrees occur in both alternative bounds on the output burstiness. For the ease of presentation, we continue to illustrate the impact of the differing scaling behaviors as well as the service curve latency and the initial burstiness of flows in the simple network from Figure 3.7. In Section 3.4.3, we extend our evaluation to more involved feed-forward networks.

Above, we discussed that left-over service curve computations retain the arrival rate in their results' latency term. For instance, the left-over latency at server s_0 is $\frac{T_{s_0}\cdot R_{s_0}+b^{xxf}}{R_{s_0}-r^{xxf}} =$

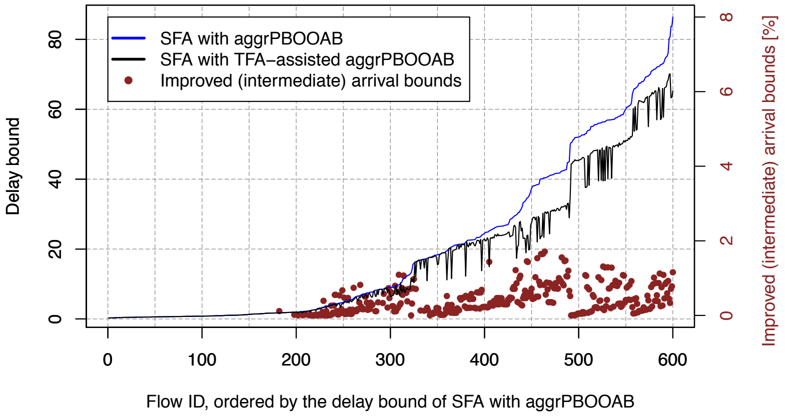
$T_{s_0} + \frac{r^{xxf} \cdot T_{s_0} + b^{xxf}}{R_{s_0} - r^{xxf}}$, i.e, it consists of a fixed and a variable part. The fixed part is defined by the service curves' initial latency $T_{s_0} = T_{s_1} = T_{s_2} =: T$ (equal for all servers in our homogeneous sample network) whose influence on the total burstiness we evaluate – increasing T decreases the impact of the variable part containing the crucial factor r. We check $T = 0$, i.e., the natural lower limit of the latency, and $T = 10^6$, a value several orders of magnitude larger than the service rate $R = 20$ and thus safe to be assumed as a realistic upper bound on T. The resulting range of T's impact is depicted by the relative difference between $B_{s_1}^{\mathrm{TFA}}$ and $b_{s_2}^{xf}$ in Figure 3.9a. Most notably, the network utilization required for the TFA backlog bound to outperform the separated flow's output burstiness is between 59% to 72% – that is, it always exists and resides at utilizations considerably lower than the network's capacity limit. Moreover, $b_{s_2}^{xf}$'s relative benefit of 50% over $B_{s_1}^{\mathrm{TFA}}$ for low utilizations is in fact small in absolute values (cf. Figure 3.8) whereas its disadvantage (right of the intersection) grows fast to become large in absolute numbers.

Last, we evaluate the impact of the remaining variable parameter besides utilization and the service curve latency: the initial burstiness of flows in the homogeneous network, b. We reduced the service curve latency's influence by assigning $\beta = \beta_{20,0.1}$. Arrival curves are $\alpha = \gamma_{r,b}$ where r is defined by the network utilization (i.e., relative to the service rate R) and b is slowly increased from 0 to the previously used value of 10. Figure 3.9b depicts the relative difference between $B_{s_1}^{\mathrm{TFA}}$ and $b_{s_2}^{xf}$ for three levels of network utilization: 59% and 72% (the intersections of both values in the latency evaluation of Figure 3.9a) as well as 100%. We can see that the TFA backlog bound at server s_1 is in fact always within the output burstiness of the same utilizations found for the latency (Figure 3.9a) – for 59%, $B_{s_1}^{\mathrm{TFA}}$ is an asymptote when increasing b, and for 72% the $b_{s_2}^{xf}$-value starts at the server backlog bound. The impact of initial burstiness of flows is similar to the latency's impact. For the maximum network utilization, $b_{s_2}^{xf}$ always exceeds $B_{s_1}^{\mathrm{TFA}}$ by at least 50% in our sample network, i.e., utilization remains most impactful.

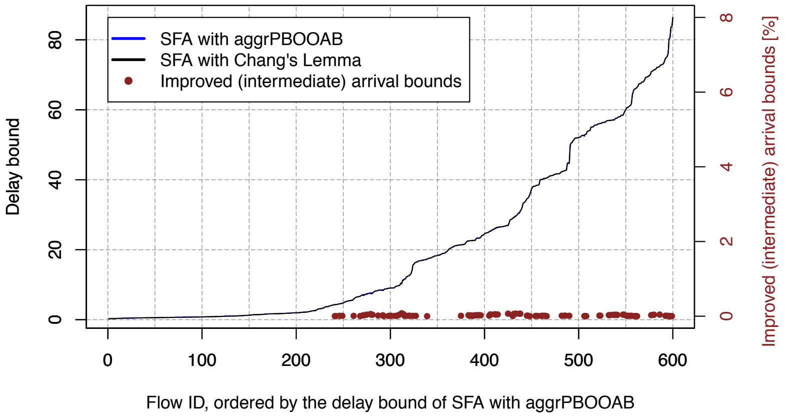
Based on these observations, we propose to improve the arrival bound of a flow (aggregate) with the TFA backlog bound and Theorem 3.2 applied at the last hop of this flow (aggregate) – of course, only if it actually improves the bound. We call this new method: TFA-assisted aggrPBOOAB.

3.4.3 Accuracy Evaluation

The potential improvement of cross-traffic bounds can be quite considerable in the small scenario of Section 3.4.2. Now we turn to the investigation of the impact on the end-to-end delay bound of flows traversing larger feed-forward networks. That is, we evaluate the improvements gained by reduced cross-traffic interference that ultimately tightens delay bounds. We use the SFA to benchmark the new TFA-assisted aggrPBOOAB against the



(a) TFA assistance (Theorem 3.2).



(b) Chang's Lemma [23] did not yield significant improvements, the lines for both SFAs are indistinguishable.

Figure 3.10: Delay analysis with aggrPBOOAB and cross-traffic burstiness improvements.

existing aggrPBOOAB without this improvement (plain aggrPBOOAB of Section 3.3).

Dense Network The first exemplary network we generated for evaluation consists of 150 homogeneous servers with service curves $\beta_{R,T} = \beta_{200,0.1}$. 600 flows with random paths and arrival curve $\alpha = \gamma_{2,0.1}$ were added to the network, i.e., it was chosen to be very dense in order to provoke the effect and thus illustrate the potential of our improvement. They are supposed to randomly generate hotspots of considerable, yet, uncontrolled utilization for the evaluation. These hotspots see the highest numbers of flows such that the impact of separation vs. aggregation can be observed – similar to heterogeneous networks where some flows outweigh others. We chose a small initial burstiness to additionally check the above claim that unavoidable burstiness increases are sufficient to cause impact of the

TFA's assistance to aggrPBOOAB.

The TFA-assisted aggrPBOOAB improved 369 out of 600 flow delay bounds over those derived with plain aggrPBOOAB (see Figure 3.10). In total, 61.5% of flows cross a hotspot that

1. enables the TFA to aggregate flows such that its backlog bounding requires less recursion levels, making it grow slower with the utilization, and
2. has a utilization large enough to allow for its backlog bound to fall below the output bound burstiness.

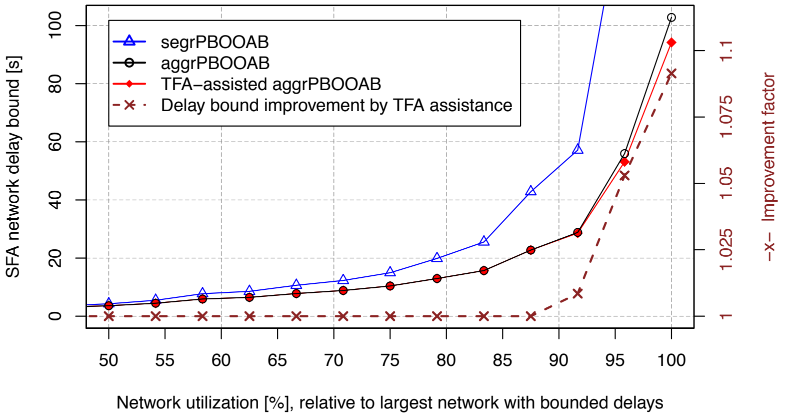
For the 33% of flows with largest delay bound (using plain aggrPBOOAB), we achieve an average flow delay bound reduction of 19.1% (improvement factor 1.237), with a maximum of 44.41% (improvement factor 1.8). The average delay bound reduction across all flows is 15% (factor 1.177).

The distribution of dots for these rightmost 200 flows in Figure 3.10a shows that the improvement is achieved without ever capping more than 2% of the arrival bounds derived during the entire feed-forward analysis (right y-axis). Moreover, it is clearly visible that an increased share of burstiness improvements causes a larger delay bound reduction. For the rightmost 200 flows in Figure 3.10a, the dots form a pattern of three "peaks" whose respective beginning and end both demarcate a step in the improved delay bounds depicted above them.

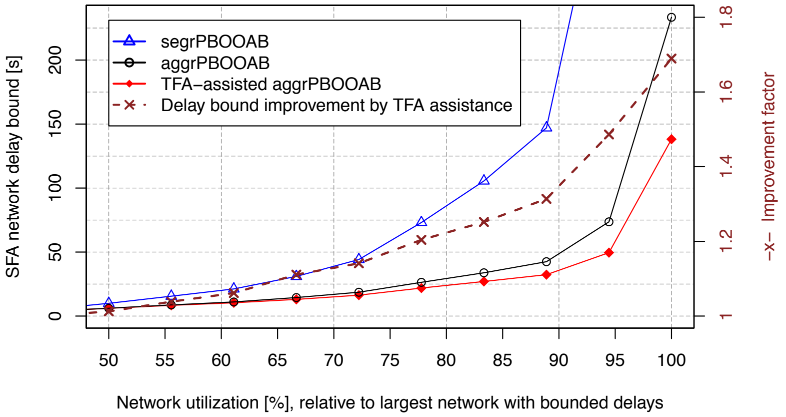
Another interesting observation is that these distinguishable peaks cause a non-uniform decrease of delay bounds. The global network delay bound – the maximum delay bound of all flows in the network – is not defined by the same flow anymore. Applying our new analysis, 11 flows that had a smaller delay bound than this flow now have a larger one. Therefore, the network delay bound decreases by less than the observed maximum for a single flow of 44.41% stated above; it is reduced by 18.75% (factor 1.231) to be precise. This reordering indicates that even when delay bounds are just used as a relative figure of merit, such as in design space explorations [82], an accurate network delay analysis is important and the first step of the compFFA procedure is crucial.

Figure 3.10b shows the results when applying of Chang's lemma to reduce cross-traffic burstiness. As expected, the impact is smaller than with our new theorem, yet, it has nearly no impact on the flows' delay bounds. Although 102 out of 600 flow delay bounds could be improved, the maximum reduction of a delay bound is only 3.25%, the average reduction of these 102 flows is 0.34% and the network delay bound is reduced by 0.02%.

Erdős-Rényi Network Evaluation We conclude the accuracy improvement investigation of compFFA step 1 by continuing the evaluation of Section 3.3.6. We use the same ER networks and benchmark the delay bound accuracy of SFA in compFFA step 2 with the three cross-traffic arrival boundings presented in this thesis, segrPBOOAB, aggrPBOOAB and TFA-assisted aggrPBOOAB (Figure 3.10). The factor on the right y-axis shows the improvement from aggrPBOOAB to TFA-assisted aggrPBOOAB. For the improvement from segrPBOOAB to aggrPBOOAB, please refer to Figure 3.6. In the flat ER network, improvements only start at a very high network utilization of $> 85\%$ and reach a relatively small factor of 1.09 – a reduction of 8.25% of the network delay bound closest to the asymptote. In contrast, the hierarchical ER network experiences larger improvements, showing the potential of the TFA assistance: At a utilization of $\frac{2}{3}$, the improvement is at 1.111 (delay bound reduction of 10%), it grows to factor 1.25 (reduction of 20%) at a utilization of $\frac{5}{6}$ and to factor 1.69 (reduction of 40.8%) at the maximum utilization.



(a) Flat ER network,



(b) Hierarchical ER network.

Figure 3.11: Accuracy evaluation of ER networks with 32 devices each: SFA with three different cross-traffic arrival bounding alternatives, segrPBOOAB, aggrPBOOAB and TFA-assisted aggrPBOOAB.

4 Distributed Sensor Network Calculus Analysis

In this Section, we advance Sensor Network Calculus (SensorNC) by transforming the recursive arrival bounding of Section 3.3 into a more efficient iterative algorithm. Finally, we show how the entire SensorNC analysis can now be deployed within a sensor network. This allows for a distributed in-network performance modeling and admission control scheme that does not require any additional protocol, only small payload additions.

Sensor Network Calculus (SensorNC) is a framework for worst-case analysis of wireless sensor networks and as such allows to derive upper bounds on end-to-end delay for sensor-to-sink data flows as well as the maximum buffer requirements for sensor nodes. SensorNC is strongly based on algNC, but partially slims it down (specific arrival and service curves are used) and partially extends it (compositional sink-tree analysis). Since the initial proposal in [71], it has been extended in several aspects: multiple sinks [76], in-network processing [77], improved delay analysis [35, 77], and [48]. It was applied for diverse purposes, e.g., to model and analyze cluster-tree based IEEE 802.15.4 networks [43, 48], to evaluate traffic splitting in meshed Wireless Sensor Networks (WSNs) [78], or to plan the trajectories of multiple mobile sinks in a large-scale, time sensitive WSN [60].

4.1 Arrivals, Service, (min,+)-Operations and Performance Bounds

Definition 4.1. (SensorNC Arrival and Service Curves) Arrival curves in SensorNC are from the set $\mathcal{F}_{\text{TB}}$ and SensorNC service curves are from the set $\mathcal{F}_{\text{RL}}$.

These restrictions on curves slim down the complexity of SensorNC compared to algNC, yet, they are still effective for modeling WSNs. E.g., sensors reporting measurement values may generate packets of size b that are periodically sent with a minimum inter-arrival time t_δ. Then, the data flow has a maximum data arrival rate of $r = \frac{b}{t_\delta}$ and it can be modeled with the single token bucket $\gamma_{r,b}$. Service curves of $\mathcal{F}_{\text{RL}}$ have been used to model TDMA channel access [35] and duty cycling sensor of nodes [6].

With these simplifications, we can explicitly solve the operations relevant for a SensorNC analysis:

Corollary 4.1. *(Arrival Curve Aggregation in SensorNC) For the aggregation of* n *arrival curves of* $\mathcal{F}_{\text{TB}}$ *it holds that*

$$\sum_{i=1}^{n} \gamma_{r_i,b_i} \quad = \quad \gamma_{\sum_{i=1}^{n} r_i, \sum_{i=1}^{n} b_i}.$$

Corollary 4.2. *(Concatenation of Servers) Consider a flow (aggregate)* $\mathbb{F}$ *crossing a tandem of servers* $\mathcal{T} = \langle s_1, \ldots, s_n \rangle$ *and assume that each* s_i, $i \in \{1, \ldots, n\}$, *offers a service curve* $\beta_{s_i} = \beta_{R_i,T_i} \in \mathcal{F}_{\mathrm{RL}}$. *The overall service curve offered to* $\mathbb{F}$ *is the concatenation of service curves*

$$\beta_{\mathcal{T}} = \beta_{s_1} \otimes \ldots \otimes \beta_{s_n} = \bigotimes_{i=1}^{n} \beta_{s_i} = \bigotimes_{i=1}^{n} \beta_{R_i,T_i} = \beta_{\min_i\{R_i\}, \sum_i T_i}.$$

Corollary 4.3. *(SensorNC Left-Over Service Curve) Consider a server* s *that offers a strict service curve* $\beta = \beta_{R,T} \in \mathcal{F}_{\mathrm{RL}}$. *Let* s *be crossed by flow (aggregate)* $\mathbb{F}_0$ *and flow (aggregate)* $\mathbb{F}_1$ *with arrival curves* $\alpha^{\mathbb{F}_0}, \alpha^{\mathbb{F}_1} \in \mathcal{F}_{\mathrm{TB}}$. *Then* $\mathbb{F}_1$*'s worst-case residual service share under arbitrary multiplexing at* s, *i.e., its left-over service curve in SensorNC at* s, *is*

$$\begin{aligned} \beta^{\mathrm{l.o.}\mathbb{F}_1} &= \beta \ominus \alpha^{\mathbb{F}_0} \\ &= \beta_{R,T} \ominus \gamma_{r^{\mathbb{F}_0}, b^{\mathbb{F}_0}} \\ &= \beta_{R - r^{\mathbb{F}_0},\, T + \frac{b^{\mathbb{F}_0} + r^{\mathbb{F}_0} \cdot T}{R - r^{\mathbb{F}_0}}}. \end{aligned}$$

Corollary 4.4. *(Performance Bounds in SensorNC) Consider a server* s *that offers a service curve* $\beta = \beta_{R,T} \in \mathcal{F}_{\mathrm{RL}}$. *Assume a flow* f *with arrival curve* $\alpha^f = \gamma_{r^f, b^f} \in \mathcal{F}_{\mathrm{TB}}$ *traverses the server. Then, we obtain the following performance bounds for* f*:*

$$\begin{aligned} &\text{(Flow) Delay Bound:} \quad \forall t \in \mathbb{R}^+ : & D^f(t) &\leq T + \tfrac{b^f}{R} \\ & & &= h\left(\alpha^f, \beta\right) =: D^f \\[2ex] &\text{Backlog Bound:} \quad \forall t \in \mathbb{R}^+ : & B^f(t) &\leq b^f + r^f \cdot T \\ & & &= v\left(\alpha^f, \beta\right) =: B^f \\[2ex] &\text{Output Bound:} \quad \forall d \in \mathbb{R}^+ : & \left(\alpha^f\right)'(d) &= \left(\gamma_{r^f, b^f} \dot{\oslash} \beta_{R,T}\right)(d) \\ & & &= \begin{cases} 0 & \text{if } d = 0 \\ \gamma_{r^f,\, b^f + r^f \cdot T}(d) & \text{otherwise.} \end{cases} \end{aligned}$$

The omitted delay bound for flow aggregates (cf. Theorem 2.1) is not required in SensorNC. Its delay analysis always proceeds end-to-end, i.e., source-to-sink, of a single flow. We marked the “deconvolution” with a dot above the operator to indicate that it is explicitly solving SensorNC output bounding operation, i.e., in contrast to the generic

$\oslash$ of Section 2.1.3, it is closed in $\mathcal{F}_0$.

4.2 The Network Model

In addition to a restriction of the arrival and service curves, SensorNC also typically assumes a restriction on the topology space: Network topologies, i.e., device graphs, are limited to sink trees with a single sink (see Figure 4.1). While multiple sinks have been addressed in [76], such topologies can be transformed into a set of sink trees with one sink each. This transformation follows the justification of Section 3.3.5's sample network. Apart from this restriction, we aim for the greatest possible amount of generality. Therefore our considerations neither require sensors to be homogeneous nor impose any restriction on the outdegree or maximum distance to the sink as found in [71, 65, 72, 35]. We use the term node synonymously with server and sensor because we assume that every node provides both functionalities, sensing data and relaying incoming flows. This is in contrast to previous work such as [48] and [43] that assume cluster trees. They explicitly distinguish between nodes in the role of a sensor and those acting as servers. Each role has its respective restrictions on aspects like connectivity.

The topology restriction has another simplifying indication: Each node only possesses a single, well-defined next hop in the device graph. Therefore, the conversion to the server graph is straight-forward. Each device corresponds to a server, except the sink device. Flows do not cross an output queue at this device. In the following, we assume an unrestricted sink, i.e., a sink device that offers infinite service [6, 8]. Thus, it behaves like a neutral element in the analysis and we can assume that device and server graph are indeed equal in SensorNC.

In WSNs, nodes usually possess a single transceiver, i.e., received data flows need not cross a switching fabric to reach the output buffer. This consideration seems to invalidate our justification for arbitrary multiplexing analysis presented in Section 2.1.2. Yet, arbitrary multiplexing analysis with SensorNC was validated using real-world experiments with MicaZ sensor nodes [65]. The attained performance bounds are shown to be accurate such that it is justified to continue relying on this analysis assumption.

4.3 Compositional Sink-tree Analysis

4.3.1 Flow of Interest Analysis

The sink-tree networks of SensorNC guarantee for nested interference in the foi analysis of compFFA step 2. In this setting, we can apply the according PMOOA which is known to outperform the SFA. In fact, in [74] it is shown that the PMOOA actually results in the tight delay bounds if the left-over service rates on the foi's path are monotonically

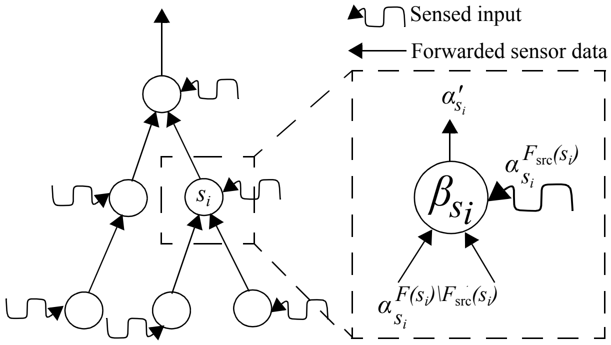

Sensed input
Forwarded sensor data
s_i
α'_{s_i}
$\alpha^{F_{\mathrm{src}}(s_i)}_{s_i}$
β_{s_i}
$\alpha^{F(s_i)\backslash F_{\mathrm{src}}(s_i)}_{s_i}$

Figure 4.1: Sensor Network Model [71].

Quantifier	Definition
$up(s)$	Set of servers 1 hop upstream of s, i.e., further away from the sink
$P(f, i)$	Server at location (index) i on f's path
$L(f, s)$	Location (index) of server s on f's path $P(f)$

Table 4.1: Sensor Network Calculus Notation extending Table 2.1.

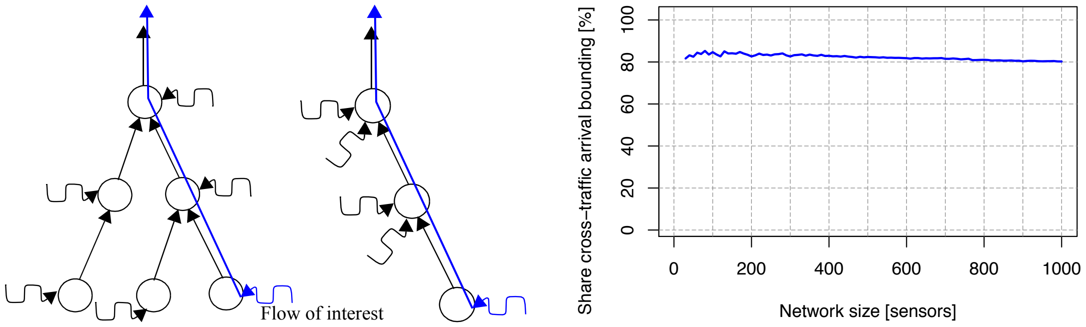
decreasing and cross-traffic arrival bounds are tight as well. SensorNC provides a specialized PMOOA that exploits the sink-tree nesting [35, 77] where multiplexed cross-flows do not demultiplex from the foi. Table 4.1 summarizes the notation required to precisely quantify all parameters involved.

Algorithm 4.1. *(SensorNC Sink-tree PMOOA Left-over Service Curve) Let a network with unrestricted sink be crossed by flow* foi *and let* $P(\mathrm{foi}) = \langle s_0, \ldots, s_k \rangle$ *be its path from its source* s_0 *to the sink* s_k*. Then, the SensorNC sink-tree PMOOA left-over service curve is derived as follows:*

$$\begin{aligned} \beta^{\mathrm{l.o.foi}} &= \beta^{\mathrm{l.o.foi}}_{s_0}, \\ \beta^{\mathrm{l.o.foi}}_{s_i} &= \left(\beta^{\mathrm{l.o.foi}}_{s_{i+1}} \otimes \beta_{s_i}\right) \ominus \left(\alpha^{F_{\mathrm{src}}(s_i)\backslash \mathrm{foi}}_{s_i} + \sum_{s_j \in up(s_i)\backslash s_{i-1}} \alpha'_{s_j}\right) \end{aligned}$$

where

- $i \in \{0, \ldots, k-1\}$,
- β_{s_i} *denotes the service curve at server* s_i,
- *the sink's service curve* β_{s_k} *is* $\delta_0(t) = \begin{cases} 0 & \text{if } t = 0 \\ +\infty & \text{otherwise} \end{cases}$,



(a) Abstraction by cross-traffic arrival bounding.

(b) Observed share of total analysis time.

Figure 4.2: Cross-Traffic Arrival Bounding: (a) tandem abstraction by cross-traffic arrival bounding; (b) observed share this step takes of the total analysis time (mean of 40 $(5, 20)$-constrained random sink trees growing in size [72]).

- $\alpha_{s_i}^{F_{src}(s_i)\setminus foi}$ *is the arrival of cross-flows originating at* s_i*, and*
- $\sum_{s_j \in up(s_i)\setminus s_{i-1}} \alpha'_{s_j}$ *is the sum of cross-flow arrivals from servers one hop upstream of* s_i *but not crossed by the* foi *itself,* $s_{-1} := s_0$*. Each* α'_{s_j} *is currently computed with Algorithm 3.1, aggrPBOOAB.*

The SensorNC PMOOA $\beta^{\text{l.o.}}$ thus differs from generic nested-interference PMOOA for the sink tree of Figure 2.2 (Section 2.3.3). Applying Algorithm 4.1, the derivation proceeds from the foi's source server to the sink, server-by-server, (foi $:= f_1$):

$$\begin{aligned}
\beta^{\text{l.o.foi}} &= \beta_{s_0}^{\text{l.o.foi}} \\
&= \beta_{s_1}^{\text{l.o.foi}} \otimes \beta_{s_0} \\
&= \left(\left(\beta_{s_2}^{\text{l.o.foi}} \otimes \beta_{s_1}\right) \ominus \alpha^{f_0}\right) \otimes \beta_{s_0} \\
&= \left(\left(\left(\beta_{s_5}^{\text{l.o.foi}} \otimes \beta_{s_2}\right) \otimes \beta_{s_1}\right) \ominus \alpha^{f_0}\right) \otimes \beta_{s_0} \\
&= \left(\left(\left(\left(\left(\beta_{s_6}^{\text{l.o.foi}} \otimes \beta_{s_5}\right) \ominus \alpha_{s_5}^{f_2}\right) \otimes \beta_{s_2}\right) \otimes \beta_{s_1}\right) \ominus \alpha^{f_0}\right) \otimes \beta_{s_0} \\
&= \left(\left(\left(\left(\left(\left(\delta_0 \otimes \beta_{s_6}\right) \otimes \beta_{s_5}\right) \ominus \alpha_{s_5}^{f_2}\right) \otimes \beta_{s_2}\right) \otimes \beta_{s_1}\right) \ominus \alpha^{f_0}\right) \otimes \beta_{s_0} \\
&= \left(\left(\left(\left(\left(\beta_{s_6} \otimes \beta_{s_5}\right) \ominus \alpha_{s_5}^{f_2}\right) \otimes \beta_{s_2}\right) \otimes \beta_{s_1}\right) \ominus \alpha^{f_0}\right) \otimes \beta_{s_0}
\end{aligned}$$

Using the commutativity of $\otimes$, we can, however, rearrange the service curves to match Section 2.3.3's term $\left(\left(\left(\left(\beta_{s_5} \otimes \beta_{s_6}\right) \ominus \alpha_{s_5}^{f_2}\right) \otimes \left(\beta_{s_1} \otimes \beta_{s_2}\right)\right) \ominus \alpha^{f_0}\right) \otimes \beta_{s_0}$.

4.3.2 Cross-traffic Arrival Bounding

In contrast to the SensorNC sink-tree PMOOA for compFFA step 2, there is no specialized SensorNC cross-traffic arrival bounding for step 1. As implied by the original SensorNC paper [71], it is assumed to be performed in a recursive manner, which can actually yield sub-optimal bounds [73]. Algorithm 4.2 depicts this cross-traffic arrival bounding. It is a simplified version of Algorithm 3.1 but it is not tailored specifically to SensorNC. Moreover, cross-traffic arrival bounding in sink trees actually consumes a high fraction of the computation time when analyzing a network. To illustrate this, we created 40 different $(5, 20)$-constrained random sink trees, i.e., sink trees with a maximum outdegree of 5 and a maximum depth of 20, that we grew from 10 servers to 1000 servers with a stepsize of 10 servers. Figure 4.2b shows the fraction cross-traffic arrival bounding took on average over the 40 equally sized and constrained networks. It can be observed that arrival bound computations consume roughly 80% of the overall runtime of the analysis – independent of the network size. Thus, improvements in the arrival bound computation are a very promising candidate to tune the performance of SensorNC computations.

Algorithm 4.2. *(Recursive Aggregate Sink-Tree Arrival Bounding) We derive $\alpha_s^{\mathbb{F}}$, i.e., a bound on flow aggregate $\mathbb{F}$ at server s. Without loss of generality we assume that $\alpha_s^{\mathbb{F}} = \alpha_s$ and that server s has at least one subtree with at least two levels. Also let $n = \max_{f \in \mathbb{F}} \{L(f, s)\}$. Then, the recursive aggregate arrival bound is computed as*

$$
\begin{aligned}
\alpha_s &= \sum_{s_1 \in up(s)} \left(\alpha_{s_1} \dot{\oslash} \beta_{s_1}\right) + \alpha^{F_{\mathrm{src}}(s)} \\
&= \sum_{s_1 \in up(s)} \left(\left(\sum_{s_2 \in up(s_1)} \left(\alpha_{s_2} \dot{\oslash} \beta_{s_2}\right) + \alpha^{F_{\mathrm{src}}(s_1)} \right) \dot{\oslash} \beta_{s_1} \right) + \alpha^{F_{\mathrm{src}}(s)} \\
\dots \\
&= \sum_{s_1 \in up(s)} \left(\dots \left(\sum_{s_n \in up(s_{n-1})} \left(\alpha_{s_n} \dot{\oslash} \beta_{s_n}\right) + \alpha^{F_{\mathrm{src}}(s_{n-1})} \right) \dots \dot{\oslash}\, \beta_{s_1} \right) + \alpha^{F_{\mathrm{src}}(s)}
\end{aligned}
$$

When the topology is traversed, intermediate arrival bounds have to be computed at every server where flows merge. This needs to be done recursively, in a separate compFFA step 1. Note, that although the PSOO principle cannot be violated in sink trees, the LPA can derive more accurate delay bounds than the compositional algNC analysis. It relates the start of backlogged periods on different branches instead of relying on worst-case assumptions.

4.4 Improving SensorNC Analysis

In this section, we generalize the classical concatenation theorem (Theorem 2.2) that is only applicable to a limited set of networks: The tandem of servers that is crossed by flows entirely from end to end. Otherwise the binary (min,+)-convolution $\otimes$ cannot be used within the analysis. Naturally, networks are more complex than such simple 1 : 1 input/output systems and decomposition into tandems is required (Section 3.1).We provide a generalized version of the concatenation theorem for arbitrary sink-tree networks in SensorNC – exactly accounting for flow entanglement in these $n : 1$ input/output systems while preserving accuracy and achieving a high reduction of complexity and computational effort.

4.4.1 The Generalized Concatenation for SensorNC

We use the following two corollaries. First, we show that the SensorNC output bound $\dot{\oslash}$ is distributive w.r.t. $+$.

Corollary 4.5. *(Distributivity of $\dot{\oslash}$ with respect to $+$) For any $\alpha^{f_1}, \alpha^{f_2} \in \mathcal{F}_{\mathrm{TB}}$ and $\beta \in \mathcal{F}_{\mathrm{RL}}$ it holds that*

$$\left(\alpha^{f_1} + \alpha^{f_2}\right) \dot{\oslash} \beta \quad = \quad \alpha^{f_1} \dot{\oslash} \beta + \alpha^{f_2} \dot{\oslash} \beta.$$

Proof. This is a direct consequence from the general distributivity of $\oslash$ over $+$ (Lemma 3.2). Nonetheless, we provide a streamlined proof for the restricted set of curves in SensorNC.

Let $\alpha^{f_1} = \gamma_{r_1,b_1}$, $\alpha^{f_2} = \gamma_{r_2,b_2}$ and $\beta = \beta_{R,T}$. From Corollary 4.4 it follows that

$$\begin{aligned}\left(\left(\alpha^{f_1} + \alpha^{f_2}\right) \dot{\oslash} \beta\right)(d) &= \left(\left(\gamma_{r_1,b_1} + \gamma_{r_2,b_2}\right) \dot{\oslash} \beta_{R,T}\right)(d) \\ &= \left(\gamma_{r_1+r_2,\, b_1+b_2} \dot{\oslash} \beta_{R,T}\right)(d).\end{aligned}$$

If $d = 0$ we have $\alpha^{f_1} \dot{\oslash} \alpha^{f_2}(d) = 0$ and for $d > 0$ we get

$$\begin{aligned}\left(\gamma_{r_1+r_2,\, b_1+b_2} \dot{\oslash} \beta_{R,T}\right)(d) &= \left(\gamma_{r_1+r_2,\, (b_1+b_2)+(r_1+r_2)\cdot T}\right)(d) \\ &= \left(\gamma_{r_1+r_2,\, b_1+r_1\cdot T+b_2+r_2\cdot T}\right)(d) \\ &= \left(\gamma_{r_1,\, b_1+r_1\cdot T} + \gamma_{r_2,\, b_2+r_2\cdot T}\right)(d) \\ &= \left(\gamma_{r_1,\, b_1+r_1\cdot T}\right)(d) + \left(\gamma_{r_2,\, b_2+r_2\cdot T}\right)(d) \\ &= \left(\gamma_{r_1,b_1} \dot{\oslash} \beta_{R,T}\right)(d) + \left(\gamma_{r_2,b_2} \dot{\oslash} \beta_{R,T}\right)(d) \\ &= \left(\alpha^{f_1} \dot{\oslash} \beta\right)(d) + \left(\alpha^{f_2} \dot{\oslash} \beta\right)(d).\end{aligned}$$

□

The composition rule of $\dot{\oslash}$ follows from $f \oslash g \oslash h = f \oslash (g \otimes h)$ [50] by an argumentation akin to Corollary 4.5.

Corollary 4.6. *(Composition of $\dot{\oslash}$) For $f, g, h \in \mathcal{F}_0$ it holds that*

$$f \dot{\oslash} g \dot{\oslash} h \;=\; f \dot{\oslash} (g \otimes h)\,.$$

Theorem 4.1. *(SensorNC Concatenation Theorem) Consider a set of flows $\mathbb{F}$, $|\mathbb{F}| = n$, with arrival curves $\alpha^{f_1}, \ldots, \alpha^{f_n} \in \mathcal{F}_{\mathrm{TB}}$. For the purpose of aggregate output bound computation from their* sink*, the share of service offered to each flow $f \in \mathbb{F}$ within this aggregate is the concatenation of the service curves on its path. Then, the entire flow aggregate's output is bounded by*

$$\alpha'_{\mathrm{sink}} \;=\; \sum_{f \in F(\mathrm{sink})} \left(\alpha^f \dot{\oslash} \bigotimes_{i=0}^{L(f,\mathrm{sink})} \beta_{P(f,i)} \right).$$

Applying Corollary 4.4, we can rephrase the equation to

$$\alpha'_{\mathrm{sink}} \;=\; \gamma_{r'_{\mathrm{sink}}, b'_{\mathrm{sink}}}$$

with

$$r'_{\mathrm{sink}} \;=\; \sum_{f \in F(\mathrm{sink})} r^f$$

$$b'_{\mathrm{sink}} \;=\; \sum_{f \in F(\mathrm{sink})} \left(b^f + r^f \cdot \sum_{i=1}^{L(f,\mathrm{sink})} T_{P(f,i)} \right).$$

Proof. First, we apply Corollary 4.4 to derive the tree's output bound from the flows at the sink.

$$\alpha'_{\mathrm{sink}} \;=\; \sum_{f \in F(\mathrm{sink})} \left(\alpha^f_{\mathrm{sink}} \dot{\oslash} \beta_{\mathrm{sink}} \right)$$

Next, we *virtually* separate all flows from each other and establish a tandem topology in their respective point of view. We recursively apply Corollaries 4.4, 4.5 and 4.6 to the subtree defining the involved output arrival bounds. Note, that every server sees all flows crossing the subtree above as there is no demultiplexing in sink trees (see Figure 4.3a). We start with the separation of a single flow f: Without loss of generality assume $L(f, \mathrm{sink}) \geq 2$ and let the path of f be $P(f) = \left\langle s^f_n, \ldots, s^f_0 \right\rangle$, with $s^f_0 = s_0 = s_{\mathrm{sink}}$ (For the ease of presentation, server numbering is reversed compared to Algorithm 4.1).

$$
\begin{aligned}
\alpha'_{\text{sink}} &= \alpha'_{s_0} \\
&= \alpha_{s_0} \dot{\oslash} \beta_{s_0} \\
&= \left(\alpha^{F_{\text{src}}(s_0)} + \sum_{s_1 \in up(s^f_0)} \left(\alpha_{s_1} \dot{\oslash} \beta_{s_1} \right) \right) \dot{\oslash} \beta_{s_0}
\end{aligned}
$$

(Separate the path taken by f)

$$
= \left(\alpha^{F_{\text{src}}(s_0)} + \alpha_{s^f_1} \dot{\oslash} \beta_{s^f_1} + \sum_{s_1 \in up(s^f_0) \setminus s^f_1} \left(\alpha_{s_1} \dot{\oslash} \beta_{s_1} \right) \right) \dot{\oslash} \beta_{s_0}
$$

(Distributivity of $\dot{\oslash}$ w.r.t. $+$: Separate f from the aggregate)

$$
\begin{aligned}
&= \left(\alpha_{s^f_1} \dot{\oslash} \beta_{s^f_1} \right) \dot{\oslash} \beta_{s_0} + \left(\alpha^{F_{\text{src}}(s_0)} + \sum_{s_1 \in up(s^f_0) \setminus s^f_1} \left(\alpha_{s_1} \dot{\oslash} \beta_{s_1} \right) \right) \dot{\oslash} \beta_{s_0} \\
&= \alpha_{s^f_1} \dot{\oslash} \left(\beta_{s^f_1} \otimes \beta_{s^f_0} \right) + \left(\alpha^{F_{\text{src}}(s_0)} + \sum_{s_1 \in up(s^f_0) \setminus s^f_1} \left(\alpha_{s_1} \dot{\oslash} \beta_{s_1} \right) \right) \dot{\oslash} \beta_{s_0} \\
&= \alpha_{s^f_1} \dot{\oslash} \bigotimes_{i=0}^{1} \beta_{s^f_i} + \left(\alpha^{F_{\text{src}}(s_0)} + \sum_{s_1 \in up(s^f_0) \setminus s^f_1} \left(\alpha_{s_1} \dot{\oslash} \beta_{s_1} \right) \right) \dot{\oslash} \beta_{s_0}
\end{aligned}
$$

(Backtrack f only)

$$
\begin{aligned}
&= \alpha^f \dot{\oslash} \bigotimes_{i=0}^{L(f,\text{sink})} \beta_{s^f_i} \\
&\quad + \left(\alpha^{F_{\text{src}}(s_n)} + \sum_{s_{n+1} \in up(s^f_n)} \left(\alpha_{s_{n+1}} \dot{\oslash} \beta_{s_{n+1}} \right) \right) \dot{\oslash} \bigotimes_{i=0}^{L(f,\text{sink})} \beta_{s^f_i} \\
&\quad \ldots \\
&\quad + \left(\alpha^{F_{\text{src}}(s_0)} + \sum_{s_1 \in up(s^f_0) \setminus s^f_1} \left(\alpha_{s_1} \dot{\oslash} \beta_{s_1} \right) \right) \dot{\oslash} \beta_{s_0}
\end{aligned}
$$

(Distributivity of $\dot{\oslash}$ w.r.t. $+$: Aggregate flows other than f)

$$
= \alpha^f \dot{\oslash} \bigotimes_{i=0}^{L(f,\text{sink})} \beta_{s^f_i} + \alpha^{F(s_0) \setminus f}_{s_0} \dot{\oslash} \beta_{s_0}
$$

This separation of a single flow leads to an intermediate view on the sink tree as

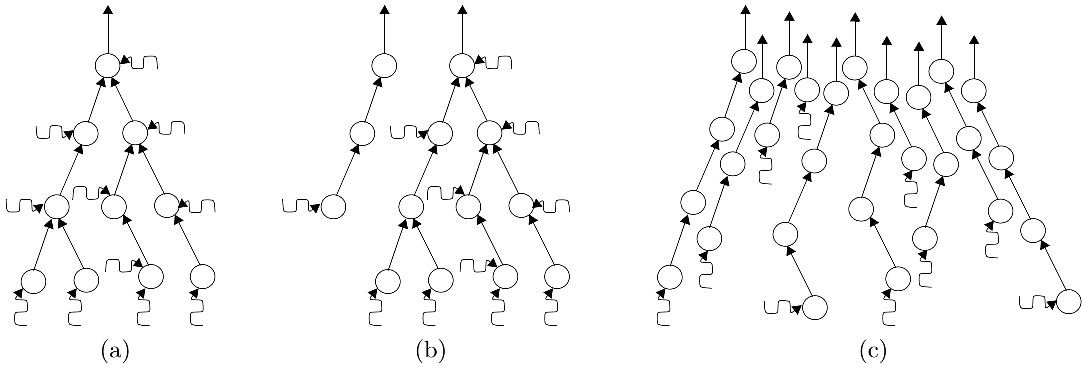

(a) (b) (c)

Figure 4.3: Cross-flow separation in a sink-tree network with Theorem 4.1:
(a) starting with a network that aggregates flows towards the sink,
(b) to separating a single flow,
(c) to the separation of all flows from each other.
Separation crucially differs from segregation as it does not require left-over service derivations.

depicted in Figure 4.3b. An alternative derivation can be found in [8].
Next, we repeat the separation for the remaining flows in $F(\text{sink})$ until we finally reach:

$$\alpha'_{\text{sink}} = \sum_{f \in F(\text{sink})} \left(\alpha^f \dot{\oslash} \bigotimes_{i=0}^{L(f,\text{sink})} \beta_{P(f,i)} \right).$$

This view on the network corresponds to Figure 4.3c.
The SensorNC specific calculation follows from Corollaries 4.1 and 4.4:

$$\begin{aligned} \alpha'_{\text{sink}} &= \gamma_{r'_{\text{sink}}, b'_{\text{sink}}} \\ &= \sum_{f \in F} \gamma'_{r^f_{\text{sink}}, b^f_{\text{sink}}} \\ &= \gamma_{\sum_{f \in F} \left(r^f_{\text{sink}}\right)', \sum_{f \in F} \left(b^f_{\text{sink}}\right)'} \end{aligned}$$

with

$$\sum_{f \in F(\text{sink})} \left(b^f_{\text{sink}}\right)' = \sum_{f \in F(\text{sink})} \left(b^f + r^f \cdot \sum_{i=0}^{L(f,\text{sink})} T_{P(f,i)} \right).$$

$\square$

The simple tandem topology lacking any nesting of flows is a special sink-tree network where the $n : 1$ input/output relation is instantiated with $n = 1$. In this case, the generalized concatenation theorem for SensorNC is specialized to

$$\alpha_{\text{sink}}^{'} \quad = \quad \alpha^f \dot{\oslash} \bigotimes_{i=0}^{L(f,\text{sink})} \beta_{P(f,i)} \, ,$$

i.e., the output bound of Theorem 2.1 with the concatenated service curve for tandems of servers (Theorem 2.2).

Note, that we *virtually* separated the flows from each other within the analysis, which is different to the cross-flow segregation of Section 3.3.1. It does no require to derive left-over service curves, i.e., the flows do not suffer from mutual interference assumptions that violate the PSOO principle and thus decrease accuracy. We call this new property *flow-locality*. As we show next, it transforms the iterative cross-traffic arrival bounding of Algorithm 4.2 into a two-tier iteration over the cross-flows and their respective paths.

4.4.2 The New SensorNC Cross-traffic Arrival Bounding

In this section, we exploit the SensorNC concatenation theorem. First, we need to extend its capabilities to derive bounds for subsets of flows in order to fulfill the PMOOA's requirement to only analyze a specific foi – similar to our new output theorem of Section 3.4.1 that has been extended by an according corollary.

Corollary 4.7. *We can apply Theorem 4.1 to a subset of flows* $F \subseteq F\,(\text{sink})$ *with*

$$\left(\alpha^F{}_{\text{sink}}\right)' \quad = \quad \sum_{f \in F} \left(\alpha^f \dot{\oslash} \bigotimes_{i=0}^{L(f,\text{sink})} \beta_{P(f,i)} \right)$$

if the remaining flows in $F(\text{sink}) \backslash F$ *have a lower priority than those in* F.

In arbitrary multiplexing tandem analyses of compFFA step 1, such as the PMOOA, the foi is always considered to have the lowest priority among all flows. This assumption preserves the worst-case semantic of (Sensor)NC.

Equipped with the new SensorNC sink-tree concatenation theorem and Corollary 4.7, we derive a novel iterative algorithm to perform the aggrPBOOAB in SensorNC.

Theorem 4.2. *(SensorNC Arrival Bounding) Given the flow of interest* foi, *we can derive its cross-traffic arrival bound at any server* s *on its path* $P(\text{foi}) = \langle s_n, \ldots, s_0 \rangle$, $s_0 = s_{\text{sink}}$, *as follows:*

$$\alpha_s^{x(\text{foi})} \quad = \quad \sum_{f \in x(\text{foi}) \backslash F_{\text{src}}(s)} \left(\alpha^f \dot{\oslash} \bigotimes_{i=1}^{L(f,s)} \beta_{P(f,i)} \right) + \alpha^{F_{\text{src}}(s) \cap x(\text{foi})}$$

Applying Theorem 4.4, we can rephrase the equation to

$$\alpha_s^{x(\text{foi})} = \gamma_{r_s^{x(\text{foi})}, b_s^{x(\text{foi})}}$$

with

$$\begin{aligned} r_s^{x(\text{foi})} &= \sum_{f \in x(\text{foi})} r^f \\ b_s^{x(\text{foi})} &= \sum_{f \in x(\text{foi}) \setminus F_{\text{src}}(s)} \left(b^f + r^f \cdot \sum_{i=1}^{L(f,s)} T_{P(f,i)} \right) + b^{F_{\text{src}}(s) \cap x(\text{foi})}. \end{aligned}$$

Proof. The cross-traffic aggregate at s, $\alpha_s^{x(\text{foi})}$, consists of the sum of all cross-flows arriving from the subtrees upstream from s as well as the cross-flows originating at s:

$$\begin{aligned} \alpha_s^{x(\text{foi})} &= \sum_{s_1 \in up(s)} \left(\sum_{f \in F(s_1) \cap x(\text{foi})} \left(\alpha^f \dot{\oslash} \bigotimes_{i=0}^{L(f,s_1)} \beta_{P(f,i)} \right) \right) + \alpha^{F_{\text{src}}(s) \cap x(\text{foi})} \\ &= \sum_{f \in x(\text{foi}) \setminus F_{\text{src}}(s)} \left(\alpha^f \dot{\oslash} \bigotimes_{i=1}^{L(f,s)} \beta_{P(f,i)} \right) + \alpha^{F_{\text{src}}(s) \cap x(\text{foi})} \end{aligned}$$

□

Using the new bounding method from Theorem 4.2 instead of the conventional one from Algorithm 4.2 has several practical advantages. We conclude this section by discussing the most important advantages before evaluating their impact.

Faster Computation The arrival bounds for cross-traffic, $\alpha_{s_i}^{x(\text{foi})}$, $s_i \in P(\text{foi})$, naturally depend on the foi. In a self-modeling WSN, a sensor node needs to compute these bounds for each flow to keep track of their state. With Theorem 4.2, we reduce the computational effort of arrival bounding by exploiting flow-locality. Aggregation of individual cross-flow arrival bounds is *virtually* shifted from servers in the subtree above s_i to s_i itself, where the bound is required. There, flow-local results are simply aggregated. This allows SensorNC to reuse these results in the derivation of any other foi's cross-traffic arrival bounds at s_i. In contrast, the recursive Algorithm 4.2 is tightly interwoven with the topology such that it is generally not possible to share any results between the derivations for different fois.

Lower Communication Overhead Flow-locality also enables to overcome the need for an additional data dissemination protocol such as Deluge [40] that distributes the information required to derive $\alpha_{s_i}^{x(\text{foi})}$. If virtually separated, a flow's arrivals at any server s_i can be calculated in a server-by-server fashion without compromising accuracy (Corollary 4.6). A flow can carry information about its current arrival bound as payload, pushing the information to all sensors concerned. Thus, the state is updated on demand, i.e., independent of a polling interval, resulting in much less communication (more details will be presented in Section 4.5).

Quick Reaction to Changes The flow-locality also affects the recomputation effort in case of parameter modifications. Using the conventional method based on Algorithm 4.2, locality of a modified parameter did not matter much due to the tree structure of flow aggregation locations; a change to a single parameter always invalidated a large amount of the derivation's intermediate results and triggered expensive recomputations, usually of the entire subtree. Theorem 4.2 prevents such an invalidation from spreading to flows not directly affected by a change, e.g., flows not crossing a sensor that adapted its rate, and thus enables a quick reaction to changes.

4.4.3 Accuracy Evaluation

In this section, we benchmark our results against existing SensorNC analysis. The evaluation setting is as follows: As the PMOOA outperforms SFA in sink trees, we compute the SensorNC sink-tree PMOOA left-over service curve (Algorithm 4.1) to derive the delay bound (compFFA step 2). In the compFFA's first step, we execute three different alternatives, the conventional aggrPBOOAB (Algorithm 3.1), our new SensorNC aggrPBOOAB (Theorem 4.2), and a segregated PMOO cross-traffic arrival bounding, segrPMOOAB. The latter one was chosen to illustrate the difference between separation and segregation. Segregation also results in bounds possessing flow-locality, but not a flow-local derivation. I.e., their derivation violates the PSOO principle and thus illustrates the negative impact of the mutual interference problem in the composition penalty. Figure 4.4 shows the resulting delay bounds for a random $(5, 20)$-constrained, i.e., maximum outdegree of 5 and maximum depth of 20 [72], homogeneous sink tree with 100 servers, unit size arrivals $\alpha = \gamma_{1,1}$ and sufficiently large service $\beta = \beta_{75,1}$.

Figure 4.4 shows the following results: Despite being flow-local, the generalized, concatenation-based aggrPBOOAB preserves the tightness of the recursive version whereas segrPMOOAB results in a significant increase of flow delay bounds. The first observation is a consequence of Corollary 4.5 and Theorem 4.1. The second one, segrPMOOAB's inferiority arises from segregating cross-flows and violating the PSOO principle: There is no prioritization among flows in a cross-traffic aggregate. Therefore, they need to be considered to mutually interfere in the worst possible way when applying segrPMOOAB.

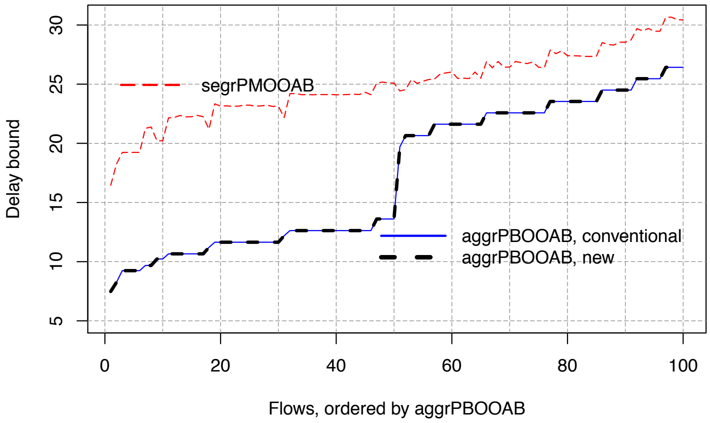


Figure 4.4: Delay bounds using different cross-traffic arrival bounding methods.

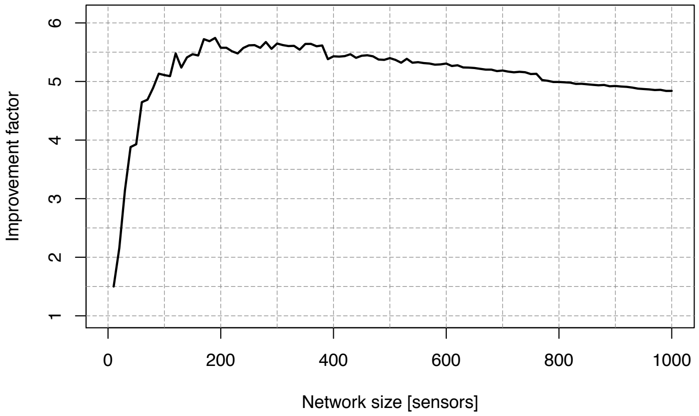
That leads to overly pessimistic cross-traffic arrivals and thus compromises accuracy of delay bounds.

4.4.4 Effort Evaluation

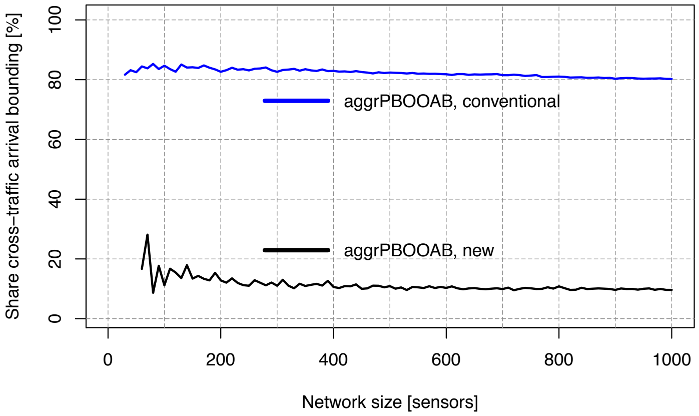
SensorNC analyses have usually been carried out by a central entity due to the knowledge about and control over the information needed to derive bounds. The most prominent example is design space exploration that evaluates the effect of different system configurations on the performance, measured by the network delay bound. We investigate the impact of Theorem 4.2 on the execution time of an end-to-end analysis of all flows in the network. We have created 40 random $(5, 20)$-constrained sink trees that we grew from 10 servers to 1000 servers with a stepsize of 10 servers. Each server offers a strict service curve $\beta_{1000,1}$ and generates a flow shaped to the arrival curve $\alpha_{0.001,0.1}$. This setting guarantees for bounded delays in all our networks. All computations were executed on a server equipped with an Intel Xeon E5420 CPU and 12 GB of RAM.

Relative Improvements Figure 4.5a shows the improvement factor for computation times. Starting with small network sizes, the effort needed to bound cross-traffic arrivals naturally increases with the number of servers and therefore the gain by our new method is getting larger, too. However, after this initial phase it slowly decreases, reaching a value of ~ 4.84 as the mean over our 40 networks with 1000 servers each. On average (over all experiments) the new SensorNC analysis was ~ 5 times faster.

Figure 4.5b illustrates the according reduction of the arrival bounding's share of the overall network analysis time: from 80% to less than 15%. Especially for small sink trees the reduction highly depends on the actual shape of the tree, which explains the visible oscillations at the beginning.



(a) Improvement factor w.r.t. computation time

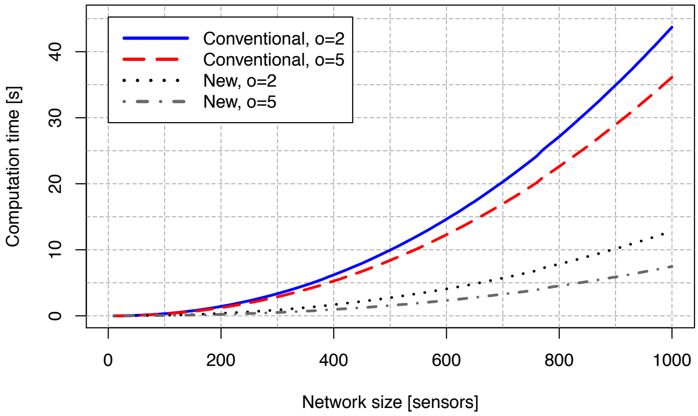


(b) Share of aggrPBOOAB on the network analysis time.

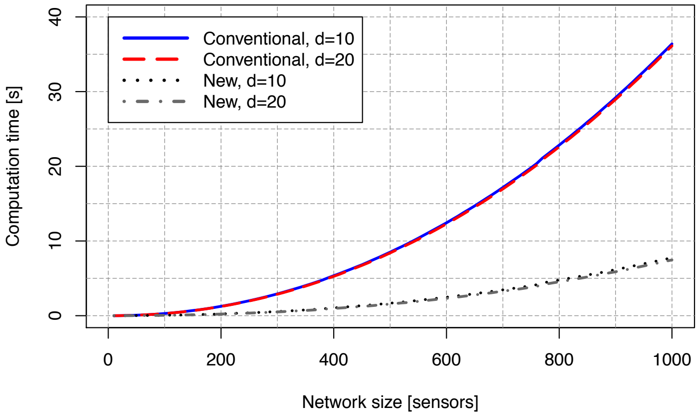
Figure 4.5: Evaluation of the computation time of cross-traffic arrival bounding in random (5,20)-constrained sink trees.

Absolute Improvements Figure 4.6 compares the scaling behavior of the PMOOA with alternative aggrPBOOABs. Both react similar to changes of o (Figure 4.6a) and d (Figure 4.6b). While our new method's run-time still scales super-linearly with the increasing network size, it does so much slower. This generates a noticeable, widening gap between the conventional and the new analysis. This improvement is probably the best one could hope for when improving SensorNC analysis without compromising on the tightness of the results.

Note, that Figure 4.6 shows the computation times for the analysis of a single network size averaged over 40 randomly created instances. When varying the configuration it is not only sensible to experiment with different values for o and d or test multiple deployments complying with the resulting (o, d)-constraint, but also to test heterogeneous network configurations by varying the two defining parameters of flows and sensors in-



(a) Altering the maximum out degree o.



(b) Altering the maximum depth d.

Figure 4.6: Scaling behavior of the improved SensorNC analysis.

dependently. Thus, a design space exploration can easily consist of tens of thousands of distinct analyses to be executed. The overall execution time clearly has to scale in this number. Therefore, the small absolute analysis run-times we report here are crucial.

4.5 Distribution of the SensorNC Analysis Procedure

The probably larger impact of our novel SensorNC arrival bounding algorithm is in its superior structure. From the perspective of a distributed application of the SensorNC, it can be applied in an in-network (sensing) task admission control system as it would be desirable for large-scale WSNs. In particular, using the conventional, recursive arrival bounding method for a distributed SensorNC requires detailed knowledge at each node about large subtrees of the network, including paths of flows as well as their merging locations. All this information is required to backtrack flows, bound cross-traffic arrivals and eventually carry out the analysis that results in the delay bound deciding upon the admission of the new (sensing) task. Moreover, even small changes in the network's defining parameters would lead to the dissemination of lots of data and the recomputations of all bounds, even if the change was only server- or flow-local.

In contrast, using the new iterative aggrPBOOAB allows SensorNC to virtually separate cross-flows from the aggregate they are merged into and separately bound their impact on it. The separation shifts the aggregation of cross-flows to the server where they influence a different flow's performance characteristics. Thus, our analysis establishes locality such that we do not require access to topological information on a cross-flow's path, i.e., every flow can collect all the parameters defining its effect. Moreover, this flow-locality makes SensorNC much more resilient to recalculations as the impact of parameter variations does not propagate to flows not directly depending on this parameter. For resource constrained wireless sensor networks, these are critical non-functional aspects of the analysis.

Neither the conventional, recursive aggrPBOOAB nor the segrPMOOAB allow for distribution of the execution over the sensor network. It would essentially be equal to the centralized execution and thus imposed the need to gather all the required information at each sensor; a characteristic we conceptually evaluate in this section.

The SensorNC aggrPBOOAB's flow-locality can be used for dissemination of the information for compFFA step 1:

- Flows carry their current arrival bound as payload such that it eventually reaches all the servers the flow traverses.
- Servers store each incoming flow's arrival bound and then adapt it according to their own service curve (Corollary 4.4, output bound).

For the second step of compFFA, i.e., bounding the delay of flow f on the path P from its source up to server s (and eventually up to the network's sink), some additional information in the payload of f suffices:

- f's left-over service curve on P, $\beta_P^{\text{l.o.}f}$. Note, that Algorithm 4.1 proceeds in a forward fashion along f's path, i.e., it starts at f's source and moves server-by-server towards the network's sink. Thus, a valid left-over service curve for f's previous path can be derived at any server, especially $\beta_P^{\text{l.o.}f}$ at s.
- f's original arrival curve α^f that is required to compute its delay bound when traversing P (Corollary 4.4).

In compFFA step 1, we reverse the aggrPBOOAB's working direction: It is not backtracking in a recursive fashion (compFFA step 1a) anymore. Applying our procedure, each server is provided with all information about cross-traffic arrivals and flow delays after an initial setup phase. Both can then be used for task admission control.

Note, that in SensorNC the derivation as well as the curves involved are lightweight. The former is depicted in Corollary 4.4, the latter can be explained as follows. Each curve $\alpha = \gamma_{r,b} \in \mathcal{F}_{\mathrm{TB}}$ and $\beta = \beta_{R,T} \in \mathcal{F}_{\mathrm{RL}}$ can be efficiently stored with two values each: the rate r and the bucket size b or the rate R and the latency T, respectively. In fact, our procedure for the distributed SensorNC analysis only adds a total of five values to every flow's payload. A flow's arrival curve and arrival bound always possess the same rate such that three values suffice in addition to the left-over service curve's two values.

Next, we exemplarily evaluate the impact of the above scheme. We consider a self-modeling WSN providing a task admission control scheme based on delay bounds. As soon as a new task is supposed to be added to the network, the WSN checks if it can schedule the task's data flow without compromising any other flow's delay constraint. In order to do so, it is necessary to derive each flow's delay bound under the hypothetical new configuration. For simplicity of the comparison, assume the sink is not unconstrained and a new task is supposed to be added to it. Moreover, we assume a fully occupied (o, d)-constrained sink tree, i.e., maximum outdegree of o and maximum depth of d are both attained by the tree. This setting allows us to compare the conventional, recursive aggrPBOOAB and the new, SensorNC aggrPBOOAB.

In such a sink tree, there are $N := \frac{o^{d+1}-1}{o-1} - 1$ nodes above the sink, all of which hold information necessary to derive delay bounds. Whereas the above scheme distributes this information during normal operation, the previous, recursive scheme requires two preceding phases to acquire it. In the request phase, there is communication to N servers to query their parameter settings (Figure 4.7a) and in the reporting phase N flows answer the query (Figure 4.7b). These $2 \cdot N$ temporary flows should not interfere with the regular traffic, i.e., they need to be scheduled at a lower priority so that they do not force existing flows to violate their deadlines. Thus, termination of the two phases is not guaranteed. In the new SensorNC scheme, neither of the two phases is required, there is no such communication overhead, only a small additional payload.

Second, we compare the *storage demand* at a node. In the conventional aggrPBOOAB, it was required to store all N service curves of a subtree's nodes, N arrival curves of the flows originating in it and the N output arrival curves of flow aggregates at each sensor node. In order to derive the latter values, it is also necessary to have exhaustive knowledge about the network: Where do flows originate? Which paths do flows take? Where do flows aggregate? (see Figure 4.3a) Further, a node needs to store the cross-traffic arrival bound for each flow f crossing it after deriving it from the topological knowledge. In contrast, the new analysis only requires the arrival curves of all N flows, their left-over service curves as well as their arrival bounds at the sensor node. Concerning cross-traffic

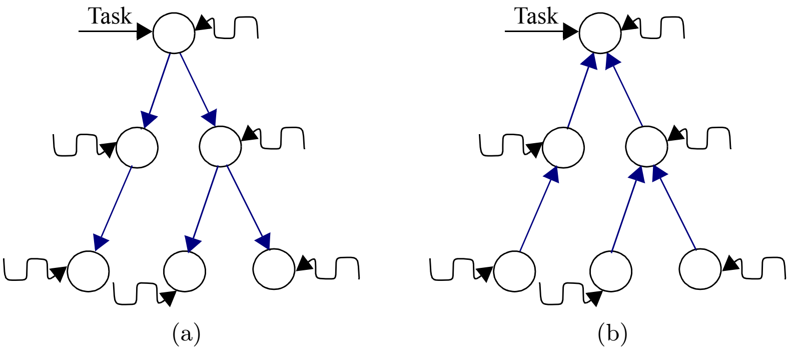


Figure 4.7: Task admission control in a $(2,3)$-constrained sink tree: (a) request phase and (b) reporting phase.

arrivals, a node simply derives the required bound by adding up already stored values. Thus, our new scheme considerably reduces the storage demand to execute SensorNC inside the network.

Regarding the *computational effort*, a sensor node previously had to bound cross-traffic arrivals at all non-leaf nodes above it, i.e., execute $N - o \cdot d$ operations, and compute the output arrival curve at all upstream nodes [71, 65], i.e., execute another N operations. In the new scheme, a node only computes the output arrival curve of flows crossing it, i.e., it just executes N operations. In the next analysis step, deriving aggregate PBOO arrival bounds, the computational effort is reduced from a repetitive execution of the operations for each flow, considering the unique aggregation on its path that defines $\alpha^{x(f)}$, to adding up the flow-local bounds that were derived only once. This reduction in complexity of the analysis leads to faster computation times as discussed in Section 4.4.4.

With the SensorNC aggrPBOOAB we can keep the communication overhead low, decrease storage demands and reduce computational effort. All of this is achieved with a simple scheme that can be deployed with the network.

5 Limitations of Linear Programming Analysis

This section provides a comprehensive analysis and evaluation of optNC. In contrast to algNC, the optNC's current analyses, LPA and ULPA, directly derive an optimization formulation from the entire feed-forward network description. I.e., an optimization analysis has global knowledge about the network. While this guarantees for best attainable results, the analysis effort becomes prohibitive in reasonably sized networks. We are the first to show that this is also true for the less accurate, more efficient of both optimization analyses, the ULPA. In extensive evaluations we also show that the improvement over the algebraic PMOOA is not necessarily large.

The previous section already addressed that the LPA can derive more accurate delay bounds than algNC. In addition to the contribution of [74], the LPA exploits its global view on the network to relate the entanglement of starts of backlogged periods on distinct paths to implement the PSOO principle. Such paths are, e.g., different branches of a tree network. It does not need to assume the worst case (see Section 2.4.1, step 2). For SensorNC, the non-functional aspects of algNC were vital, yet, in general feed-forward network analysis, delay bound tightness might be more important. Therefore, we analyze the linear programming analysis in this section.

5.1 Scalability Issues

The initial proposal of an optimization-based NC analysis [74] was evaluated in [45, 46]. It was shown that computational hardness arises from the decomposition of arrival curves into token buckets and service curves into rate latencies and their subsequent analysis. For each combination, a new optimization has to be executed, resulting in computational infeasibility to analyze tandems of servers when curves become more complex than simple token buckets and rate latencies. The LPA does not decompose curves in order to first operate on pairs of simpler curves before combining partial results. Its NP-hardness results from the extension of a partial order to the set of all compatible total orders – a different source for combinatorial explosion. It does not affect tandem topologies. There, the order derived by the LPA's backtracking step is the only total order. This is the best case for the LPA. In the previous section we saw that the algNC sink-tree analysis scales very well. We will use these topologies to investigate the scalability issues of the LPA. In the worst case for the LPA, we have a so-called fat tree with a root node and $n-1$ leaf nodes directly connected to it, resulting in $(n-1)!$ total orders that each will become one linear program. In a full binary tree, the number of linear programs is lower bounded by $\Omega\left(\left(\frac{n}{2}\right)!\right)$. The latter fact can be derived from a result by Ruskey [66]:

Assume a sink tree with n nodes that are numbered 1 to n, starting from the sink. Let k_i denote the number of nodes upstream of node i, including i itself. In a binary sink-tree network $\mathcal{N}_2$ the amount of total orders is

$$\mathrm{Tot}(\mathcal{N}_2\ ) \quad = \quad \frac{n!}{k_1 \cdot k_2 \cdots k_n} = \frac{n!}{\prod_{i=1}^{n} k_i} \ .$$

For a fully occupied binary tree of depth d, $\mathcal{N}_2'$, the amount of total orders can be bounded as follows:

$$\begin{aligned}
\mathrm{Tot}\left(\mathcal{N}_2'\right) \quad &= \quad \frac{n!}{k_1 \cdot k_2 \cdots k_n} \\
\text{(on level } d\text{, } k_i = 1\text{)} \quad &= \quad \frac{n \cdot (n-1) \cdots \left\lceil \frac{n}{2} \right\rceil \cdots 1}{n \cdot k_2 \cdots k_{\left\lceil \frac{n}{2} \right\rceil + 1} \cdot 1 \cdots 1} \\
\text{(left fraction } \geq 1\text{)} \quad &= \quad \frac{n \cdot (n-1) \cdots \left\lceil \frac{n}{2} \right\rceil + 1}{n \cdot k_2 \cdots k_{\left\lceil \frac{n}{2} \right\rceil}} \cdot \frac{\left\lceil \frac{n}{2} \right\rceil \cdots 1}{1 \cdots 1} \\
&\geq \quad \frac{\left\lceil \frac{n}{2} \right\rceil \cdot \left(\left\lceil \frac{n}{2} \right\rceil - 1 \right) \cdots 1}{1 \cdots 1} \\
&\geq \quad \left(\frac{n}{2} \right)! \qquad \in \Omega\left(\left(\frac{n}{2} \right)! \right) \\
\text{(Stirling's formula)} \quad &\sim \quad \sqrt{\pi n} \left(\frac{n}{2e} \right)^{\frac{n}{2}}
\end{aligned}$$

For fully occupied (o, d)-constrained sink trees, i.e., sink trees with maximum outdegree of o and maximum depth of d, this formula can be generalized to

$$\frac{n!}{\prod_{i=2}^{d} \left(\left\lfloor \frac{o^i - 1}{o - 1} \right\rfloor \right)^{o^{d-i}}} \ .$$

In general, calculating the number of total orders being compatible with a given partial order is itself not a simple problem. One solution is the Varol-Rotem algorithm [85]; we implemented this algorithm to provide some numbers for the case of full d-ary trees in Table 5.1. It is obvious that the computational effort to solve such large numbers of linear programs becomes quickly prohibitive even for moderate network sizes. In feed-forward networks other than trees, this is, however, not the final set of linear programs. The LPA implements the PSOO principle by adapting the total orders. It requires to iterate over the set of total orders to apply modifications.

5.2 A Tradeoff between Accuracy and Efficiency

The authors of the LPA were aware that the extension of partial orders is prone to a combinatorial explosion that prohibits the analysis of feed-forward networks. On the other hand, they observed that the single total order derived for a tandem network could be solved in a small amount of time. Therefore, they provide an analysis that is based on their derivation of the optimization formulation, yet, results in a single linear program for any feed-forward network.

5.2.1 The Unique Linear Programming Analysis

In contrast to the algNC, the LPA approaches the tight delay bound from the region of invalid result candidates. Only a subset of the linear programs derived by the LPA produces a valid (the tight) bound. As the bound is unknown until the end, the LPA cannot be terminated before taking the maximum over all its result candidates or even be restricted to a subset of linear programs from the start. I.e., the LPA constitutes an all-or-nothing analysis approach. It is, however, possible to reduce the analysis to the set of constraints that are shared among all linear programs of the LPA. In practice, this means skipping the extension of the partial order to the set of compatible total orders. Obviously, this circumvents the combinatorial explosion in the number of linear programs. Instead, it results in a single, the *unique* LPA (ULPA). The ULPA neither relates the backlogged periods of (partially) parallel paths, e.g., between different branches of a tree, nor does it implement the PSOO principle. However, worst-case modeling of NC ensures valid upper bounds and in [12] the ULPA was shown to stay very close to the LPA in an example network. This observation raises hope for computational feasibility and performance bound accuracy in larger networks.

Improving the ULPA The ULPA constitutes the return to accurate, yet, untight bounds for general feed-forward networks. We know that the ULPA has less constraints than any of the LPA's linear programs. Improving the ULPA's result can only be achieved by adding more constraints to it. The addition of constraints found in a total order can, however, result in a linear program that produces an invalid bound. Identifying constraints that improve the derived bound while guaranteeing to retain its validity is

		Depth d			
		0	1	2	3
Outdegree o	1	1	1	1	1
	2	1	2	80	21,964,800
	3	1	6	7,484,400	$3.54 \cdot 10^{37}$
	4	1	24	$3.89 \cdot 10^{15}$	$1.12 \cdot 10^{110}$
	5	1	120	$3.41 \cdot 10^{28}$	$5.88 \cdot 10^{246}$

Table 5.1: Number of linear programs to solve for fully occupied (o, d)-constrained sink trees of moderate size. For instance, for a depth $d = 3$ and outdegree of $o = 4$, the tree has $n = \frac{o^{d+1}-1}{o-1} = 85$ nodes but already yields $1.12 \cdot 10^{110}$ programs.

an open research topic of optNC.

5.3 Evaluation of Accuracy and Effort

The literature presents an evaluation of the two networks posing problems for algNC, but are solved with the LPA [12, 42, 10, 16]. On the one hand, there is the tandem with non-nested interference of cross-flows as already used in the first paper to motivate optimization in NC [74]. On the other hand, the authors derive bounds for the square network in order to evaluate the impact of the PSOO principle (cf. Figure 3.2a), a key strength of the LPA. In both networks, the compositional analysis procedure of Section 3.1 enforces cross-flow segregation that leads to mutual interference assumptions. In this section, we want to address the question of accuracy of the existing NC analyses. For the ULPA, we strive for insight on the impact of not implementing the PSOO principle and how the analysis compares to the compositional algNC. As the LPA lacks tool support for general feed-forward networks due to its limited applicability, we need to restrict our evaluation to the aforementioned networks. In the tandem network, we use the ULPA implementation provided by the literature as the LPA and the ULPA coincide. For the square network, we use the provided set of linear programs [12]. For algNC, we provide the equations created by different analyses in order to understand where the composition penalty causes overly pessimistic assumptions. An exception is the Total Flow Analysis (TFA) that neither implements the PBOO principle nor the PMOO principle. Its delay bounds are strictly inferior to the SFA bounds and will not be analyzed in detail. We restrict the presentation to results derived with the DiscoDNC [7] in order to illustrate the negative impact of this lack of analysis principle implementation.

5.3.1 The Non-Nested Tandem

Analyzing tandem networks with the optNC is made possible with a tool provided with the (U)LPA [12]. It transforms a text file describing the tandem (servers and service curves; flows, their paths and arrival curves) into the (U)LPA – given as a LpSolve lp file. In [12], the tandem with overlapping interference, a non-nested interference pattern, is analyzed. It consists of a sequence of n servers crossed by the foi ($P(\text{foi}) = \langle s_1, s_2, \ldots, s_{n-1}, s_n \rangle$) and overlapping interference of cross-flows such that there are three flows at every server (see Figure 5.1). This tandem constitutes a standard example in NC; it has already been used for introducing the PMOOA [75].

Two investigations are carried out: one that evaluates the impact of increasing tandem length, and another one varying the utilization for the tandem of length 20.

Arrival curves and service curves are provided by [12]: Both evaluations assign rate-latency service $\beta_{R,T} = \beta_{10,\frac{1}{10}}$. The evaluation with varying utilization $u \in \{0.1, \ldots, 0.9\}$

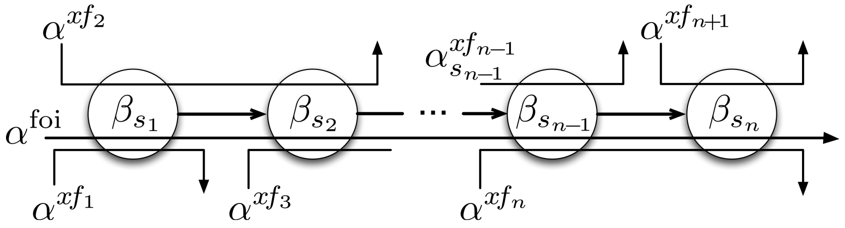


Figure 5.1: Tandem network with overlapping interference.

assigns token-bucket arrival curves of $\alpha = \gamma_{r,b} = \gamma_{\frac{10u}{3},1}$ where $\frac{10u}{3}$ is rounded to two decimal digits. The derivation of results for increasing tandem length is carried out at an utilization of 0.2 [12], i.e., all arrivals are shaped to $\alpha = \gamma_{0.67,1}$.

Separate Flow Analysis First, we derive the foi's end-to-end left-over service curve (compFFA step 2):

$$\begin{aligned}
\beta^{\text{l.o.foi}}_{P(\text{foi})} &= \beta^{\text{l.o.foi}}_{s_1} \otimes \beta^{\text{l.o.foi}}_{s_2} \otimes \ldots \otimes \beta^{\text{l.o.foi}}_{s_n} \\
&= \Big(\beta_{s_1} \ominus \Big(\alpha^{xf_1}_{s_1} + \alpha^{xf_2}_{s_1}\Big)\Big) \otimes \Big(\beta_{s_2} \ominus \Big(\alpha^{xf_2}_{s_2} + \alpha^{xf_3}_{s_2}\Big)\Big) \otimes \ldots \otimes \Big(\beta_{s_n} \ominus \Big(\alpha^{xf_n}_{s_n} + \alpha^{xf_{n+1}}_{s_n}\Big)\Big) && (5.1) \\
&= \Big(\beta_{s_1} \ominus \Big(\alpha^{xf_1} + \alpha^{xf_2}\Big)\Big) \otimes \Big(\beta_{s_2} \ominus \Big(\alpha^{xf_2}_{s_2} + \alpha^{xf_3}\Big)\Big) \otimes \ldots \otimes \Big(\beta_{s_n} \ominus \Big(\alpha^{xf_n}_{s_n} + \alpha^{xf_{n+1}}\Big)\Big) && (5.2)
\end{aligned}$$

In this derivation, we can see that the PMOO principle is not implemented. We need to pay for xf_m arrivals, $m \in \{2, \ldots n\}$ at both servers a cross-flow shares with the foi. For this reason, the SFA results are expected to be worse than the PMOOA results in the non-nested tandem.

Before we provide delay bounds for benchmarking against the LPA, we first need to compute the arrivals of xf_m at their respective second server of interference with the foi (compFFA step 1, Equation 5.2: αs with server indices). For each server s_i, $i \in \{2, \ldots, n\}$, cross-traffic arrivals are derived as follows:

$$\begin{aligned}
&\beta_{s_i} \ominus \Big(\alpha^{xf_i}_{s_i} + \alpha^{xf_{i+i}}_{s_i}\Big) \\
&\quad = \Big(\beta_{s_i} \ominus \Big(\Big(\alpha^{xf_i} \oslash \beta^{\text{l.o.}xf_i}_{s_{i-1}}\Big) + \alpha^{xf_{i+1}}\Big)\Big) \\
&\quad = \Big(\beta_{s_i} \ominus \Big(\Big(\alpha^{xf_i} \oslash \Big(\beta_{s_{i-1}} \ominus \alpha^{xf_{i-2}}_{s_{i-1}}\Big)\Big) + \alpha^{xf_{i+1}}\Big)\Big) \\
&\quad = \Big(\beta_{s_i} \ominus \Big(\Big(\alpha^{xf_i} \oslash \Big(\beta_{s_{i-1}} \ominus \Big(\alpha^{xf_{i-2}} \oslash \Big(\ldots \Big(\beta_{s_1} \ominus \alpha^{xf_1}\Big) \ldots\Big)\Big)\Big)\Big) + \alpha^{xf_{i+1}}\Big)\Big) && (5.3)
\end{aligned}$$

At every server, the cross-traffic arrival bounding demands to recursively backtrack cross-flows of cross-flows. The recursion terminates when the sources of flows are reached. See

Equation 5.3 where αs do not have server indices as we know $\alpha_{s_1}^{xf_1} = \alpha^{xf_1}$, $\alpha_{s_n}^{xf_{n+1}} = \alpha^{xf_{n+1}}$ and $\alpha_{s_{m-1}}^{xf_m} = \alpha^{xf_m}$ for $m \in \{2, \ldots n\}$ from the network description.

Pay Multiplexing Only Once Analysis We start with the left-over service curve derivation of the PMOOA:

$$\beta_{P(\text{foi})}^{\text{l.o.foi}} = \beta_{R_{P(\text{foi})}^{\text{l.o.foi}}, T_{P(\text{foi})}^{\text{l.o.foi}}} \tag{5.4}$$

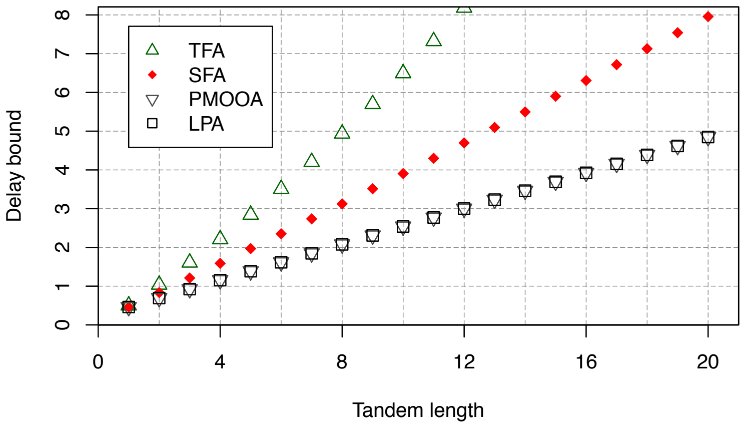
$$R_{P(\text{foi})}^{\text{l.o.foi}} = \left(R_{s_1} - r^{xf_1} - r^{xf_2}\right) \wedge \left(R_{s_2} - r^{xf_2} - r^{xf_3}\right) \wedge \ldots \wedge \left(R_{s_n} - r^{xf_n} - r^{xf_{n+1}}\right) \tag{5.5}$$

$$\begin{aligned} T_{P(\text{foi})}^{\text{l.o.foi}} = \; & T_{s_1} + T_{s_2} + \ldots + T_n \\ & + \frac{b_{s_1}^{xf_1} + b_{s_1}^{xf_2} + b_{s_2}^{xf_3} + \ldots + b_{s_{n-1}}^{xf_n} + b_{s_n}^{xf_{n+1}}}{R_{P(\text{foi})}^{\text{l.o.foi}}} \\ & + \frac{\left(r_{s_1}^{xf_1} + r_{s_1}^{xf_2}\right) \cdot T_{s_1} + \left(r_{s_2}^{xf_2} + r_{s_2}^{xf_3}\right) \cdot T_{s_2} + \ldots + \left(r_{s_n}^{xf_n} + r_{s_n}^{xf_{n+1}}\right) \cdot T_{s_n}}{R_{P(\text{foi})}^{\text{l.o.foi}}} \end{aligned} \tag{5.6}$$

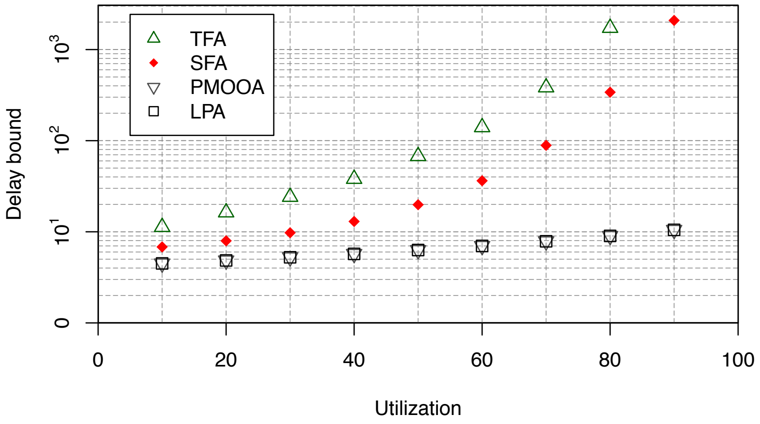
In contrast to the SFA, xf burstiness terms only appear once in the $\beta_{P(\text{foi})}^{\text{l.o.foi}}$-derivation. I.e., the PMOOA does not demand a cross-flow arrival bounding with compFFA step 1 like the SFA does. From the foi's point of view, multiplexing with cross-traffic is payed for only once.

LPA and Comparison The results for the LPA are depicted alongside the PMOOA, SFA and TFA delay bounds in Figures 5.2a and 5.2b. Most notably, PMOOA and LPA results are equal. Moreover, the TFA exhibits a super-linear growth of delay bounds – when increasing the tandem length and when increasing the utilization of the 20-servers tandem. The SFA, PMOOA and the LPA, in contrast, show a linear growth w.r.t. the tandem length (Figure 5.2a), the PBOO principle implemented by all of them is responsible for this improvement. In the utilization-focused analysis (Figure 5.2b), we see that PMOOA and LPA delay bounds grow super-linearly as well. Yet, the PMOO principle implemented by both leads to a much smaller growth than in TFA and SFA (Figure 5.2b). These analyses pay multiplexing twice for every cross-flow xf_m, $m \in \{2, \ldots, n\}$. For both parts of this non-nested tandem evaluation, the PMOOA performs equal to the LPA.

After manually deriving the SFA (Equations 5.1 to 5.3) and the PMOOA (Equations 5.4 to 5.6), we can contribute an in-depth explanation of this observation:



(a) Delay bounds depending on the tandem length.



(b) Delay bounds for a tandem of 20 servers, depending on the utilization.

Figure 5.2: Delay bound evaluation results for the non-nested tandem with maximally overlapping interference of cross-flows.

The PMOO principle was designed to counteract bursts being considered in the foi analysis multiple times. Therefore, PMOOA performs better on the non-nested tandem than SFA. The utilization increase distributes uniformly over the servers such that PMOOA and LPA delay bounds increase in equal steps, starting from the same value; the PMOOA's weakness is known to only surface in tandems with unbalanced (left-over) service rates [74].

5.3.2 The Square Network

For the square network in [12] (see Figure 3.2a), the LPA's linear programs as well as the ULPA's one are provided as LpSolve lp files by Bouillard et. al. For this evaluation, service curves remain $\beta_{s_i} = \beta_{R,T} = \beta_{10,\frac{1}{10}}$, $i \in \{1, 2, 3, 4\}$, and arrival curves are again adapted to the utilization $u \in \{0.1, \ldots, 0.9\}$. As there are only two flows per server, this setting translates to $\alpha^{f_j} = \gamma_{r,b} = \gamma_{\frac{10u}{2},1}$, $j \in \{1, 2, 3, 4\}$.

Separate Flow Analysis

We start with the SFA left-over service curve derivation for all utilizations:

$$\beta^{\text{l.o.}f_1}_{\langle s_3,s_4\rangle} \quad = \quad \beta^{\text{l.o.}f_1}_{s_3} \otimes \beta^{\text{l.o.}f_1}_{s_4} \quad = \quad \left(\beta_{s_3} \ominus \alpha^{f_2}_{s_3}\right) \otimes \left(\beta_{s_4} \ominus \alpha^{f_4}_{s_4}\right) \tag{5.7}$$

Cross-traffic arrivals are bounded as follows (aggrPBOOAB as well as segrPBOOAB):

$$\begin{aligned} \alpha^{f_2}_{s_3} &= \alpha^{f_2} \oslash \beta^{\text{l.o.}f_2}_{s_1} \oslash \beta_{s_2} \;=\; \alpha^{f_2} \oslash \left(\beta_{s_1} \ominus \alpha^{f_3}\right) && (5.8) \\ \alpha^{f_4}_{s_4} &= \alpha^{f_4} \oslash \beta^{\text{l.o.}f_4}_{s_2} \;=\; \alpha^{f_4} \oslash \left(\beta_{s_2} \ominus \alpha^{f_3}_{s_2}\right) \\ &= \alpha^{f_4} \oslash \left(\beta_{s_2} \ominus \left(\alpha^{f_3} \oslash \beta^{\text{l.o.}f_3}_{s_1}\right)\right) \;=\; \alpha^{f_4} \oslash \left(\beta_{s_2} \ominus \left(\alpha^{f_3} \oslash \left(\beta_{s_1} \ominus \alpha^{f_2}\right)\right)\right) && (5.9) \end{aligned}$$

The cross-traffic arrival bounding shows the mutual interference assumption already depicted in Figure 3.2b: We have to compute $\beta_{s_1} \ominus \alpha^{f_3}$ and $\beta_{s_1} \ominus \alpha^{f_2}$, both will be in the SFA left-over service curve derivation of Equation 5.7 – the PSOO principle is not implemented. This derivation illustrates the different approaches applied by algNC to model the foi's worst-case setting in a feed-forward network. On the foi's path $P(\text{foi})$ (i.e., the foi tandem analysis of step 2), the left-over service curve derivation assumes lowest priority for the foi (cf. Theorem 2.3). In the arrival bounding, each flow's worst case is modeled individually. Therefore, at server s_1, flows f_2 and f_3 are considered to be mutual interference: $\beta^{\text{l.o.}f_2}_{s_1} = \left(\beta_{s_1} \ominus \alpha^{f_3}\right)$ and $\beta^{\text{l.o.}f_3}_{s_1} = \left(\beta_{s_1} \ominus \alpha^{f_2}\right)$. For rejoining interference with the foi (see Diamond network in [12]) this means the foi is assumed to interfere with its own cross-flows during their arrival bounding, i.e., the PSOO principle can even impact the analysis on the foi's path. The lowest priority modeling applied in compFFA step 2 is not carried over to cross-traffic arrival bounding of step 1, i.e., algNC analysis does not assign lowest priority to the foi within the entire feed-forward network but depending on the current compFFA step (see Section 3.1).

Pay Multiplexing Only Once Analysis

The PMOO left-over service curve is derived as follows:

$$\beta^{\text{l.o.}f_1}_{\langle s_3,s_4\rangle} = \beta_{R^{\text{l.o.}f_1}_{\langle s_3,s_4\rangle},T^{\text{l.o.}f_1}_{\langle s_3,s_4\rangle}} \tag{5.10}$$

$$R^{\text{l.o.}f_1}_{\langle s_3,s_4\rangle} = \left(R_{s_3} - r^{f_2}\right) \wedge \left(R_{s_4} - r^{f_4}\right) \tag{5.11}$$

$$T^{\text{l.o.}f_1}_{\langle s_3,s_4\rangle} = T_{s_3} + T_{s_4} + \frac{b^{f_2}_{s_3} + b^{f_4}_{s_4} + r^{f_2}_{s_3} \cdot T_{s_3} + r^{f_4}_{s_4} \cdot T_{s_4}}{R^{\text{l.o.}f_1}_{\langle s_3,s_4\rangle}} \tag{5.12}$$

The mutual interference modeling persists. It can be seen in the cross-flow burst terms $b^{f_2}_{s_3}$ and $b^{f_4}_{s_4}$ in Equation 5.12 that equal the one of the according arrival bounds $\alpha^{f_2}_{s_3}$ and $\alpha^{f_4}_{s_4}$ used in the SFA. Cross-traffic is bounded with Equations 5.8 and 5.9, again, as it belongs to compFFA step 1. I.e., the PMOOA left-over service curve derivation of Equation 5.10 will violate the PSOO principle due to Equation 5.12.

LPA, ULPA and Comparison

Based on the derivation of the algNC analysis equations, i.e., the internal model of the network these analyses operate on, we can already predict the relative performance of NC analyses in the square network:

The TFA analysis performs far worse than the other analyses. This can be explained as follows: Not implementing the PBOO principle results in the foi's burst terms appearing multiple times in the derivation – at every server an independent worst-case delay bounding is executed (instances of compFFA step 1). This translates to a behavior that can neither be attained by a realistic system nor is it assumed by any other analysis. The TFA proceeds as if the foi's burst does not level off but overtakes itself in a subsequent queue. With increasing network utilization, the foi's worst-case burstiness increases and thus the negative assumption becomes more impactful. The result of this escalation is partially depicted in Figure 5.3, the TFA computes a delay bound of ~ 13.58 at a utilization of 90%.

The LPA approach enumerates all potential entanglements of flows by extending the partial order (defined by consecutive hops of flows) to the set of all compatible total orders. For the square network, this extension already results in 11 linear programs. The ULPA, in contrast, is solely based on the partial order, the extension step is omitted due to its potential combinatorial explosion. This results in a single linear program with a smaller set of constraints as the potential entanglements of f_2 and f_3 at s_1 are not exhaustively modeled anymore. Instead, they are assumed to constitute the respective flow's worst case. Based on this insight, it is not surprising that the ULPA actually does not beat the PMOOA in the square network (see Figure 5.3). In fact, even the SFA yields the same results. The reason for SFA's equal accuracy is the lack of multi-hop

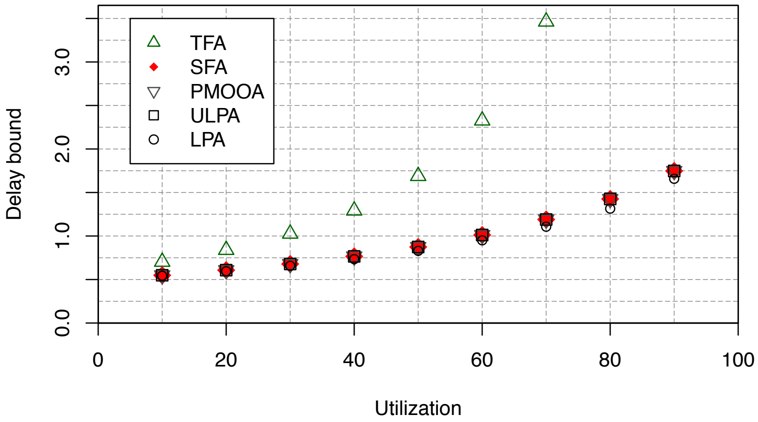


Figure 5.3: Delay bounds in the square network.

interference. The effect captured with the PMOO principle does not manifest in this network. Our evaluation shows that the ULPA may model the worst-case interference between f_2 and f_3 in the same way as algNC.

Finally, Figure 5.3 also illustrates the impact of implementing the PSOO in the LPA analysis: The improvement it achieves over the other analyses varies between 1.01% at a utilization of 10% and 7.79% at a utilization of 80%.

These evaluation results are promising for algNC analyses, however, in Section 3.2 we derived the composition penalty and in Section 3.1 we showed that it affects algNC. The square network is not sufficiently complex to evoke it – even the SFA performs equal to the ULPA in the square network. In the last part of this section, we will provide the first comprehensive evaluation of the ULPA in feed-forward networks. It finally verifies [12]'s conclusion: The ULPA considerably outperforms the SFA w.r.t. flow delay bounds.

5.3.3 Feed-forward Networks

For our numerical investigation, we decided to scale to larger networks and depart from the previous ER topology generation. Instead, we created Internet-like topologies according to the general linear preference (GLP) model [22][7]. Traffic was created with a fixed server-to-flow ratio of 1:4 to generate load in all networks. Table 5.2 shows the devices, servers and flows for all of the GLP networks we evaluate in this thesis. Service curves resemble transmissions via 10Gbps full duplex links. Flows are routed on the shortest server graph path between two randomly chosen network devices. Their arrival curves are uniformly shaped to token buckets with rate 5Mbps and bucket size 5Mb.

---

[7] We applied the default GLP parameter setting ($m_0 = 20$, $m = 1$, $p = 0.4695$, $\beta_{\text{GLP}} = 0.6447$) and used the aSHIIP tool [84] to generate these device graphs.

Devices	Servers	Flows
20	38	152
40	118	472
60	164	656
80	282	1128
100	364	1456
120	398	1592
140	512	2048
160	572	2288
180	646	2584
200	740	2960
220	744	2976
240	882	3528
260	976	3904
280	994	3976
300	1124	4496
400	1478	5912
500	1876	7504
1000	3626	14504

Table 5.2: GLP network sizes (amount of devices) and their respective number of servers and flows.

We employ the DiscoDNC for the compFFA (SFA with aggrPBOOAB). Moreover, we also extended the DiscoDNC to derive the ULPA for arbitrary feed-forward networks – the linear program can be formatted to be either solved with the open-source solver LpSolve or with IBM CPLEX. We used the DiscoDNC 2.2.3, Java 8, LpSolve 5.5.2.0 and CPLEX 12.6.2. All computations were executed on identical servers, each equipped with an Intel Xeon E5420 CPU with four physical cores and 12GB of RAM.

Accuracy Figure 5.4 depicts the reduction of delay bounds the ULPA achieves compared to the algNC analyses with aggrPBOOAB. Due to the composition penalty's "decision on incomplete knowledge" aspect, we ran both, SFA and PMOOA. The statistics comprise of all flows in GLP networks of sizes 20 to 180, i.e., 12376 analyzed flows in total.

Compared to the SFA, the ULPA could reduce delay bounds by an average of 15.18% (improvement factor 1.203), however, more than 30% of SFA flow delay bounds could be reduced by more than 20% with the ULPA and the maximum improvement we observed is as high as 72.65%, i.e., the ULPA bound is smaller than one third of the SFA's delay bound. Clearly, the conclusion of [12], that the ULPA considerably outperforms the SFA in feed-forward networks, is correct.

Compared to the PMOOA, the ULPA does not perform equally well. It only reduced delay bounds by 1.23% on average. Figure 5.4 illustrates that most delay bounds are within close distance of the PMOOA, however, there are some outliers: 32 delay bounds are reduced by the ULPA by more than 3%, three by more than 5% and two by more than 7%. The maximum can be found at 7.63%. This seems to suggest to always execute a PMOOA, i.e., the decision to take on incomplete knowledge seems rather simple. Yet, for 517 out of the 12376 flows the SFA computed a better delay bound than the PMOOA,

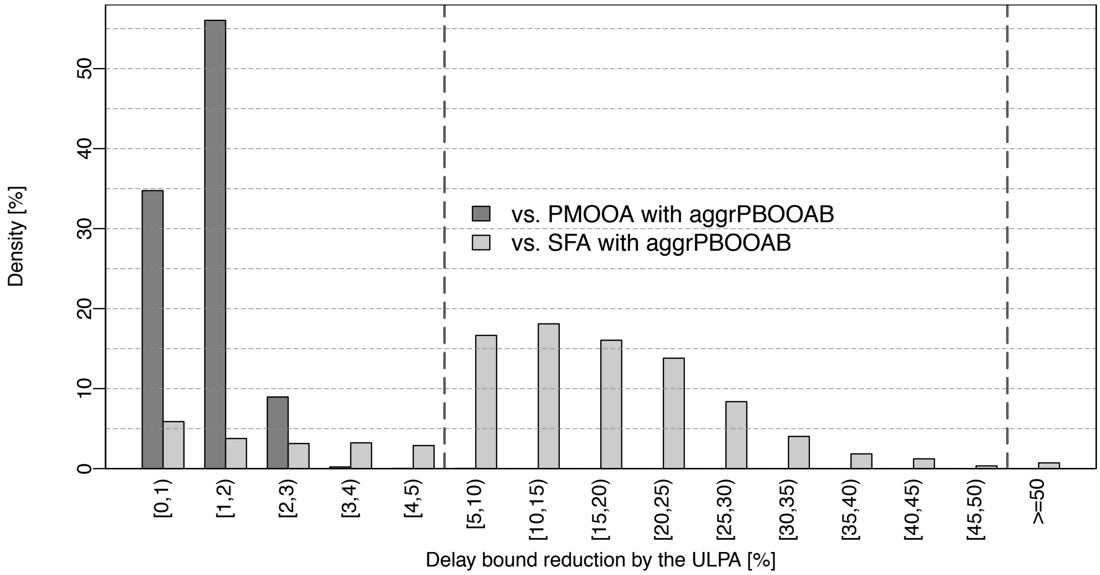


Figure 5.4: Delay bound reduction distribution: ULPA's improvement over flow delay bounds of algNC with aggrPBOOAB in GLP networks of sizes 20 to 180, i.e., comparison of 12376 flows analyses.

and for another 453 flows the results were equal[8].

Computational Effort Last, let us extend our reflection of NC analyses to computational effort. It is the very reason for the step away from tight LPA delay bounds, back to an accurate analysis. Figure 5.5 depicts how, in our sample, the time to analyze an entire GLP network scales with their size.

These results defeat the hope that the ULPA is an efficient optimization-based analysis for feed-forward networks. Employing LpSolve, that was found to perform well on tandems, already yielded problems in the smallest network of this numerical evaluation. There, LpSolve fails to solve one of the linear programs, reporting it to be unbounded after an extensive period of computations. In the GLP network with 40 devices, LpSolve fails with a larger number of linear programs, reporting "unbounded" or "failed". The entire analysis of this small network devices took more than 129 hours. IBM CPLEX performs considerably better, however, at 180 devices it suffers from a network analysis time of ~ 13 days. Thus, the ULPA becomes computationally infeasible to apply, even though it is by far more efficient than the LPA it was derived from.

The network analysis time for algNC also grows with the network size, yet, where the ULPA requires ~ 13 days, SFA finishes in less than 30 minutes and PMOOA takes ~ 22.5

---

[8] Not accounting for potential double rounding errors that may increase this number.

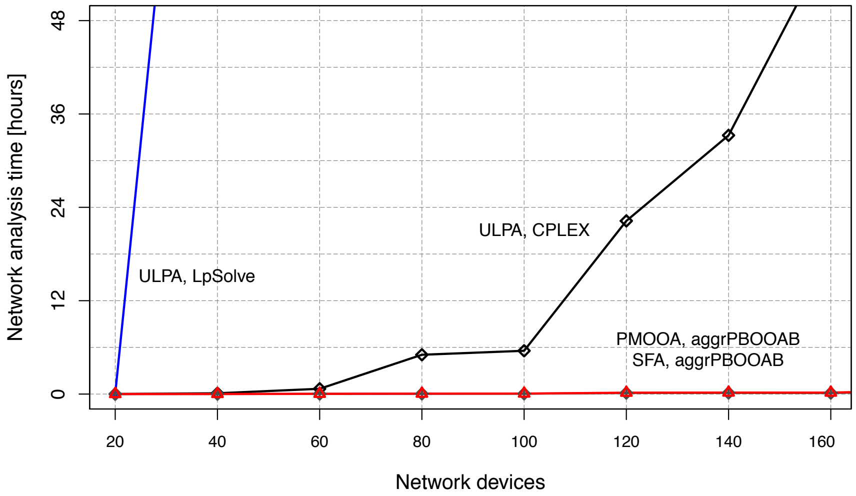


Figure 5.5: Benchmarking NC feed-forward analyses w.r.t. the network analysis time: OptNC's ULPA computation time strongly depends on the employed optimization software, yet, for networks of reasonable size, it cannot compete with algNC's SFA or PMOOA.

minutes.

The observations of this section reveal that NC analyses can be distinguished regarding their fundamental tradeoff between efficiency and accuracy:

1. AlgNC is computationally efficient, however, accuracy of attained delay bounds is not consistently competitive with current optNC analyses. We could ultimately provide evidence that this conclusion is especially correct for the SFA. However, the PMOOA does perform quite well on average. This might be due to the worst-case assumptions of the ULPA. They seem to constrain its search space to a region similar to the algNC. Nonetheless, in some cases the PMOOA performs noticeably worse than optNC's ULPA. Particularly in the context of schedulability decisions, these outliers can be decisive and should be reduced as much as possible.
2. OptNC theoretically allows for the derivation of tight bounds. Yet, in practice neither tight nor accurate bounds are computationally feasible to derive with the current analyses LPA and ULPA. Moreover, the ULPA's accuracy cannot be improved easily and its computational performance strongly depends on the employed optimization software.

6 Optimization Principles in Algebraic Network Calculus

We know from the previous section that compositional algNC analysis can be competitive, especially if the PMOOA is applied in compFFA step 2, the foi's tandem analysis. Now we extend the current algNC analyses with optimization principles. Our new analyses consider a larger search space for the algNC solution and thus become competitive with optNC in more general feed-forward networks as well.

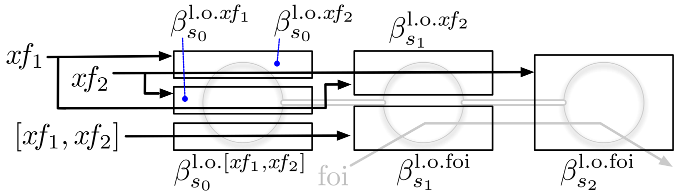
6.1 Exhaustive Aggregate Cross-traffic Arrival Bounding

In our first contribution that incorporates optimization principles into algNC, we derive an algorithm to exhaustively compute aggregate cross-traffic arrival bounds. It searches through all possibilities of applying either SFA or PMOOA on the compFFA's tandems. This approach maximally counteracts the composition penalty's segregation problems with the two current algNC tandem analyses. In evaluations, we show significantly improved delay bounds over the literature's cross-flow segregation but also prohibitive effort in moderately sized feed-forward networks.

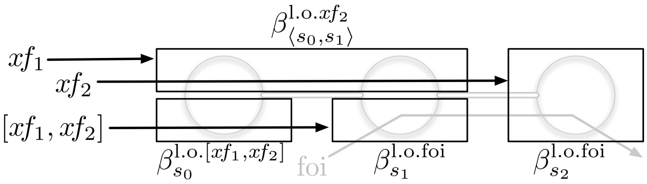
The search space algNC traverses to find the best delay bound is not yet maximized, i.e., it can still be further exhausted within the compFFA framework. We already saw that the composition penalty's "decisions on incomplete knowledge" aspect results in an extended search: In compFFA step 2, SFA and PMOOA are both used to analyze the foi. We now derive a scheme to extend the search in compFFA step 1, the cross-traffic arrival bounding, to an equally exhaustive scheme.

In section 3.3.2, we provide Algorithm 3.1 for aggrPBOOAB. It proceeds server-by-server in order to derive left-over service curves for cross-traffic aggregates. E.g., in Figure 6.1a the arrival bounding of xf_2 at the right-most server requires both its left-over service curves $\beta_{s_0}^{\mathrm{l.o.}xf_2}$ and $\beta_{s_1}^{\mathrm{l.o.}xf_2}$. While this implements the PBOO principle, multiplexing with xf_2's cross-traffic on these two consecutive servers might have to be payed for multiple times. The PMOO principle can, however, only be implemented on tandems. For single servers, the interference pattern will always be nested and left-over service curve computations of the SFA and the PMOOA coincide. Therefore, Algorithm 3.1 must be extended to backtrack through the network in a tandem-by-tandem fashion. The step from Figure 6.1a to Figure 6.1b illustrates this conversion with xf_2's arrival bound derivation at server s_2:

$$\begin{aligned}
\alpha_{s_2}^{xf_2} &= \alpha_{s_0}^{xf_2} \oslash \beta_{s_0}^{\mathrm{l.o.}xf_2} \oslash \beta_{s_1}^{\mathrm{l.o.}xf_2} \\
&= \alpha_{s_0}^{xf_2} \oslash \left(\beta_{s_0}^{\mathrm{l.o.}xf_2} \otimes \beta_{s_1}^{\mathrm{l.o.}xf_2}\right) \\
&= \alpha_{s_0}^{xf_2} \oslash \beta_{\langle s_0, s_1\rangle}^{\mathrm{aggrPBOOl.o.}xf_2}
\end{aligned}$$


$\beta_{s_0}^{\text{l.o.}xf_1}$
$\beta_{s_0}^{\text{l.o.}xf_2}$
$\beta_{s_1}^{\text{l.o.}xf_2}$
xf_1
xf_2
$[xf_1, xf_2]$
$\beta_{s_0}^{\text{l.o.}[xf_1,xf_2]}$
foi
$\beta_{s_1}^{\text{l.o.foi}}$
$\beta_{s_2}^{\text{l.o.foi}}$

(a) SFA with aggrPBOOAB (cf. Figure 3.3c).


$\beta_{\langle s_0,s_1\rangle}^{\text{l.o.}xf_2}$
xf_1
xf_2
$[xf_1, xf_2]$
$\beta_{s_0}^{\text{l.o.}[xf_1,xf_2]}$
foi
$\beta_{s_1}^{\text{l.o.foi}}$
$\beta_{s_2}^{\text{l.o.foi}}$

(b) SFA with aggrPMOOAB.

Figure 6.1: Extending aggrPBOOAB to aggrAB.

where $\beta^{\text{aggrPBOOl.o.}}$ denotes the aggrPBOOAB left-over service curve as employed by the SFA. In order to implement the PMOO principle, we replace $\beta^{\text{aggrPBOOl.o.}}$ with its PMOO counterpart $\beta^{\text{aggrPMOOl.o.}}$, the left over service curve of the PMOOA (see Section 2.3). This results in the following derivations for SFA delay bounding (Figure 6.1a):

- $\beta_{s_1}^{\text{l.o.foi}} \otimes \beta_{s_2}^{\text{l.o.foi}} = \beta_{\langle s_1,s_2\rangle}^{\text{l.o.foi}}$ is the SFA foi analysis of compFFA step 2.
- $\beta_{s_0}^{\text{l.o.}[xf_1,xf_2]}$ exploits aggregate cross-traffic bounding at s_1, circumventing mutual interference assumptions between xf_1 and xf_2 at s_0 in compFFA step 1.
- $\beta_{\langle s_0,s_1\rangle}^{\text{l.o.}xf_2}$ is the tandem analysis that bounds $\alpha_{s_2}^{xf_2}$. It is an independent instance of compFFA step 1.

Overall, we see that fundamental problems of compFFA persist: In case $\beta_{\langle s_0,s_1\rangle}^{\text{l.o.}xf_2} = \beta_{\langle s_0,s_1\rangle}^{\text{aggrPBOOl.o.}xf_2}$, the PSOO principle is violated at s_0. Moreover, the left-over derivations for cross-traffic arrival bounding operate with their independent worst-case assumptions that are contradicting in the global view: $\beta_{s_0}^{\text{l.o.}[xf_1,xf_2]}$ aggregated both cross-flows whereas $\beta_{\langle s_0,s_1\rangle}^{\text{l.o.}xf_2}$ always considers xf_1 cross-traffic of xf_2. Exhausting the cross-traffic arrival bounding thus doubles the alternatives on every tandem to be analyzed in compFFA step 1. We abbreviate the exhaustive aggregate cross-traffic arrival bounding with aggrAB and will apply it to both analyses, SFA and PMOOA. But first we demonstrate that it can yield delay bound improvements in the example given in Figure 6.1.

Example 6.1. *(Exhaustive Aggregate Cross-traffic Arrival Bounding)* Consider the two alternative cross-traffic arrival bounding procedures for $\alpha_{s_2}^{xf_2}$ depicted in Figures 6.1a and 6.1b (network of Figure 3.3). Note, that there are no further alternatives as the remaining left-over service curves $\beta_{s_0}^{\text{l.o.}[xf_1,xf_2]}$, $\beta_{s_1}^{\text{l.o.foi}}$ and $\beta_{s_2}^{\text{l.o.foi}}$ are all single-server ones. Therefore, the difference in the foi's delay bound is solely defined by the $\beta_{s_2}^{\text{l.o.foi}}$'s latency $T_{s_2}^{\text{l.o.}}$ that, in turn, is defined by the only cross-flow at s_2: xf_2. Its arrivals differ in both alternatives as follows:

$$\begin{aligned}
\alpha_{s_2}^{\text{aggrPBOO}xf_2} &= \alpha^{xf_2} \oslash \beta_{s_0}^{\text{l.o.}xf_2} \oslash \beta_{s_1}^{\text{l.o.}xf_2} \\
&= \alpha^{xf_2} \oslash \left(\beta_{s_0} \ominus \alpha^{xf_1}\right) \oslash \left(\beta_{s_1} \ominus \alpha_{s_1}^{xf_1}\right) \\
&= \alpha^{xf_2} \oslash \left(\beta_{s_0} \ominus \alpha^{xf_1}\right) \oslash \left(\beta_{s_1} \ominus \left(\alpha^{xf_1} \oslash \beta_{s_0}^{\text{l.o.}xf_1}\right)\right) \\
&= \alpha^{xf_2} \oslash \left(\beta_{s_0} \ominus \alpha^{xf_1}\right) \oslash \left(\beta_{s_1} \ominus \left(\alpha^{xf_1} \oslash \left(\beta_{s_0} \ominus \alpha^{xf_2}\right)\right)\right),
\end{aligned}$$

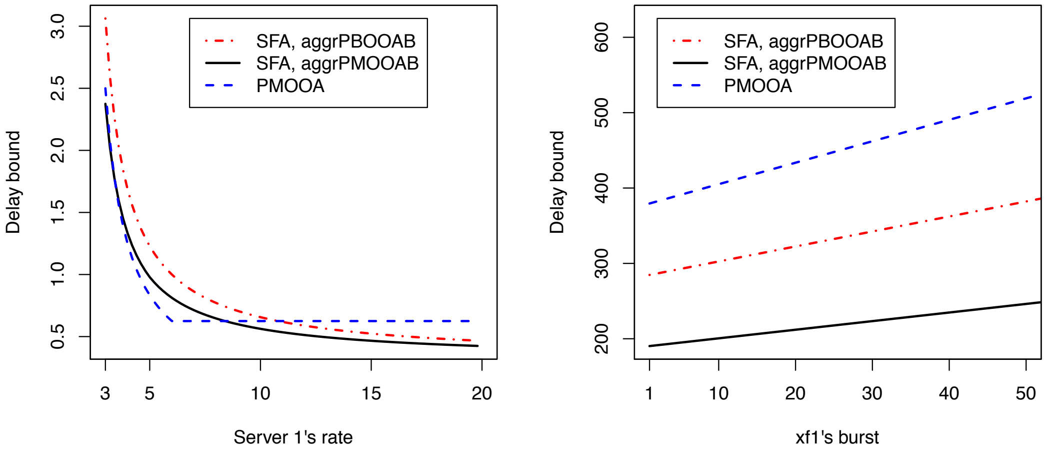
$$\begin{aligned}
\alpha_{s_2}^{\text{aggrPMOO}xf_2} &= \alpha^{xf_2} \oslash \beta_{\langle s_0,s_1\rangle}^{\text{l.o.}xf_2} \\
&= \alpha^{xf_2} \oslash \left(\left(\beta_{s_0} \otimes \beta_{s_1}\right) \ominus \alpha^{xf_1}\right),
\end{aligned}$$

i.e., they can both outperform each other, depending on the left-over rate at s_1 due to $\otimes$'s commutativity. Therefore, the composition penalty's "decisions on incomplete knowledge" also applies to cross-traffic arrival bounding. Note, that s_1 is the first server on the foi's path, yet, the last server crossed by xf_2. As a result, the observation made in [74] that SFA can outperform PMOOA if the latter servers are faster must be generalized in feed-forward networks. Here, even fast servers at the front impose problems for the PMOOA due to cross-traffic arrival bounding.

Figure 6.2 illustrates the impact of this observation. As depicted, different alternatives of foi analysis (compFFA step 1, including PMOOA) and cross-traffic arrival bounding (compFFA step 2) can yield the best delay bound.

- In example 1 (Figure 6.2a), we chose the following parameter setting: $\alpha^{xf_1} = \alpha^{xf_2} = \alpha^{\text{foi}} = \gamma_{1,0}$, $\beta_{s_0} = \beta_{5,1}$, $\beta_{s_1} = \beta_{R_{s_1},0}$, and $\beta_{s_2} = \beta_{5,0}$, i.e., R_{s_1} is varied.
- In example 2 (Figure 6.2b), we set $\beta_{s_0} = \beta_{25,5}$, $\beta_{s_1} = \beta_{25,0}$, $\beta_{s_2} = \beta_{3,5}$, $\alpha^{\text{foi}} = \gamma_{0.5,5}$, $\alpha^{xf_1} = \gamma_{2.5,b^{xf_1}}$, $\alpha^{xf_2} = \gamma_{2.5,5}$ and increase b^{xf_1}.

Summing up, we showed that neither of the two algebraic alternatives to derive the left-over service curve strictly outperforms the other one in general feed-forward networks. Moreover, their results depend on actual parameters of strict service curves and cross-traffic arrival bounds. Whereas the service curves are known, the arrival bounds need to be computed first. When decomposing the server graph into left-over service derivations, the superior $\beta^{\text{l.o.}}$ is thus still unknown. For this reason, we propose to apply all alternative derivations from the start and on each level of the cross-traffic arrival bounding recursion.



(a) Delay bounds under increasing R_{s_1}.

(b) Delay bounds under increasing b^{xf_1}.

Figure 6.2: Delay bound comparison for different parameter settings (see Example 6.1).

Finally, we check every combination for its result in order to find the best delay bound. We call this method the exhaustive aggregate arrival bounding aggrAB. It is implemented in Algorithm 6.1 where lowercase letters denote variables and blackboard bold letters denote sets of variables. The algorithm proceeds as follows:

The objective is similar to Algorithm 3.1 and so is the entry point – a set of flows $\mathbb{F}$ to bound and a server s to bound their arrivals at. Like the non-exhaustive algorithm, we first iterate over the inlinks of s (lines 2 to 6) and then add the arrivals of flows originating at s (lines 8 to 10). A difference to the previous recursive approach resides in line 7: AggrAB returns two bounds per inlink, one for aggrPBOOAB and one for aggrPMOOAB. The exact amount of returned bounds depends on the tandems that are analyzed deeper in the recursion, i.e., in the following instances of compFFA step 1.

Inlinks define the locations where recursive cross-traffic bounding may be required for (subsets of) the currently analyzed flows or their respective cross-traffic must be bounded. In both cases, new instances of compFFA step 1 are required to derive aggrPBOOAB and aggrPMOOAB results – these may violate the PSOO principle. For every inlink, the arrival bound is computed by first searching for the longest tandem flows traverse aggregately ($\mathcal{T}_{\text{shared}}$, line 13), then recursively deriving the cross-traffic arrivals for this tandem (lines 14 to 17), applying the SFA and PMOOA left-over service curve derivation (line 18, see Section 2.3) and computing the flow aggregate's output from $\mathcal{T}_{\text{shared}}$ (lines 19 to 23). Moreover, the TFA assistance to cap cross-traffic burst at the output of $\mathcal{T}_{\text{shared}}$ is implemented in aggrAB as well (lines 21 and 24).

Algorithm 6.1: AggrAB Computation.

```
Input  : Set of flows $\mathbb{F}$ to bound at server $s$
Output: Set of arrival bounds $\mathbb{A}_s^{\mathbb{F}}$

ArrivalBoundsAtServer(Server $s$, Flow set $\mathbb{F}$)
    foreach Link $l \in s$.getInLinks() do
        $\mathbb{F}_l = \mathbb{F} \cap F(l)$;
        $\mathbb{A}_l^{\mathbb{F}}$ = ArrivalBoundsOnLink($l$, $\mathbb{F}_l$);
        $\mathbb{A}_{\text{inlinks}}^{\mathbb{F}}$.put($l$, $\mathbb{A}_l^{\mathbb{F}}$);
    end
    $\mathbb{A}_s^{\mathbb{F}}$ = getABCombinations($\mathbb{A}_{\text{inlinks}}^{\mathbb{F}}$);
    foreach $\alpha \in \mathbb{A}_s^{\mathbb{F}}$ do
        $\alpha \mathrel{+}= F_{\text{src}}(s) \cap \mathbb{F}$;
    end
return $\mathbb{A}_s^{\mathbb{F}}$;

ArrivalBoundsOnLink(Link $l$, Flow set $\mathbb{F}$)
    $\mathcal{T}_{\text{shared}}$ = getSharedPathTo($\mathbb{F}$, $l$.getSource());
    foreach Server $s \in \mathcal{T}_{\text{shared}}$ do
        $\mathbb{F}_s^{x(\mathbb{F})} = F(s) \setminus \mathbb{F}$;
        $\mathbb{A}_{\mathcal{T}_{\text{shared}}}^{x(\mathbb{F})}$.put($s$, ArrivalBoundsAtServer($\mathbb{F}_s^{x(\mathbb{F})}$));
    end
    $\mathbb{B}_{\mathcal{T}_{\text{shared}}}^{\text{l.o.}\mathbb{F}}$ = LeftOverBetas($\mathcal{T}_{\text{shared}}$, $\mathbb{A}_{\mathcal{T}_{\text{shared}}}^{x(\mathbb{F})}$);
    $source$ = $\mathcal{T}_{\text{shared}}$.getSource();
    $\mathbb{A}_{source}^{\mathbb{F}}$ = ArrivalBoundsAtServer($source$, $\mathbb{F}$);
    $B^{\text{TFA}}$ = TFAbacklogBound($l$.getSource())
    foreach $\beta \in \mathbb{B}_{\mathcal{T}_{\text{shared}}}^{\text{l.o.}\mathbb{F}}$ do
        $\mathbb{A}_l^{\mathbb{F}}$.addAll($\mathbb{A}_{source}^{\mathbb{F}} \oslash \beta$);
        $\mathbb{A}_l^{\mathbb{F}}$.capAll($B^{\text{TFA}}$);
    end
return $\mathbb{A}_l^{\mathbb{F}}$;
```

6.1.1 Accuracy Evaluation

We continue to incrementally evaluate the impact of improvements with a numerical evaluation. In addition, we provide a case study that depicts our overall progress.

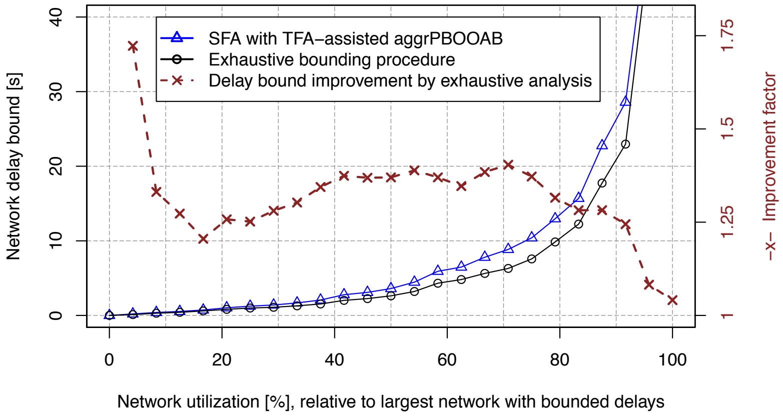
Note, that in an exhaustive accuracy evaluation of network delay bounds, we do not simply compute $\min\left(D^{\text{SFA}}, D^{\text{PMOOA}}\right)$ where D^{SFA} denotes the network delay bound of an SFA with aggrAB and D^{PMOOA} the respective PMOOA one – a straight-forward extension of previous analysis proceedings. Both of these network delay bounds are valid so their minimum is a valid network delay bound as well. However, this countermeasure to the composition penalty only finds the best solution if we have a single cross-traffic arrival bounding alternative. Therefore, we do not consider this limited search an implementation of optimization principles. It lacks exhaustiveness. Instead, we compare both alternative delay bounds for every individual flow in order to derive D^{exh}. Let us illustrate the difference in a small example with three flows, f_1, f_2 and f_3. Assume that their delay bounds derived by SFA are $D^{f_1}_{\text{SFA}} = 5$, $D^{f_2}_{\text{SFA}} = 4$ and $D^{f_3}_{\text{SFA}} = 4$. For PMOOA let them be $D^{f_1}_{\text{PMOOA}} = 4$, $D^{f_1}_{\text{PMOOA}} = 4$ and $D^{f_1}_{\text{PMOOA}} = 6$. Then we get $\min\left(D^{\text{SFA}}, D^{\text{PMOOA}}\right) = \min\left(\min\left(5,4,4\right), \min\left(4,4,6\right)\right) = 5$ whereas $D^{\text{exh}} = \min\left(\min\left(5,4\right), \min\left(4,4\right), \min\left(4,6\right)\right) = 4$, i.e., $D^{\text{exh}} < \min\left(D^{\text{SFA}}, D^{\text{PMOOA}}\right)$. We observed this benefit of our more exhaustive procedure in the following evaluations, especially in networks approaching their utilization limits.

Numerical Experiments In this accuracy evaluation, we use the same networks as in Section 3.3.6, i.e., we benchmark the scaling behavior w.r.t. an increasing utilization in two randomly created Erdős-Rényi networks – a flat one and an hierarchical one. Figure 6.3 shows the results. We compare the latest algNC analysis without optimization principles, the SFA with TFA-assisted aggrPBOOAB, with the results of the exhaustive bounding procedure (SFA and PMOOA in compFFA step 2, each with TFA-assisted aggrAB in step 1).

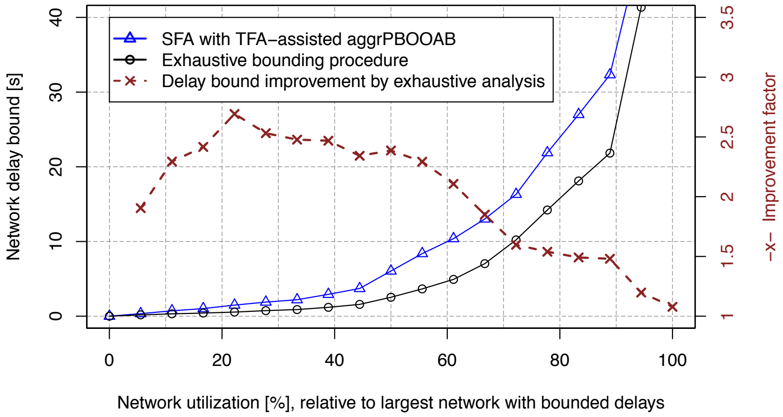
In both networks, the maximum delay bounds for the respective network utilization are improved. In contrast to the previous evaluation, the reduction of bounds decreases when the utilization becomes large and the delay bounds approach their asymptote. At 100%, the improvement factor of both networks is at ∼1, i.e., the network delay bounds are (almost) identical. For the range of network utilizations where algNC derives small delay bounds, the exhaustive bounding procedure even improves the results. Its improvement factor is almost always above 1.25, in the hierarchical ER network where results are usually more pronounced, it even reaches values above 2.

Moreover, we could observed that $D^{\text{exh}} < \min\left(D^{\text{SFA}}, D^{\text{PMOOA}}\right)$. In the flat ER network it only occurred in the highest utilization setting, but in the hierarchical network this was the case for every network delay bound we computed for utilizations above $\frac{7}{9}$.

AFDX Case Study Next, we evaluate the algNC improvements contributed in this thesis in an industrial avionics case study. The network we exemplarily analyze is dimensioned similarly to the AFDX (Avionics Full-Duplex Switched Ethernet) backbone network in the Airbus A380. It has a dense core of 16 switches that connect a total of 125 end-systems in the network periphery. Each server has a service curve resembling a



(a) Flat ER network with 32 devices.



(b) Hierarchical ER network with 32 devices.

Figure 6.3: Network delay bounds for flat and hierarchical Erdős-Rényi networks with $n \in \{32, 64\}$ devices.

100Mbps Ethernet link. We created a representative AFDX topology according to the algorithm presented in [20]. This topology generation scheme has some random factors in it, i.e., from an industrial point of view, the network we analyze here corresponds to a single alternative in a pre-deployment design space exploration.

According to the current AFDX specification, flows are routed within so-called virtual links (VLs). Each VL connects a single source end-system to multiple sink end-systems (in the device graph) with fixed resource reservation on the path between these systems. In the view of current NC analyses, VLs correspond to multicast flows that reserve large resource shares. An examination of the problems due to VLs' coarse granularity can be found in [55]. Moreover, NC does not provide a specialized analysis for multicast communication that implements the PBOO or even the PMOO principle. Implementing them to some degree requires another network transformation before the actual analysis:

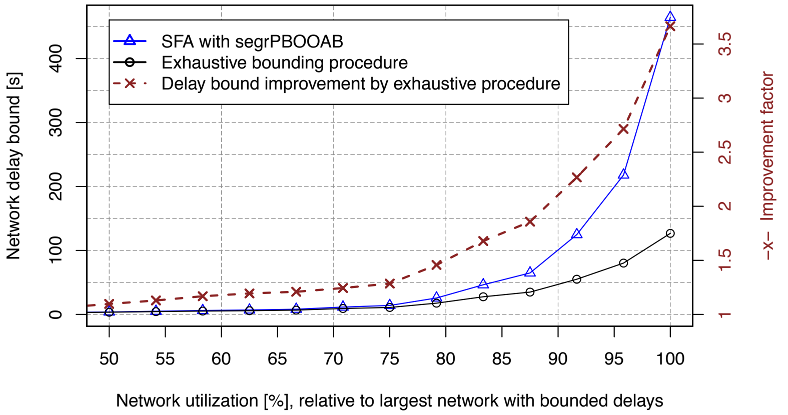


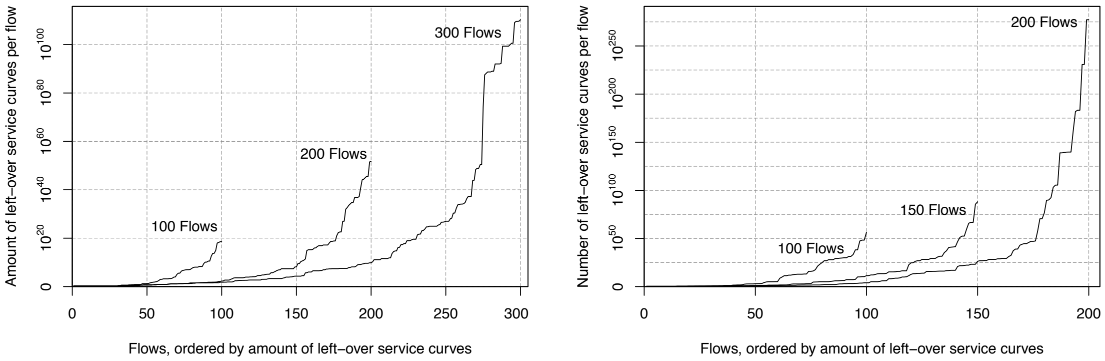
Figure 6.4: Network delay bounds in the AFDX topology, minimum of SFA and PMOOA with segrPBOOAB or aggrAB.

Each multicast flow is converted into a set of independent unicast flows; one for every source-sink pair of connected devices in order to model the network. This step, however, constitutes a flow segregation similar to the ones mentioned earlier. For this reasons, we restrict our evaluation to the immutable part of AFDX: The already deployed networks' common topology design.

The evaluation proceeds like the ER one (cf. Section 3.3.6): We continuously add unit size flows, routed in the server graph between two randomly chosen network devices from the device graph. Figure 6.4 shows the impact of improved arrival bounding as derived in this thesis.

The AFDX topology can benefit much from our method. Starting at an improvement factor of 1.1, the gap between old and new network delay bound grows fast until it reaches the bottleneck capacity and factor 3.36. Each of the 250 links to and from an end-system gets congested easily, similar to the few bottlenecks connecting different levels in the hierarchical Erdős-Rényi network. Moreover, AFDX's flat core is very small and thus prone to the dynamic bottleneck emergence already observed in the flat Erdős-Rényi network. Therefore, the results are particularly pronounced in the AFDX topology. The network delay bound as well as the improvement both grow super-linearly with the bottleneck utilization but aggrAB reduces the network delay bound compared to segrPBOOAB by 70%.

Further, we observed $D^{\text{exh}} < \min\left(D^{\text{SFA}}, D^{\text{PMOOA}}\right)$ in the AFDX topology for all network utilizations of 87.5% or larger, i.e., different flows define the respective network delay bound.



(a) Flat ER network with 32 devices.

(b) Hierarchical ER network with 32 devices.

Figure 6.5: Amount of each flow's of left-over service curves derived with aggrAB. Growing a network in size by adding flows reveals the combinatorial explosion this approach already suffers from in the small sample networks.

6.1.2 Effort Considerations

We already pointed out that this optimization principle is prone to a combinatorial explosion. Next, we evaluate the extent by counting the left-over service curves derived for the foi. The amount of these curves is a good indicator for the analysis effort. It doubles with every new instance of compFFA step 2 and the involved operations to derive a curve can be implemented efficiently ($\mathcal{O}(1)$, see [17]).

Figure 6.5 depicts the total number of curves for each analyzed flow in one hierarchical as well as one flat ER network of our set of networks for evaluation. We show three different network sizes in terms of number of flows to indicate scalability of the exhaustive approach. Neither of the networks exhibits the efficiency of algNC anymore, applying the exhaustive search for best cross-traffic arrival bounds renders this approach infeasible, even in small networks. I.e., the composition penalty continues to demand to decide upon a small subset of alternative analyses for cross-traffic arrival boundings to retain computational feasibility.

6.2 The Tandem Matching Analysis

The section contributes the tandem matching analysis (TMA). It is more flexible w.r.t. the conceptual compFFA procedure for feed-forward networks (Section 3.1); demarcation between flow of interest analysis and cross-traffic arrival bounding as well as the restriction to either applying SFA or PMOOA are relaxed. Our evaluations show that TMA delay bounds become very competitive with the optimization-based ULPA, deviating only by an average of 1.16% in our comprehensive tests with Internet-like topologies.

Moreover, we show that the TMA generalizes Section 6.1's aggrAB that was developed to adhere to compFFA's restrictions. Last, we demonstrate that this generalization retains the problem of combinatorial explosion and thus prohibitive effort before Section 7 contributes countermeasures.

We propose a feed-forward network analysis that achieves more accurate delay bounds by compiling the system description into a model best suited for compFFA. I.e., we consider algNC analysis' strengths and weaknesses presented in Section 3.1 in order to minimize the composition penalty. We call this optimization principle Tandem Matching (TM). The according foi analysis implementing TM in compFFA step 2 is the Tandem Matching Analysis (TMA). The TMA does not create a new analysis in the sense of an alternative solution for the left-over service curve derivation or the direct computation of bounds. We rather provide an entirely new analysis procedure consisting of (a preceding) step that incorporates optimization principles into the algNC analysis.

6.2.1 Matching Tandems

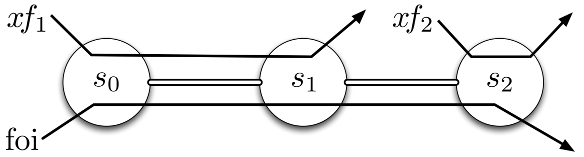
Currently, there is no generic tandem decomposition scheme for algebraic tandem analyses. The procedure mainly depends on the choice of analysis (recall Section 3.1 as well as Figures 3.3c and 3.3b). The SFA first matches shortest possible sub-tandems in a preceding step, whereas the PMOO starts with the foi's entire path. The decision for either must be taken without knowing which alternative is superior. Moreover, neither of both actually guarantees for the best alternative as this example illustrates:

Example 6.2. *(Tandem Matching Analysis)* Consider the network in Figure 6.6a that depicts the same sample network as in Section 3.4 but with a different foi to analyze. The foi crosses all three servers of the tandem. Figure 6.6b shows the SFA's server-by-server decomposition for analysis and Figure 6.6c shows the PMOOA analyzing the entire tandem at once. As before, we assume that the arrival curves are from $\mathcal{F}_{\mathrm{TB}}$ and the service curves are from $\mathcal{F}_{\mathrm{RL}}$. We get the following left-over service curve derivations:

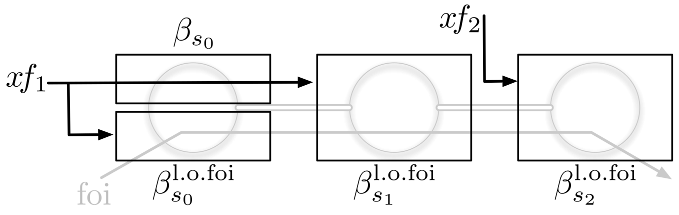
$$
\begin{aligned}
\beta^{\mathrm{SFAl.o.foi}} &= \beta_{s_0}^{\mathrm{l.o.foi}} \otimes \beta_{s_1}^{\mathrm{l.o.foi}} \otimes \beta_{s_2}^{\mathrm{l.o.foi}} \\
&= \left(\beta_{s_0} \ominus \alpha^{xf_1}\right) \otimes \left(\beta_{s_1} \ominus \left(\alpha^{xf_1} \oslash \beta_{s_1}\right)\right) \otimes \left(\beta_{s_2} \ominus \alpha^{xf_2}\right)
\end{aligned}
$$

$$
\begin{aligned}
\beta^{\mathrm{PMOOAl.o.foi}} &= \beta_{R^{\mathrm{l.o.}},T^{\mathrm{l.o.}}} \text{ with} \\
R^{\mathrm{l.o.}} &= \left(R_{s_0} - r^{xf_1}\right) \wedge \left(R_{s_1} - r^{xf_1}\right) \wedge \left(R_{s_2} - r^{xf_2}\right) \\
T^{\mathrm{l.o.}} &= T_{s_0} + T_{s_1} + T_{s_2} + \frac{b^{xf_1} + b^{xf_2} + r^{xf_1} \cdot (T_{s_0} + T_{s_1}) + r^{xf_2} \cdot T_{s_2}}{R^{\mathrm{l.o.}}}
\end{aligned}
$$

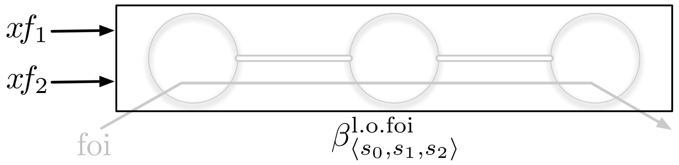
In the derivation of $\beta^{\mathrm{SFAl.o.foi}}$, multiplexing with cross-flow xf_1 is payed for twice and in



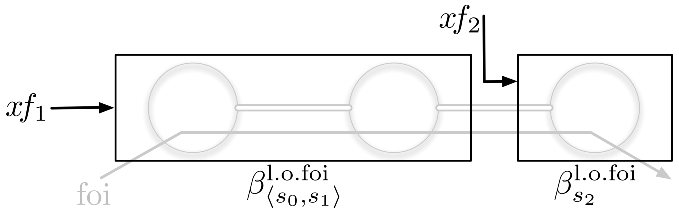
(a) Sample Network.


β_{s_0}
xf_2
xf_1
foi
$\beta^{\text{l.o.foi}}_{s_0}$
$\beta^{\text{l.o.foi}}_{s_1}$
$\beta^{\text{l.o.foi}}_{s_2}$

(b) Figure 6.6a's SFA-internal model.



(c) Figure 6.6a's PMOOA-internal model.


xf_2
xf_1
foi
$\beta^{\text{l.o.foi}}_{\langle s_0,s_1\rangle}$
$\beta^{\text{l.o.foi}}_{s_2}$

(d) An additional TM alternative.

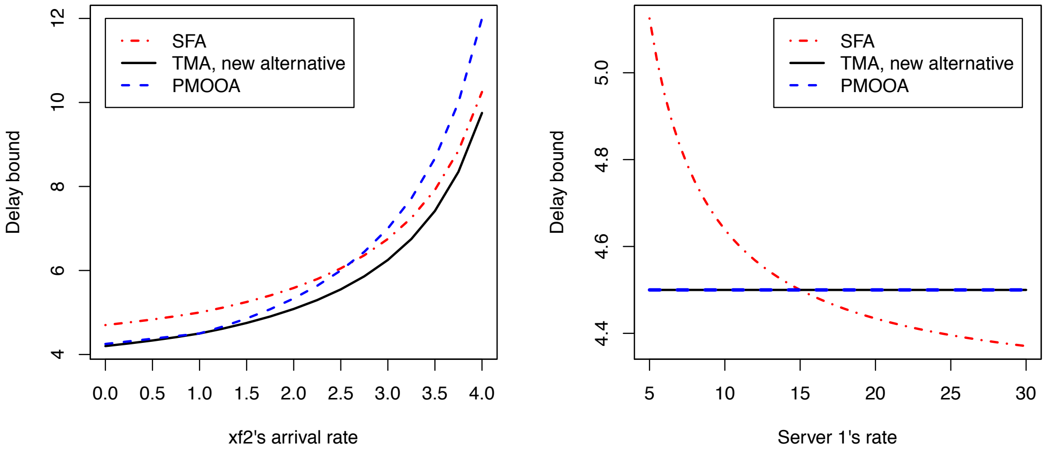
Figure 6.6: Sample network and tandem decompositions of different analysis approaches.

$\beta^{\text{PMOOAl.o.foi}}$ we cannot benefit from fast residual service rates on the foi's path $P(\text{foi})$ as both cross-traffic bursts (and their respective increase) are served with $R^{\text{l.o.}}$, the minimum over all three servers on the tandem.

We see, both current alternatives to match tandems onto $P(\text{foi})$ in order to define the left-over service curve have their respective composition penalty. However, these alternatives only demarcate the search space of the tandem matching onto $P(\text{foi})$. Figure 6.6d shows an additional alternative that we analyze in an exhaustive TMA[9]. It matches a tandem to the first two servers in order to benefit from implementing the PMOO principle

---

[9] We omit the fourth alternative to match on s_0 and $\langle s_1, s_2\rangle$ for brevity. It is valid but does not contribute to this example: $\beta^{\text{PMOOAl.o.foi}}_{\langle s_1,s_2\rangle} \leq \left(\beta_{s_1} \ominus \alpha^{xf_1}_{s_1}\right) \otimes \left(\beta_{s_2} \ominus \alpha^{xf_2}_{s_2}\right) = \beta^{\text{SFAl.o.foi}}_{\langle s_1,s_2\rangle}$. The inequality holds due to PMOOA's left-over rate problem mentioned above.



(a) Delay bounds under increasing r^{xf_2}.

(b) Delay bounds under increasing R_{s_1}.

Figure 6.7: Delay bound comparison for different parameter settings (see Example 6.2).

and separates the last server to potentially benefit from its residual service rate:

$$\begin{aligned}
\beta^{\text{l.o.foi}} &= \beta^{\text{PMOOAl.o.foi}}_{\langle s_0,s_1\rangle} \otimes \beta^{\text{PMOOAl.o.foi}}_{s_2} \\
&= \beta_{R^{\text{l.o.}},T^{\text{l.o.}}} \text{ with} \\
R^{\text{l.o.}} &= \left(R_{s_0} - r^{xf_1}\right) \wedge \left(R_{s_1} - r^{xf_1}\right) \wedge \left(R_{s_2} - r^{xf_2}\right) \\
T^{\text{l.o.}} &= T_{s_0} + T_{s_1} + T_{s_2} + \frac{b^{xf_1} + r^{xf_1} \cdot (T_{s_0} + T_{s_1})}{(R_{s_0} - r^{xf_1}) \wedge (R_{s_1} - r^{xf_1})} + \frac{b^{xf_2} + r^{xf_2} \cdot T_{s_2}}{R_{s_2} - r^{xf_2}}
\end{aligned}$$

Figure 6.7 illustrates the impact of this TM alternative on the delay bound.

- In example 1 (Figure 6.7a), we chose the following parameter setting: $\alpha^{xf_1} = \alpha^{\text{foi}} = \gamma_{1,1}$, $\alpha^{xf_2} = \gamma_{r^{xf_2},1}$ $\beta_{s_0} = \beta_{s_1} = \beta_{s_1} = \beta_{5,1}$, i.e., the cross-traffic arrival rate r^{xf_2} at the last server is increased to reduce the residual rate $\beta^{\text{PMOOAl.o.foi}}$ assumes for all servers on the tandem. After if falls below the residual rate on the first servers, PMOOA derives larger delay bounds than our new TM alternative. The SFA additionally violates the PMOO principle and thus cannot outperform it. However, it can outperform PMOOA as soon as the impact of the PMOO principle is outweighed by the impact of the server at the end.

- In example 2 (Figure 6.7b), we set $\alpha^{xf_1} = \alpha^{xf_2} = \alpha^{\text{foi}} = \gamma_{1,1}$, $\beta_{s_0} = \beta_{s_1} = \beta_{5,1}$, $\beta_{s_1} = \beta_{R_{s_1},1}$, and increase R_{s_1}. Among our three alternatives, only the SFA separates the middle server from the others and can thus benefit from a fast rate there.

Based on the basic insight provided by Example 6.2, we propose to search for the best delay bound considering all combinations a tandem can be decomposed into, i.e., among all tandem matchings onto it. As every link can be the start and the end of two adjacent tandems of a TM alternative, this will result in a total of $2^{|L|}$ alternatives where $|L|$ denotes the amount of links. Again, the delay bound derived with one of the the alternatives depends on the actual parameters. Therefore, the best tandem matching alternative cannot be derived from the given network description in a static fashion. Cross-traffic arrival bounds, in turn, depend on an accurate, recursively derivation themselves. In order to maximize the search space, TMA should be accompanied by a Tandem Matching Arrival Bounding (TMAB) that also exhaustively matches to the tandems that are derived during aggrAB. Exhaustive tandem matching, i.e., TMA for compFFA step 2 and TMAB for compFFA step 1, generates the entire set of TM alternatives for a feed-forward network. On the individual tandems, we apply the PMOOA left-over service curve – the server-by-server matching of SFA is now explicitly executed by the exhaustive tandem matching. I.e., TMA generalizes all existing algebraic tandem analyses by making the step from the network description to the analysis' internal model explicit. In our search space for algNC, we then try to find the alternative with the minimal composition penalty – similar to optimization searching for the best result in the constrained region of results. This approach is inspired by the LPA but in contrast to it, the TMA with TMAB does not generate invalid delay bounds. It does not constitute an all-or-nothing approach. Our new method is applicable to any feed-forward network. If not explicitly mentioned, we use the term TMA to refer to the entire TM-based analysis that consists of TMA for compFFA step 2 with TFA-assisted TMAB for compFFA step 1.

6.2.2 Related Work

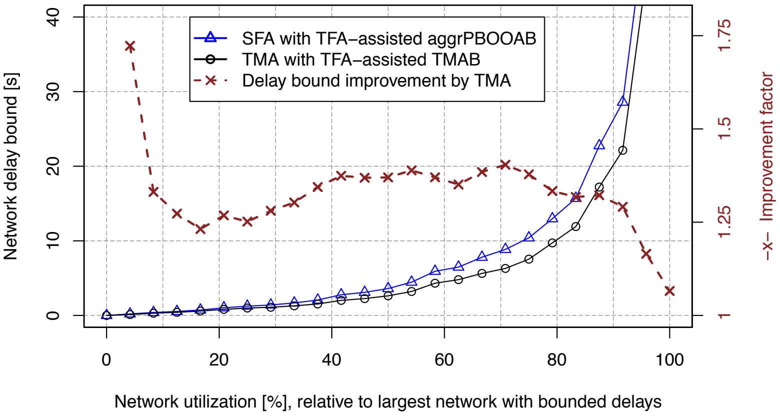
In this section, we rely on the PMOOA left-over service curve for arbitrary multiplexing of flows, i.e., the worst-case remaining service capturing any possible scheduling order of flows at a server. NC also offers a left-over service curve for FIFO-multiplexing servers: $\beta_{\theta}^{\text{l.o.}}$ [50]. Like the arbitrary multiplexing one given in Theorem 2.3, it is only applicable to a server. Thus, it allows for an analysis akin to the SFA presented in Section 2.3. Again, this raises the problem of paying multiplexing more than once if cross-flows share multiple consecutive servers with the analyzed foi. Effort focussed on improved analysis for sub-path sharing, however, an algebraic end-to-end FIFO analysis that reduces paying for cross-traffic multiplexing to a single location on the tandem does not exist. The most advanced algebraic tandem FIFO analysis is the so-called Least Upper Delay Bound (LUDB) [53]. In case of nested cross-traffic interference patterns, the LUDB suggests to convolve servers before removing cross-flows, subject to the nesting relation of flows. The latter is done by computing the FIFO left-over service curve. In Figure 6.6a, servers s_0 and s_1 are convolved before xf_1's arrivals are removed, resulting in $\beta_{\theta_1,\langle s_0,s_1\rangle}^{\text{l.o.}}$. Next, the left-over service curve at s_2, $\beta^{\text{l.o.}}_{\theta_2,s_2}$, is derived and both are convolved to the foi's end-to-end service curve. Finally, the delay bound can be derived which, in turn, requires to find the optimal solution for θ_1 and θ_2. Note, that this approach enforces segregated cross-flow arrival boundings according to the nesting, i.e., the composition penalty presented for the PMOOA in Section 3.2 also applies to the LUDB. If paths are not nested, this approach cannot be executed, e.g., if xf_2 additionally crossed s_1. In this case, [53] suggests to cut the tandem into several sub-tandems such that each sub-tandem sees nested interference only. Then, the foi's delay is derived for every sub-tandem; they are added up to the end-to-end delay bound. If there are multiple alternatives to cut a tandem (in case of prolonging xf_2 both links), all alternatives, i.e., all tandem matchings, are tested in an exhaustive fashion and the least among all resulting delay bounds is the final result. Unfortunately, the approach adding up partial delay bounds does not implement the PBOO principle. Therefore, the authors of the LUDB adapted their analysis in a follow-up article [3]. There, the foi is not cut anymore. Only cross-flows are cut such that the interference pattern is converted into a nested one. Then, an end-to-end left-over service curve can be derived for the foi. While this implements the PBOO principle, PMOO is not implemented as cutting cross-traffic requires deriving its arrival bound to be used at the cut.

The sub-tandem cutting of these approaches and the tandem matching of our new analysis are similar, yet, they also differ in some key aspects. First, our TMA does not require to result in tandems with nested interference because the PMOOA can handle overlapping sub-paths of cross-flows[10]. Our exhaustive approach thus results in more sub-tandem matching alternatives than the LUDB's cutting. Secondly, [53] and [3] are concerned with a tandem analysis only. They do not address the composition penalty of the compFFA we presented in Section 3.2, yet, both LUDB approaches are based on the idea of sub-path sharing and thus suffer from it, too. Moreover, they do not provide a technical solution for the potential problem of combinatorial explosion problem; [3] rather presents a heuristic to trade accuracy against computational effort. Applying the LUDB in the exhaustive tandem matching of this work, is, however, possible.

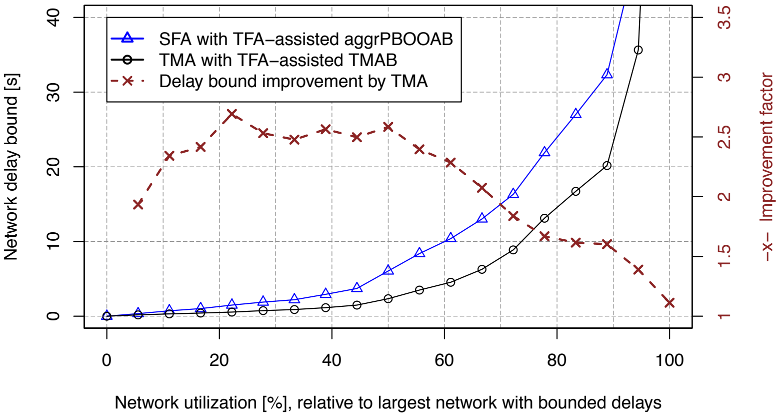
An optimization-based NC approach for tight FIFO-multiplexing feed-forward network analysis exists as well [15]. It transforms the NC description of the network into a Mixed Integer Linear Program (MILP) where the integer constraints encode branching of flows. This circumvents the step of explicitly extending a partial order to the set of all compatible total orders. However, the computational effort to solve the MILP for large networks does not seem computationally feasible either. Instead, the authors

---

[10] Here, we exclusively refer to the non-nested PMOOA left-over service curve of [75] on every tandem, independent of the actual cross-traffic interference pattern. Note, that the nested variant would have derived delay bounds equal to the TM alternative in Example 6.2, however, it is not applicable in slightly more involved networks. The TMA thus additionally solves searching for nesting patterns.



(a) Flat ER network with 32 devices.



(b) Hierarchical ER network with 32 devices.

Figure 6.8: TMA accuracy improvements in the ER sample networks.

advise to remove all constraints with integer variables in order to obtain an ordinary LP formulation that derives an upper bound. I.e., tightness is traded for computational effort; similar to the ULP.

6.2.3 Accuracy Evaluation

Numerical Experiments 1: Network Delay Bounds in ER Networks As with the evaluation of aggrAB in Section 6.1.1, we first benchmark TMA, our second proposal for optimization principles in algNC, against the latest results we reported for algNC, the SFA with TFA-assisted PBOOAB. Note, that TMA uses the TFA-assisted TMAB for compFFA step 1 and the TMA for step 2. For comparability, we use the same ER networks, a flat and an hierarchical one.

Figure 6.8 shows the results: Maximum delay bounds for the respective network uti-

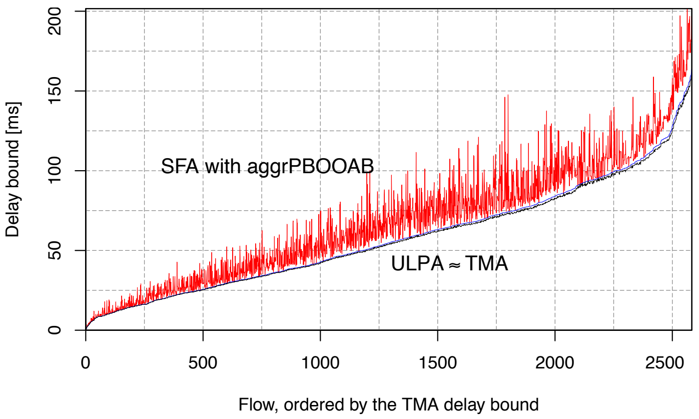


Figure 6.9: Flow delay bounds in the GLP network with 180 devices.

lization are improved. Similar to the previous evaluation, this reduction decreases with increasing utilization and thus increasing delay bounds. At 100%, the delay bounds become nearly identical and the improvement factor of both networks is at ~ 1. For smaller utilizations, the TMA achieves an improvement factor between ~ 1.25 and 1.75 in the flat network and between 1.5 and 2.75 in the hierarchical network. Both factors lead to considerable delay bound reductions.

Numerical Experiments 2: Flow Delay Bounds in GLP Networks Next, we extend the evaluation of Section 5.3.3 with our new TMA. I.e., we depart from the network delay bounds and investigate the impact on individual flow delay bounds.

Figure 6.9 shows analysis results for an Internet-like network with 180 devices. All the 2584 flows' end-to-end delay bounds are depicted. Sorting the bounds ascending by their TMA delays reveals two important observations:

- The ULP and the TMA delay bounds increase pretty much in lockstep and stay close to each other. I.e., the TMA does not have to compromise much on the delay bound accuracy. This holds true for all smaller network sizes as well.
- The SFA results oscillate wildly with a large amplitude. This behavior can be observed across all network sizes. In contrast to the TMA, the lack of consistency in the SFA's gap to the ULP bounds even prohibits using it as a proxy metric for accurate flow delay bounds.

Both observations are confirmed by the overall results of our experimental investigation. Figure 6.10 shows all evaluation results for GLP networks of sizes up to 180 devices, i.e., the delay bounds of 12376 flows across all network sizes shown in Table 5.2, left. The TMA accuracy is stable across the network size and stays within a small deviation from

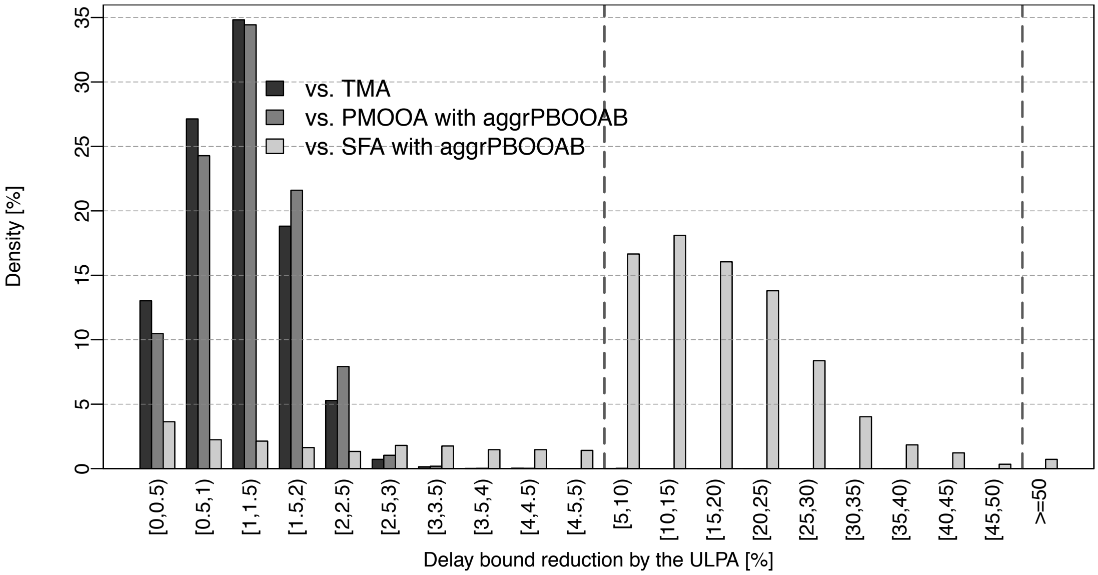


Figure 6.10: Flow delay bounds relative to the ULPA across nine network sizes computationally feasible to analyze with the ULPA (Table 5.2, left).

the ULP. Compared to the evaluation of the PMOOA in Section 5.3.3, Figure 5.4, we see that the accuracy has improved: The three leftmost TMA bars have increased while the following bars that depict larger deviations all decreased. Consequently, the amount of outliers has been reduced as well. Except for a single outlier, all derived bounds are at most 5% larger, mean and median are both at only 1.16%. The outlier deviates 8.23% from the ULP and thus is still more accurate than most of the SFA delay bounds.

6.2.4 Effort Considerations

The computational effort of exhaustive tandem matching becomes apparent when constructing the set of all TM alternatives, i.e., the entire TMA-internal model of the network that is used to derive the delay bound. We exemplify the construction on a tandem network with n servers and m flows; Figure 6.11 illustrates the steps we take and the amount of tandem matching alternatives in several, repeating steps of this procedure. Assume the worst case where each server is crossed by every flow, yet, equal to the SFA, cross flows are not aggregately bounded. Let us start on the foi's path $P(\text{foi})$ – we call this tandem $\mathcal{T}_{\text{orig}} = \langle s_1, \ldots, s_n \rangle$. The number of different matching alternatives for $\mathcal{T}_{\text{orig}}$ is already 2^{n-1}. As such an alternative derives the foi's end-to-end delay bound, each has between 1 and n distinct sub-tandems in order to cover all servers on $\mathcal{T}_{\text{orig}}$. In our example, we always proceed with tandem matching alternatives that separate the last hop and we bound a single segregated cross-flow's arrivals at this server. In the following

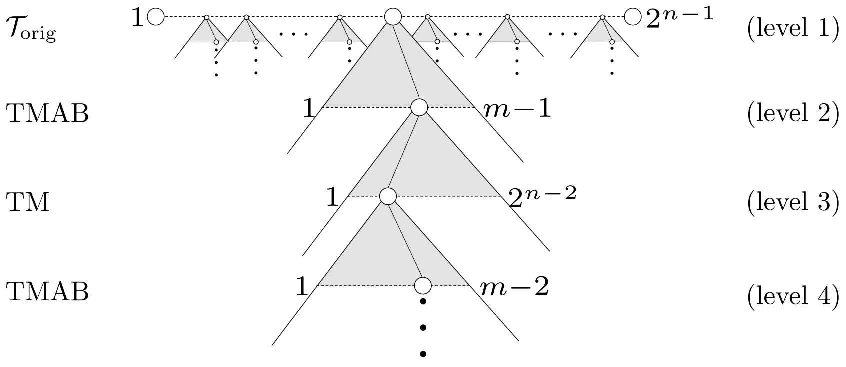


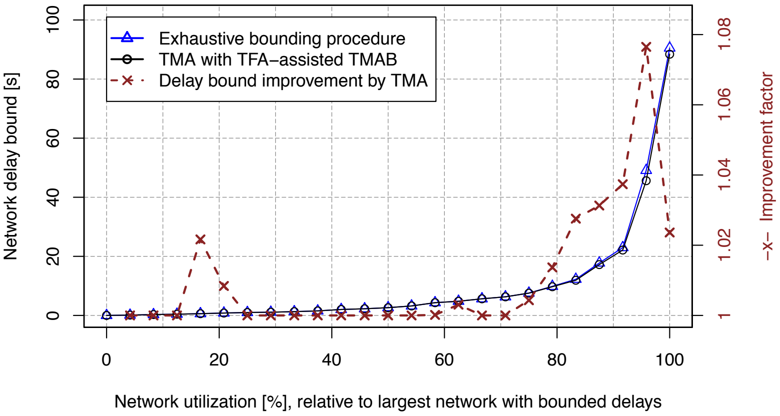
Figure 6.11: Search tree [1] for the n servers, m flows sample tandem representing all required arrival boundings.

step (level 2), TMAB is executed. The foi has $m-1$ cross-flows, defining the search space in the first TMAB step. To bound a single cross flow's arrival at level 2, a TMAB is applied to its path of length $n-1$, resulting in 2^{n-2} TM alternatives (level 3). Again, it is necessary to bound this flow's cross-traffic arrivals recursively (level 4). As the cross-flow under consideration does not interfere with itself and as we are on the foi's path, there are $m-2$ cross-flows to bound. In contrast to the shortening path lengths, this number will not decline until the arrival bounding terminates. We skip the remaining steps as they are repetitions of level 3 (with increasing $x \leq n$ in 2^{n-x}) and level 4. Continuing on this *trajectory* through the search tree, bounding a single cross-flow of a single sub-tandem of a single TMA alternative of $\mathcal{T}_{\text{orig}}$ already triggers $(m-1)+(n-1)\cdot(m-2)$ TMABs on the evenly numbered levels. Levels with odd numbers (≥ 3) scale this number by $2^{(n-2)!}$ TM alternatives until we eventually reach the end of the tandem.

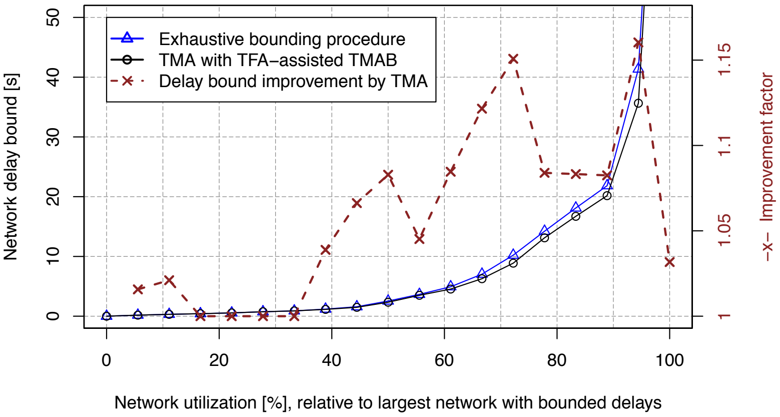
Obviously, TMA is prone to a combinatorial explosion similar to the extension of partial orders to compatible total orders as required for the LPA. However, in contrast to LPA, each trajectory starting from $\mathcal{T}_{\text{orig}}$ enables to derive a valid arrival bound. In principle, this would allow for a simple, yet flexible tradeoff between computational effort and delay bound accuracy. However, we opted for the exhaustive tandem matching and designed an efficient algorithm which keeps the computational effort still tolerable by exploiting the compositionality of the TMA (Section 7).

6.3 Combining Search Spaces of Optimization Principles

Last, we compare both optimization principles. Figures 6.3 and 6.8 showing the respective approach's improvement factor over the same algNC analysis reveal quite similar impact of the approaches; the pattern followed by the improvement factor is quite similar in both network types. Only the delay bounds in the hierarchical ER network show a visible



(a) Flat ER network with 32 devices.



(b) Hierarchical ER network with 32 devices.

Figure 6.12: Delay bound accuracy comparison for the algNC with different optimization principles.

difference. In Figure 6.12, we depict the relative performance of TMA and the exhaustive analysis. The improvement factor was chosen from the TMA's point of view. It does not fall below 1, i.e., TMA performs at least as good as the exhaustive analysis with aggrAB. This can be explained with a comparison of the search spaces covered by each principle. To be precise, we can deduce that the search space of exhaustive aggrAB is a subset of the TMA's one. The TMA exhaustively tests every TM alternative. Matching to the longest possible tandem coincides with the PMOOA; matching to the smallest possible tandems, i.e., single servers, coincides with the SFA. Transferring this approach to the TMAB means embedding it into the recursive aggregate cross-traffic arrival bounding scheme. There, the two special cases of TMAB coincide with aggrPMOOAB and aggrPBOOAB, the two alternatives defining aggrAB. Therefore, the TMA's search space completely covers the aggrAB's one. Not only does this explain the superiority of TMA delay bounds but also indicates larger effort for finding them.

In this section, we contributed insight on where to search for tightness improvements in algNC by identifying parts of its search space that have been neglected previously. As a result, delay bounds improve but the effort for algNC increases to the level of optNC – the analysis approaches become computationally infeasible. In fact, the accuracy evaluations shown in this section cannot be obtained with pure computational performance. In the next section, we provide efficiency improvements allowing us to derive delay bounds with algNC that incorporates optimization principles.

7 Efficiency of Algebraic Network Calculus Analysis

This chapter is dedicated to the efficient computation of algebraic network calculus analyses. First, we present two general enhancements for the feed-forward analysis: 1) Reduction of intermediate results by convolution of arrival bounds and 2) caching of intermediate arrival bounds. Both allow for a more efficient computation without sacrificing accuracy of results. Then, in Section 7.2, we implement these improvements in an algorithm for the TMA. In fact, without these improvements, the delay bounds presented in the previous section could not have been computed. We benchmark our new algorithm against ULPA, SFA and PMOOA. We show run times several orders of magnitude smaller than the ULPA.

The accuracy improvements for algNC presented in this thesis all require additional effort. AlgNC does, in contrast to the LPA, provide valid bounds in case the effort to be put into an analysis must be reduced. I.e., trading accuracy against effort is achievable in a straight-forward fashion; exhaustive approaches can be scaled back easily. However, we aim for the theoretically achievable bounds where exhaustive algNC procedures belong to the category of computationally infeasible analyses. This section provides work on efficiency of the algNC analysis such that effort vastly shrinks and accurate NC analysis finally becomes scalable.

7.1 Analysis Execution Efficiency

We reinforce the integration of the three substeps of compFFA step 1: backtracking dependencies, deriving the analysis equation and executing the computation. This integration still allows us to access all the useful intermediate results – a key difference to optimization-based analysis of optNC. The ULPA must execute separate steps for backtracking and computation in order to interface NC with the external linear program solver. We already showed in Section 5.3.3 that the choice of optimization software is crucial for the analysis performance and will also continue to investigate improvability of it.

7.1.1 Convolution of Intermediate Arrival Bounds

The first efficiency improvement to counteract combinatorial explosion in our algNC analyses strives for a reduction of intermediate results. As depicted in Figure 6.11, each TMAB matches tandems (oddly numbered levels), i.e., it derives alternative cross-traffic arrival bounds, one for every TM alternative. These arrival bounds are used to derive alternative left-over service curves that, in turn, are used to derive the arrival bounds required on the previous recursion level. In the integrated scheme we propose, the arrival bounds at every level of the recursion of compFFA step 1 are known. Moreover, it holds that for two alternative arrival curves α_1 and α_2 for a flow f, $\alpha_1 \otimes \alpha_2$ is also an arrival curve for f. Exploiting this insight, we need not traverse all trajectories through the search tree as we do not rank alternatives by the delay bound they eventually compute. This is in contrast to the LUDB and the straight-forward implementation of our optimization principles for algNC that compare alternative delay bounds at the end of the search. In fact, we do not rank alternative intermediate results at all, we combine them by convolution. This feature is unique to algNC as it retains the semantics of the NC model whereas the (U)LPA does not. Reduction to a single arrival bound per recursion level inhibits a combination any two adjacent levels' results. Thus, we strongly counteract the combinatorial explosion; potential search trajectories are narrowed down, search trees are thinned out.

7.1.2 Caching of Intermediate Arrival Bounds

The second improvement we propose is of technical nature. We extended the DiscoDNC with a caching mechanism for intermediate arrival bounds. Prerequisite for the retrieval of cached arrival bounds is their unique identification. The following parameters are required to identify an intermediate cross-traffic arrival bound:

Bounded Flows The set of flows whose arrivals are bounded by the curve.

Server The server at which flows are bounded with the cached curve. Remember that we need to segregate cross-flows according to the inlink they arrival on. Therefore, we may cache multiple arrival bounds per set of flows and per server. Separate entries according to the inlink help to maximize reuse and thus minimize derivation effort – similar to the considerations for an in-network SensorNC procedure (Section 4.5).

Flow of Interest The foi must be known in case the arrival bounding is started on its path and immediately proceeds with servers on it. This special case applies to the SFA and the TMA, e.g., see $\alpha_{s_2}^{f_0}$ in Section 2.3.2. This property must be indicated as these arrival bounds cannot be safely reused in any other case. There, the foi will be considered interference with the flows to bound, i.e., the arrival bound will be larger due to a violation of the PSOO principle.

We do not require the tandem analysis applied in compFFA step 2 to identify an arrival bound. In fact, the cache thus even allows algNC to share results between SFA and PMOOA with aggrAB (Section 6.1).

The cache fosters termination of recursive cross-traffic arrival bounding before reaching the source servers of flows – the previous termination condition segrPBOOAB as well as Algorithms 3.1, 4.2 and 6.1 (implementations of compFFA step 1). The search trees presented in the previous section thus become shorter as the length of search trajectories is reduced.

Convolution and caching of arrival bounds supplement each other well: The former ensures that a single arrival bound is derived on every recursion level of the compositional procedure of algNC. Therefore, it reduces the amount of intermediate bounds the cache has to store. Moreover, it allows the TMA algorithm to operate on a single, large search plane instead of the separate search tree branches shown in Figure 6.11. This "search plane" represents the same search space as before – the network remains the same – but it inhibits redundancies when searching it. Remember, that the TMA defines the most comprehensive search space of our algNC analyses incorporating optimization principles. In the remainder of this section, we will investigate the impact of our efficiency improvements on this analysis.

7.2 The Efficient TMA Algorithm

In this section, we provide a fast algorithm that integrates the transformation of the system description to all TM alternatives into the analysis itself. That is, TM is not executed statically in a step preceding the analysis. Intermediate results can then be used to dynamically reduce the amount of matching combinations as shown above. Although this reduces computational effort, our efficient TMA depicted in Algorithm 7.1 does not pay for this efficiency gain with less accurate delay bounds. It guarantees for the most accurate algNC bounds in feed-forward networks.

Algorithm 7.1 derives a foi's alternative end-to-end service curves for the TMA. Nomenclature follows Algorithm 6.1, i.e., lowercase letters denote variables and blackboard bold letters denote sets of variables. First, $\texttt{get}\mathbb{B}^{\texttt{l.o.}\mathbb{F}}_{\mathcal{T}_{\text{orig}}}$ is invoked with the foi's path $P(\text{foi})$, i.e., $\mathcal{T}_{\text{orig}} := P(\text{foi})$, and the foi itself – the algorithm needs to reverse the conceptual two steps of an analysis presented in Section 3.1. $P(\text{foi})$ constitutes the first tandem to match with 2^{n-1} disjoint sub-tandems, where n denotes the number of server on $P(\text{foi})$. For every sub-tandem we then derive the cross-traffic arrivals with TMAB (lines 32 – 37). Thus, we derive all TM alternatives within this recursive arrival bounding method. After the arrival bounding recursion has terminated (guaranteed by the cycle-free feed-forward network and flows of finite length), its result is used to derive the foi's left-over service curves by convolving the matched sub-tandems' $\beta^{\text{l.o.}}$ curves – similar to SFA but $\beta^{\text{l.o.}}$ are not necessary single server left-over service curves. In a subsequent step (not depicted in Algorithm 7.1), we derive delay bounds for all alternative end-to-end service curves for the foi on $P(\text{foi})$. In contrast to the individual linear programs of the LPA, all are valid. I.e., we chose the smallest result as final delay bound.

Algorithm 7.1: Efficient TMA Algorithm.

```
get$\mathbb{B}^{\texttt{l.o.}\mathbb{F}}_{\mathcal{T}_{orig}}$(Tandem $\mathcal{T}_{orig}$, Flow set $\mathbb{F}$)
    $\{\mathcal{T}_1, \ldots, \mathcal{T}_{2^{n-1}}\}$ = getSubtandems($\mathcal{T}_{orig}$); /* Disjoint divisions of $\mathcal{T}_{\text{orig}}$ */
    foreach Subtandem sequence $\mathcal{T} \in \{\mathcal{T}_1, \ldots, \mathcal{T}_{2^{n-1}}\}$ do
        foreach Subtandem $T \in \mathcal{T}$ do
            $\mathcal{A}_T^{\mathbb{F}}$ = TMArrivalBounding($T, \mathbb{F}$);
            $\beta_T^{\text{l.o.}\mathbb{F}}$ = PMOO$\beta^{\texttt{l.o.}}$($T, \mathcal{A}_T^{\mathbb{F}}$);
            $\beta_{\mathcal{T}}^{\text{l.o.}\mathbb{F}} \otimes= \beta_T^{\text{l.o.}\mathbb{F}}$;
        end
        $\mathbb{B}^{\text{l.o.}\mathbb{F}}_{\mathcal{T}_{\text{orig}}}$.put($\beta_{\mathcal{T}}^{\text{l.o.}\mathbb{F}}$);
    end
return $\mathbb{B}^{\text{l.o.}\mathbb{F}}_{\mathcal{T}_{\text{orig}}}$

TMArrivalBounding(Tandem $T$, Flow set $\mathbb{F}$)
    try{ return getCacheEntry($T$, $\mathbb{F}$) } /* Check for cached $\mathcal{A}_T^{\mathbb{F}}$ */
    $\mathbb{G}$ = xtxSegregation($T, \mathbb{F}$); /* Cross-traffic segregation */
    foreach xtxGroup $G \in \mathbb{G}$ do
        $\mathcal{A}_{\text{src}}^{G}$ = getSourceFlow$\alpha$s($G$); /* Arrival curve for $G$'s source flows */
        $\mathcal{A}_T^{\mathbb{F}}$.put($\mathcal{A}_{\text{src}}^{G}$);
        /* Flow aggregation requires equal inlinks */
        foreach Link $l \in T$.getInLinks() do
            $s = l$.getSource();
            $\alpha_T^G = \left(\alpha_s^G\right)'$ = OutputBound($s, G$);
            $\mathcal{A}_T^{\mathbb{F}}$.put($\alpha_T^G$);
        end
    end
    addCacheEntry($\mathcal{A}_T^{\mathbb{F}}$) /* Cache $\mathcal{A}_T^{\mathbb{F}}$ */
return $\mathcal{A}_T^{\mathbb{F}}$

OutputBound(Server $s$, Flow set $\mathbb{F}$)
    $\mathcal{T}_{\text{shared}}$ = getSharedServers($s, \mathbb{F}$); /* Aggregate flows on shared servers */
    $\mathbb{B}^{\text{l.o.}\mathbb{F}}_{\mathcal{T}_{\text{shared}}}$ = get$\mathbb{B}^{\texttt{l.o.}\mathbb{F}}_{\mathcal{T}_{orig}}$($\mathcal{T}_{\text{shared}}, \mathbb{F}$);
    $\alpha_{\mathcal{T}_{\text{shared}}} = \alpha^{G}_{\mathcal{T}_{\text{shared}}.\texttt{getSource()}}$; /* Arrival bound of flows in $G$ at $\mathcal{T}_{\text{shared}}$'s start */
    $B^{\text{TFA}}$ = TFAbacklogBound($s$)
    foreach $\beta^{\text{l.o.}\mathbb{F}}_{\mathcal{T}_{\text{shared}}} \in \mathbb{B}^{\text{l.o.}\mathbb{F}}_{\mathcal{T}_{\text{shared}}}$ do
        $\left(\alpha_s^{\mathbb{F}}\right)' \otimes= \alpha_{\mathcal{T}_{\text{shared}}} \oslash \beta^{\text{l.o.}\mathbb{F}}_{\mathcal{T}_{\text{shared}}}$; /* Complexity reduction: Convolve all $\alpha'$ */
        $\alpha_{\mathcal{T}_{\text{shared}}}$.cap($B^{\text{TFA}}$); /* TFA assistance */
    end
return $\left(\alpha_s^{\mathbb{F}}\right)'$
```

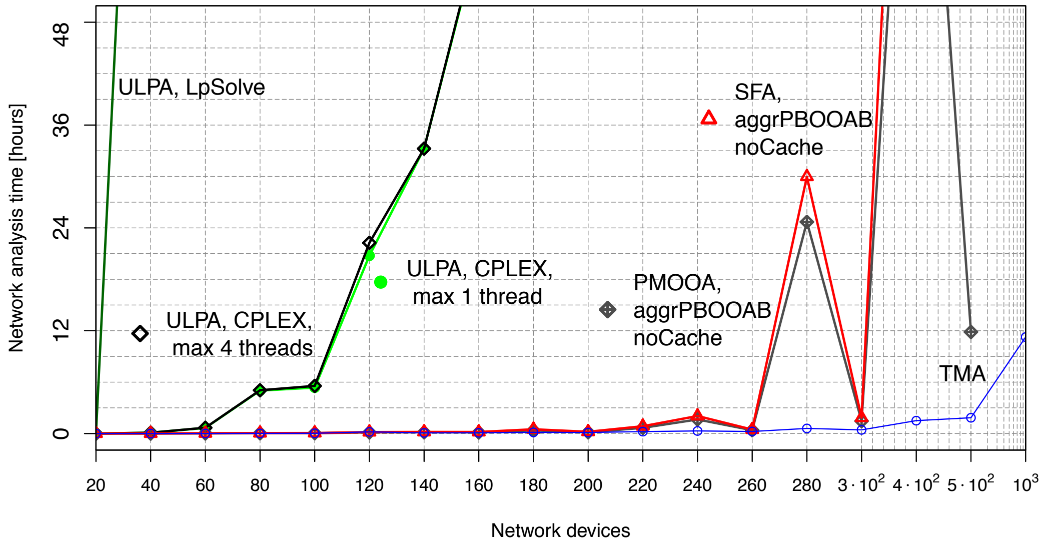


Figure 7.1: Computational Effort Evaluation: Graphical comparison of network analysis times of optNC (ULPA), and algNC without (SFA, PMOOA) and without our improvements (TMA).

Computational Effort Reduction Access to intermediate results, i.e., the output of the three individual functions in Algorithm 7.1, allows the TMA to implement several efficiency improvements:

1. The aggregate cross-traffic arrival bounding on tandems shared by multiple flows (lines 32 – 43, $\mathcal{T}_{\text{shared}}$) naturally counteracts cross-traffic segregation. Aggregately bounding cross-flows prevents the PSOO effect to arise and the principle to be violated by the TMA. From the computational efficiency point of view, less mutual interference assumption means less cross-traffic arrival bounding instances. I.e., it prevents recursively bounding cross-flows of cross-flows; all with exhaustive TMAB. This vastly reduces the effort as our considerations concerning combinatorial explosion show (Section 6.2.4).
2. Every recursion level has its $\mathcal{T}_{\text{shared}}$ that TMAB matches tandems to. Each of the resulting left-over service curves yields a cross-traffic arrival bound to be used in the previous recursion level. The second efficiency improvement is to convolve all of the alternative arrival bounds when $\mathcal{T}_{\text{shared}}$ is analyzed (lines 39 – 42).
3. The third improvement is arrival bound caching that allows for storage and retrieval of intermediate results (lines 15 and 30) which are plentiful in the compositional analysis. It impacts the TMA when alternative matchings share sub-tandems that require equal cross-traffic arrival boundings.

Devices	ULP, CPLEX	SFA,	PMOOA,	TMA
	(4 threads)	aggrPBOOAB (1 thread)		(1 thread)
20	00:00:13	00:00:07	00:00:02	00:00:11
40	00:06:04	00:00:30	00:00:22	00:00:13
60	00:40:41	00:01:49	00:01:18	00:00:38
80	05:03:00	00:03:01	00:02:21	00:01:14
100	05:33:44	00:03:39	00:02:48	00:01:54
120	22:15:22	00:10:19	00:08:22	00:11:44
140	33:14:00	00:11:18	00:09:05	00:04:48
160	58:20:54	00:10:55	00:08:16	00:05:35
180	∼13 days	00:29:52	00:22:28	00:08:19
200	–	00:12:15	00:09:18	00:07:23
220	–	00:51:16	00:40:04	00:13:43
240	–	02:02:55	01:38:44	00:18:18
260	–	00:27:49	00:23:34	00:14:26
280	–	29:58:21	24:41:47	00:35:53
300	–	01:50:44	01:27:45	00:25:50
400	–	128:27:16	100:18:33	01:30:25
500	–	–	–	01:50:51
1000	–	–	–	11:15:34

Table 7.1: Computational effort evaluation: network analysis times.

Summing up, we designed an algebraic, compositional NC analysis that is strictly superior to the existing SFA as well as the PMOOA. The more comprehensive procedure of transforming a network description to analysis-internal models allows optimal application of algNC operations and therefore increases the accuracy of delay bounds. Moreover, with Algorithm 7.1 we provide an efficient solution to mitigate our analysis' potentially high computational effort.

7.2.1 Effort Evaluation

Numerical Experiments With TMA deriving accurate delay bounds, we now evaluate the effort involved in our new analysis in order to show that the computational efficiency improvements incorporated into Algorithm 7.1 indeed have a crucial impact. Figure 7.1 and Table 7.1 depict the time to complete a network analysis for our set of Internet-like topologies. Although the experimental results are subject to fluctuations due to the high degree of randomness in network and flow creation, we can draw very clear conclusions. Computational effort and thus the time to analyze grows (fast) with the network size. The ULPA's effort is vastly reduced in contrast to the LPA, yet, our evaluation reveals that it still becomes computationally infeasible quickly. With 180 network devices, the

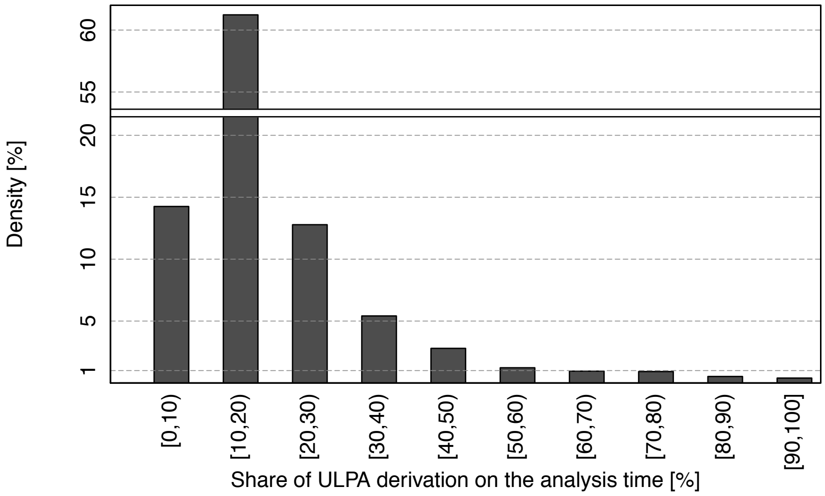


Figure 7.2: Share of the DiscoDNC linear program creation on the ULPAs.

analysis execution already took ~ 13 days. The larger the network, the more effort is required to trace the influence of flows on each other. Yet, among the two steps of the ULPA, deriving the constraints for the linear program with the DiscoDNC and optimizing it with CPLEX, the latter takes on average 81.53% of the time (cf. Figure 7.2). Faster ULPA optimization could not be achieved with CPLEX; increasing the parallelism from 1 to 4 threads[11] even yielded slightly slower overall network analysis times due to the overhead for thread synchronization. Figure 7.1 depicts the single-threaded CPLEX version (dotted line). As mentioned, it is slightly below the ULPA line (blank diamond) that shows the results when running CPLEX with up to 4 threads. The ULPAs seem to become very complex. The ULPA file sizes may hint at its complexity; we observed CPLEX lp files of several gigabytes in size.

Analyzing large feed-forward networks in practice remains solely possible with algebraic, compositional NC analyses. However, Figure 7.1 and Table 7.1 also show that the SFA with aggrPBOOAB becomes computationally infeasible at 400 network devices (994 servers, 3976 flows; 13 days for analysis) – although the aggregate arrival bounding reduces the effort compared to the segrPBOOAB. Our TMA scales best with the network size thanks to the improvements of Algorithm 7.1: Aggregate bounding of cross-traffic as well as the convolution and caching its intermediate arrival bounds. The TMA not only outperforms the other analyses, it also seems more resilient to the random factors of our network creation. Most notable are the changes in network analysis times when increasing the amount of devices from 260 to 300 in a stepsize of 20 devices. The first

---

[11] Note, that among the tools employed in this evaluation, only CPLEX offers an option for parallel execution. We chose four threads to match the number of CPU cores.

increase only adds 18 servers and 72 flows to the network but the entanglement of flows is considerably more complex such that the SFA run-time grows by an order of magnitude, from ~ 30 minutes to ~ 30 hours. The next step increases the amount of servers by 130 and the flows by 520, however, this seven times larger growth of the network size results in a large decrease of the SFA run-time, down to $\sim$2 hours. The PMOOA reveals a nearly equal behavior but the TMA, on the other hand, has only a small oscillation, jumping from 15 minutes to 36 minutes and back to 26 minutes. The TMA scales much better with the network size and outperforms the other analyses by several orders of magnitude. It is now possible to analyze a GLP network with 1000 devices. The sample network of this size had 3626 servers and 14504 flows, i.e., more flows than analyzed with the ULPA in total in $\sim$18 days, yet, all in a single network with more comprehensive dependencies to backtrack and convert into an analysis equation. This makes the TMA by far the fastest accurate algNC analysis and enables algNC to analyze large feed-forward networks.

7.3 Network Calculus on Compact Domains

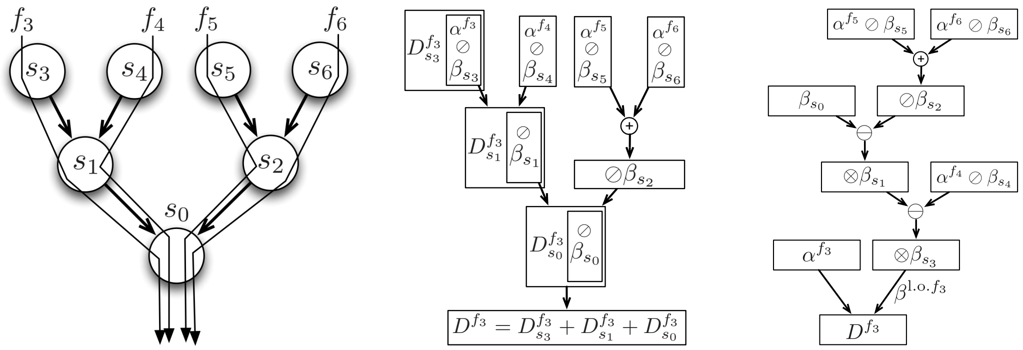
As a last efficiency improvement, we propose a solution to the problem of exploding curve complexity in an algNC. If a (min,+)-operation is applied to pseudo-periodic input curves f and g, then the resulting output curve h's period grows to the least common multiple (hyperperiod) of f and g's periods. Hence, subsequent operations on h become more complex. In a network analysis, this growth of complexity results in prohibitive effort fast. Our solution to this problem is based on the following insight from the literature [37]: The eventual performance bound derivation only requires a finite, initial part of the curve. We propose a solution to derive more accurate upper bounds on exactly this part of curves. Moreover, our solution achieving this is more general, allowing for the application to SFA and PMOOA with nested interference. We evaluate our findings with the case study of an avionics in-cabin network.

One of the main assumptions of this thesis so far was that arrivals are shaped to multi-token-buckets, $\alpha \in \mathcal{F}_{\mathrm{mTB}}$, and service curves belong to the set of multi-rate-latency curves, $\beta \in \mathcal{F}_{\mathrm{mRL}}$. These curves are widely used in practice, yet, they may already be the first accuracy tradeoff. For instance, a different approach to characterize the arrivals of task activations is the (p, j, t_δ)-model where p denotes the period, j the jitter and t_δ the minimum inter-arrival time. The parameters are used to create a staircase arrival curve $\alpha_{p,j,t_\delta}(d) = \min\left(\left\lceil \frac{d+j}{p} \right\rceil, \left\lceil \frac{d}{t_\delta} \right\rceil\right) \in \mathcal{F}_0$ [81, 59]. For (strict) service curves, the TDMA can be best described with the (k, c, b)-model where k is the slot length available for submission, c is the TDMA cycle length this slot is embedded into, and b is the total bandwidth [58]. This resource model results in a service curve $\beta_{k,c,b}(d) =$

$\left(\left\lfloor \frac{[d-c+k]^+}{c} \right\rfloor \cdot k + \min\left([d-c+k]^+ \bmod c,\, k\right)\right) \cdot b$. Bounding these curves with a multi-token bucket or a multi-rate latency, respectively, constitutes an overapproximation. The inaccurate model thus forces the NC analysis to account for data arrivals that are actually known not to occur but are within the region bounded by α. Moreover, the approximated service prevents NC from exploiting capabilities not captured in β. Therefore, accuracy of NC bounds also depends on the accuracy of the model, i.e., even the LPA only derives tight bounds within the given network description. The optNC analyses strictly rely on $\alpha \in \mathcal{F}_{\mathrm{mTB}}$ and $\beta \in \mathcal{F}_{\mathrm{mRL}}$, so does the non-nested PMOOA. The left-over service curve of Theorem 2.3, in contrast, does not. It allows the SFA as well as the nested and the sink-tree PMOOA to analyze networks with arrival and service curves are from $\mathcal{F}_0$. However, for practical reasons this potential has not yet been exploited.

The Hyperperiod Problem The above curves, α_{p,j,t_δ} and $\beta_{k,c,b}$, are piece-wise linear (PWL) and pseudo-periodic. Due to the former, we have $\alpha_{p,j,t_\delta} \notin \mathcal{F}_{\mathrm{mTB}}$ and $\beta_{k,c,b} \notin \mathcal{F}_{\mathrm{mRL}}$ and the latter causes the hyperperiod problem [17]. Let α_{p,j,t_δ} and $\beta_{k,c,b}$ be the input curves to a (min,+)-algebraic operation, then the resulting curve is PWL and pseudo-periodic, yet, its period is the least common multiple of the input curves' periods. I.e., with the provided curve models we get an output curve period of $\mathrm{lcm}(p, c)$. In the algNC cross-traffic equation derived in compFFA step 1b and solved in step 1c as well as the foi equation of step 2 (Section 3.1), the hyperperiod problem renders the analysis computationally intractable. It can be alleviated with overapproximated curves. E.g., multi-token buckets and multi-rate latencies circumvent it and aligning the periodicity of curves such that they result in a small lcm inhibits uncontrolled growth. However, these countermeasures compromise on model accuracy and thus eventually on performance bound accuracy as well.

The Compact Domains Solution Closure of this class of curves is shown in [17], where the authors also show that NC operators suffer from the hyperperiod problem. Yet, in our work, we observe that this complete characterization is rarely required when bounding delay or backlog. The bounds are commonly found on the initial parts of curves. Therefore, we propose to alleviate the hyperperiod problem by using NC on compact domains, i.e., restricting curves to a bounded domain $[0, K]$ instead of $[0, +\infty)$. The sizes of a curve's compact domain is crucial for both of the aspects in the focus of in this thesis: delay bound accuracy and computational effort. On the one hand, compact domains need to be sufficiently large such that performance bounds are not negatively affected. On the other hand, we want to decrease computational effort and thus strive for small compact domains. We achieve a balance between these aspects by analyzing the algNC analysis equation and then deriving a bound on the domain length ensuring that delay and backlog



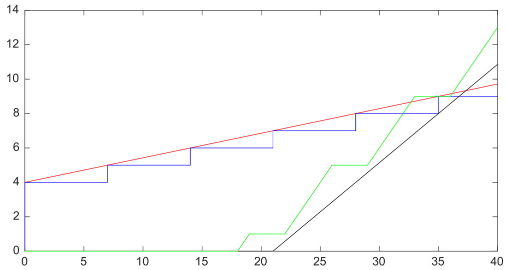
(a) Sample Sink-Tree Network. (b) TFA Delay. (c) Nested PMOOA Delay.

Figure 7.3: Sample network and its analysis equations for TFA and nested PMOOA.

bounds are enclosed. This initial domain bound is then used to derive the required domain lengths of curves occurring in the entire equation. Maintaining a domain of sufficient length for each of these curves is crucial, as they all influence the final performance bound calculation. Any loss in precision, therefore, affects the accuracy of the derived performance bounds. Here, we make use of the backtracking scheme of compFFA step 1a to propagate the restrictions through the analysis equation. In the remainder of this section, we derive the compact domain required for each operation's input curves in order to guarantee for the known domain requirement at its output. This makes our approach applicable to any algNC analysis using the operations of Section 2.1.3. It is key for significant improvements in the computational efficiency of these analyses without sacrificing the accuracy of the NC network model.

7.3.1 Related Work

Reducing precision of curves to finite intervals has been introduced and empirically evaluated in the work of [80, 81]. It is, however, preliminary, as it lacks proofs for the preservation of accuracy of performance bounds. [37] derived conditions for the lengths of finite intervals and proved that they preserve accuracy of the performance bounds. However, these proofs only hold for the input A / output A' relations of the greedy-processing component (GPC) commonly used in Real-Time Calculus (RTC) [83]. Although algNC and RTC are based on the same dioid algebra, the component-level of abstraction in the proofs imposes crucial limitations for application in algNC. The nested PMOOA that implements the PBOO and the PMOO principle does not preserve the component-structure of the network (Figure 7.3c). The only analysis preserving it is the TFA (Figure 7.3b). It can directly exploit the work of [37], however, TFA is known to derive inferior delay bounds (see Sections 5.3.1 and 5.3.2).



(a) Arrival curves α and PMOOA $\beta^{\text{l.o.}f_3}$: steps functions (blue, green); linear (red, black).

	Steps	Linear
TFA	38	42.5
PMOO	25	28
ULP	–	28

(b) Delay Bounds of the Example.

Figure 7.4: Input curves and computed delay bounds.

Example 7.1. We derive the delay bounds for the small sink tree example of Figure 7.3. All arrival curves were set to $\alpha_{p,j,t_\delta} = \alpha_{7,12,0}(d)$ as this parameter setting results in the curves' first step to be as long as the following ones. Therefore, the token bucket arrival curve $\gamma_{r,b} = \gamma_{\frac{1}{7},7}$ is a tight overapproximation. It passes though all starting points of each step of α_{p,j,t_δ}. See Figure 7.4a where $\alpha_{7,12,0}$ and $\gamma_{\frac{1}{7},7}$ are depicted. Strict service curves were set to $\beta_{1,0}$, i.e., only the overapproximation of arrivals causes the difference in results. Table 7.4b shows the delay bounds derived by different NC analyses for the respective curve settings[12]. Even in this simple example, there is a $\sim 10\%$ difference between the end-to-end delay bounds of the accurate model and the linear overapproximation. As expected, the TFA performs worse than the PMOOA that, in turn outperforms the ULP if executed with the more precise step functions.

7.3.2 Deriving Compact Curve Domains

In this section, we introduce the idea of compact domains for algNC by departing from the component-wise view. Instead, we derive domains on the actual operations applied in algNC equations.

Preliminaries First, we need to transform the left-over service curve derivation of Theorem 2.3 to apply the (max,+)-convolution instead of the non-decreasing upper closure. The (max,+)-convolution, denoted as $\overline{\otimes}$, is defined as

$$\left(f \overline{\otimes} g\right)(d) = \sup_{0 \le \tau \le d} \left\{ f(d-\tau) + g(\tau) \right\}$$

---

[12] The DiscoDNC does not support PWL, pseudo-periodic step functions. Therefore, we used the MPA toolbox [58] in this section.

and with it, we can compute the left-over service curve as

$$\beta_s^{\text{l.o.}\mathbb{F}_1} = \beta_s \ominus \alpha^{\mathbb{F}_0} = \left(\beta_s - \alpha^{\mathbb{F}_0}\right) \overline{\otimes} \lambda_0$$

with $\lambda_0(t) = 0$:

$$\begin{aligned}
\left(\left(\beta_s - \alpha^{\mathbb{F}_0}\right) \overline{\otimes} \lambda_0\right)(d) &= \sup_{0\le s\le d} \left\{\left(\beta_s - \alpha^{\mathbb{F}_0}\right)(d-s) + \lambda_0(s)\right\} \\
&= \sup_{0\le s\le d} \left\{\left(\beta_s - \alpha^{\mathbb{F}_0}\right)(d-s) + 0\right\} \\
&= \sup_{0\le s\le d} \left\{\left(\beta_s - \alpha^{\mathbb{F}_0}\right)(d-s)\right\} \\
&(s := d-u) \\
&= \sup_{0\le (d-u)\le d} \left\{\left(\beta_s - \alpha^{\mathbb{F}_0}\right)(d-(d-u))\right\} \\
&= \sup_{0\le (d-u)\le d} \left\{\left(\beta_s - \alpha^{\mathbb{F}_0}\right)(u)\right\} \\
&\text{(reverse the supremum's search direction)} \\
&= \sup_{0\le u\le d} \left\{\left(\beta_s - \alpha^{\mathbb{F}_0}\right)(u)\right\} \\
&= \left(\beta_s \ominus \alpha^{\mathbb{F}_0}\right)(d)\,.
\end{aligned}$$

Next, we define the compact domain as well as arrival and service curves compacted to this closed interval.

Definition 7.1. Let $\mathbb{D} = [0, K]$ be a closed and bounded interval. If a curve κ provides a mapping $\kappa : [0, K] \to \mathbb{R}_\infty^+$, we denote $\mathbb{D}$ as its compact domain.

In this setting we define the following notation:

- Arrival curves α and strict service curves β of $\mathcal{F}_0$ operate on an unrestricted domain, they provide a mapping $\mathbb{R} \to \mathbb{R}_\infty^+$ (Section 2.1.1).
- The corresponding curves α^K and β^K are restricted to the compact domains $[0, K_\alpha]$ and $[0, K_\beta]$.

For unrestricted curves α and β, we define the following approximations:

- An arrival curve $\overline{\alpha}$ is an overapproximation of an unrestricted arrival curve α if $\forall d \in \mathbb{R} : \overline{\alpha}(d) \;\ge\; \alpha(d)$.
- A service curve $\underline{\beta}$ is an overapproximation of an unrestricted service curve β if $\forall d \in \mathbb{R}_\infty^+ : \underline{\beta}(d) \;\le\; \beta(d)$.

The intuition for these overapproximations is as follows: Arrival curve $\overline{\alpha}$ allows more data arrivals than originally specified with α and in case of a service curve $\underline{\beta}$, the assumed service applied to a flow is decreased when compared to the original service curve β. Vice versa, underapproximations $\underline{\alpha}$ and $\overline{\beta}$ are more optimistic than original curves and thus can only lower bound these. They are defined analogously.

In the following, the concrete form of approximations is irrelevant as long as the above conditions are met, e.g., one could use specific traffic patterns as proposed in [59, 8], approximations based on single linear segments [81], or minimum-composed sets of linear segments [49]. The only assumption we have is that, in the long term, there is enough service available to process all inputs.

Enclosure of Backlog and Delay Bounds Deriving a performance bound, backlog or delay, constitutes the eventual step of any NC analysis; see 7.3b and 7.3c. Working on compact domains, we need to guarantee that these are large enough to enclose the bounds. Otherwise, the bounds derived on the compact domain are not valid. The following corollary is a direct consequence of Theorem 2.1. It ensures enclosure of delay and backlog bounds.

Corollary 7.1. (Backlog and Delay Bound Enclosure) *Assume the delay and backlog bound for an unrestricted arrival curve α and an unrestricted service curve β to be known. That means we also know the values of the parameters u, s and r for which the suprema in Theorem 2.1 are assumed. Then, we can derive the domains of the restricted curves α^K and β^K by*

$$K_\alpha = \max(u,\, r) \quad , \quad K_\beta = \max(u + \tau,\, r) \ .$$

Corollary 7.1's prerequisite to know the performance bounds for α and β a priori is crucial. Clearly, both are only known a posteriori, i.e., after we carried out the analysis with unrestricted curves. Yet, the analysis with unrestricted arrival and service curves imposes a high analysis effort, when being performed with accurate network models. This requirement is thus contradicting our aim to execute the analysis with less computational effort, it constitutes a seemingly vicious circle.

We break this circle by evaluating delay and backlog bounds both with linearly underapproximated and overapproximated arrival and service curves. Figure 7.4a provides an example for such an arrival curve overapproximation (for the underapproximation, it is simply shifted to the opposite side of the step function). This step can be executed very efficiently if the chosen approximations are token buckets for arrival curves and rate latencies for service curves. We execute an entire analysis with approximated curves in order to obtain $\overline{\alpha}$, $\underline{\alpha}$ as well as $\overline{\beta}$ and $\underline{\beta}$ that are required to bound delay and backlog.

More precisely, we can compute approximations for delay and backlog and use these to bound the domains of α and β:

1. With the pair $\underline{\alpha}$ and $\overline{\beta}$, we compute the estimates $\underline{D}$ and $\underline{B}$ for bounding the delay and backlog. As the involved curves $\underline{\alpha}$ and $\overline{\beta}$ are underapproximations of the unrestricted curves α and β, these estimates are *lower bounds.*
2. For the bounds $\underline{D}$ and $\underline{B}$, we compute the largest value of d, such that for the function values beyond d only smaller delay and backlog bounds can be derived. This is the domain bound required to enclose D and B computed from the original curves.
3. With the pair $\overline{\alpha}$ and $\underline{\beta}$, we similarly compute the estimates $\overline{D}$ and $\overline{B}$ that denote upper bounds on the delay and backlog that would have been derived with α and β.

We formalize the domain bound finding as follows:

Theorem 7.1. *We define the following variables:*

$$\begin{aligned} U &:= \sup\left\{d \geq 0 \ : \ \overline{\alpha}(d) \geq \underline{\beta}(d + \underline{D})\right\} \text{ and} \\ R &:= \sup\left\{d \geq 0 \ : \ \overline{\alpha}(d) - \underline{\beta}(d) \geq \underline{B}\right\}. \end{aligned}$$

For $K_\alpha \geq max(U, R)$ and $K_\beta \geq max(U + \overline{D}, R)$ the domain bounds K_α and K_β are sufficiently large such that accurate delay and backlog bounds can be computed with α^K and β^K.

Proof. We defined the largest pseudo-inverse of the lower delay and backlog bound as follows:

$$\begin{aligned} u^* &:= \sup\left\{\mathbf{u} \geq 0 : \underline{D} \leq \sup_{u \geq \mathbf{u}}\left\{\inf\left\{\tau \geq 0 \ : \ \alpha'(u) \leq \beta'(u + \tau)\right\}\right\}\right\}, \\ r^* &:= \sup\left\{\mathbf{r} \geq 0 : \underline{B} \leq \sup_{r \geq \mathbf{r}}\left\{\alpha'(r) - \beta'(r)\right\}\right\}. \end{aligned}$$

Sub- and superadditivity of curves α and β, together with $\underline{D} \leq \overline{D}$ implies that $u^* \leq U$. Thus, U is a safe upper bound for guaranteeing delay bound enclosure in curve α.

Exploiting $u^* \leq U$ together with the inequality $D \leq \overline{D}$ yields $u^* + D \leq U + D \leq U + \overline{D}$. Note that D is a bound on τ in Theorem 2.1. This implies that $U + \overline{D}$ is an upper bound for guaranteeing delay bound enclosure in curve β.

With r^* being bounded by R, we get that $K_\alpha = max(U, R)$ and $K_\beta = max(U + \overline{D}, R)$ also guarantee backlog bound enclosure. It can be found if α and β are restricted to $[0, K_\alpha]$, resp. $[0, K_\beta]$, i.e., with the curves α^K and β^K. ☐

An alternative bound on the domain size that could also be used here is given in [37]. The authors propose to use the maximum backlogged period bound $bp(\overline{\alpha}, \underline{\beta})$. However, this leads to significantly larger domain sizes K_α and K_β, especially when the utilization is high.

Propagating Compact Domains to the Network Description's Curves We now know how to efficiently derive a bound on the compact domains for arrival and service involved in bounding delay and backlog. Next, we want to use this insight to increase the computational efficiency of the entire analysis. We aim to derive the compact domain for each arrival and service curve of the network description. The smaller these domains, the faster the analysis can be executed in comparison to the unrestricted curves because domain restrictions prevent ever growing hyperperiods. The part growing over the compact domain is simply cut off. This requires to backtrack through the analysis equation, i.e., in our integrated scheme for cross-traffic arrival bounding (compFFA step 1) another backtracking through the network. Either way, we traverse the servers and thus the algNC operations in the opposite direction of the analysis. We can take the domain bound required at the output of an operation, i.e., for its output curve, and derive the domain bounds for this operation's input curves. We start this scheme with the above bounds guaranteeing the enclosure of delay and backlog bound.

Let $\odot$ be a binary algNC-operator, i.e., $\odot \in \{\otimes, \overline{\otimes}, \oslash, \min, +, -\}$, and let κ be the output curve of the operation. When backtracking, we know the bound for the compact domain of output curve κ, K_κ. Moreover, from the backtracking itself equation we know $(\alpha \odot \beta)\,(d) = \kappa(d)$, i.e., the actual operation $\odot$ and both its input curves α and β[13]. In the following, we derive the conditions for $(\alpha^K \odot \beta^K)(d) = \kappa(d)$, i.e., the domain length K_α and K_β such that κ remains defined on its domain K_κ. This directly implies that $\forall d \in [0, K_\kappa] : \kappa^K(d) = (\alpha^K \odot \beta^K)(d) = (\alpha \odot \beta)(d) = \kappa(d)$ and that the eventually derived performance bounds are safely enclosed.

Compact Domains for $\odot \in \{\otimes, \overline{\otimes}, \min, +, -\}$ Given the operation $(\alpha \odot \beta)(d) = \kappa^K(d)$ and $\odot \in \{\otimes, \overline{\otimes}, \min, +, -\}$, we can compact the domains of α and β to $K_\alpha = K_\beta = K_\kappa$. This is a direct consequence from the operator definitions; computing $\kappa(d) = (\alpha \odot \beta)(d)$ only requires to evaluate α and β on $[0, d]$ and from κ^K we know that $d \leq K_\kappa$.

Compact Domains for $\oslash$ Deriving the compact domains of the (min,+)-deconvolution is more involved.

---

[13] For, e.g., aggregation both curves might be arrival curves. This does not impact the derivations given in this section. Therefore, we refrain from a detailed distinction of cases for the ease of presentation.

Theorem 7.2. (Domain bounds for min-plus deconvolution) *As before, let K_κ be the domain size of the output curve. We define U to be the largest input value for which $\overline{\alpha}(U + K_\kappa) \geq \underline{\beta}(U)$, i.e.,*

$$U \quad := \quad \min\left(0, \sup\{d \geq 0 : \overline{\alpha}(d + K_\kappa) - \underline{\beta}(d) \geq 0\}\right) .$$

This yields that for $K_\alpha = U + K_\kappa$, $K_\beta = U$ and $\forall d \in [0, K_\kappa]$, we have $\left(\alpha^K \oslash \beta^K\right)(d) = (\alpha \oslash \beta)(d)$.

Proof. For the output curve κ, we are only interested in function values for the compact domain $[0, K_\kappa]$. By exploiting U as defined above, we obtain that

$$\forall d \leq K_\kappa : \sup_{u>U}\{\alpha(d+u) - \beta(u)\} < 0.$$

This shows that function values from α and β that are beyond $U + K_\kappa$, resp. U, turn irrelevant as $\alpha(d+u) - \beta(u)$ becomes negative for these input values. Hence, the positive supremum of $\alpha^K(t+u) - \beta^K(u)$ can only be found for input values to α and β from the compact domain $[0,\ K_\alpha := U + K_\kappa]$, resp. $[0,\ K_\beta := U]$. As α^K coincides with α for $d \in [0, K_\alpha]$ and β^K coincides with β for $d \in [0, K_\beta]$, the above construction yields that for $d \leq K_\kappa$: $(\alpha^K \oslash \beta^K)(d) = (\alpha \oslash \beta)(d)$. ☐

As an example, we show how the domain bounding parameter U can be found when using overapproximations from $\mathcal{F}_{\mathrm{TB}}$ for arrival curves and $\mathcal{F}_{\mathrm{RL}}$ for service curves. Accordingly, we define an overapproximation $\overline{\alpha}(t)$ to an input curve α as:

$$\alpha(d) \leq \overline{\alpha}(d) = \begin{cases} \max\left(0,\ N_\alpha + \rho \cdot d\right) & \text{if } \overline{\alpha} > 0 \\ 0 & \text{otherwise} \end{cases}$$

and let $\underline{\beta}(t)$ be defined analogously with an rate-latency curve such that $\beta(t) \geq \underline{\beta}(t)$ holds. The slope of $\underline{\beta}(t)$ is denoted σ and the intersection with the y-axis is N_β. In this setting, we bound U for a constant K_κ as follows:

$$\begin{aligned}
&\alpha(u+d) - \beta(u) \geq 0 \\
&\text{with } \alpha(d) \leq \overline{\alpha}(d) \text{ and } \beta(d) \geq \underline{\beta}(d) \text{ we have} \\
&\overline{\alpha}(u+d) - \underline{\beta}(d) \geq \alpha(u+d) - \beta(u) \geq 0\,. \\
&\text{With } 0 \leq d \leq K_\kappa \text{ we get} \\
&\overline{\alpha}(u+K_\kappa) - \underline{\beta}(u) \geq \overline{\alpha}(u+d) - \underline{\beta}(d)\,. \\
&\text{The definitions of the linear bounding functions yield:} \\
&N_\alpha + \rho(u+K_\kappa) - (N_\beta + \sigma u) \geq 0
\end{aligned}$$

thus $u \leq \frac{N_\beta - N_\alpha + K_\kappa \cdot \rho}{\rho - \sigma}$
and for $N_\beta < N_\alpha$ and $\rho < \sigma$ this is equal to$u \leq \frac{N_\alpha - N_\beta + K_\kappa \cdot \rho}{\sigma - \rho}$
such that for $U = \frac{N_\alpha - N_\beta + K_\kappa \cdot \rho}{\sigma - \rho}$
we get $u \leq U$.

For $d \leq K_\kappa$ and $u > U : \alpha(u + d) - \beta(u) < 0$, the requested bounds are $U + K_\kappa \leq K_\alpha$ and $U \leq K_\beta$. Beyond this compacted domain, the (min,+)-deconvolution yields negative values. Hence, the function value of $\kappa(d)$ for any $d \in [0, K_\kappa]$ takes its source in the compact domains of the input curves $[0, K_\alpha]$ and $[0, K_\beta]$.

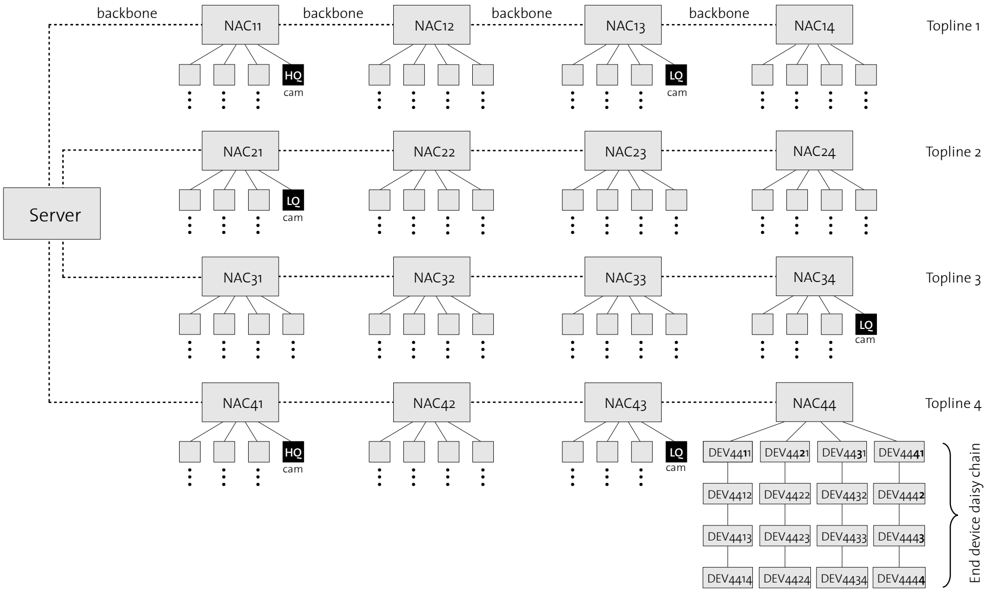


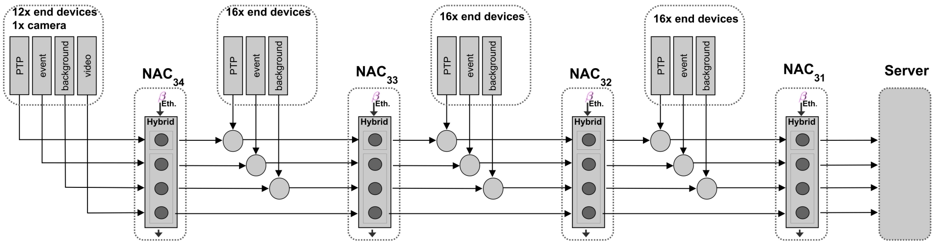
Figure 7.5: Topology of the studied distributed heterogeneous communication system.

7.3.3 Accuracy and Effort Evaluation

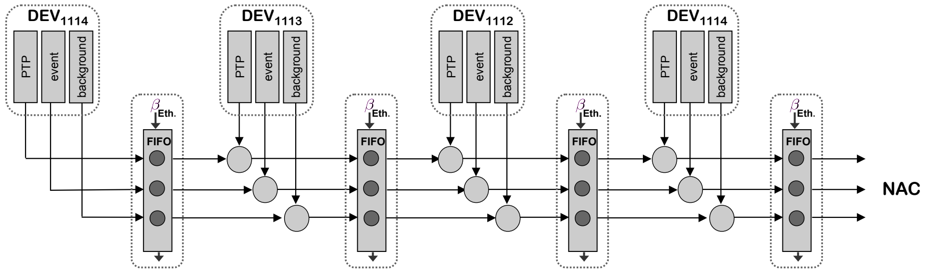
We illustrate the impact of NC on compact domains with an avionics case study. We analyze a so-called distributed heterogeneous communication system (HCS) by EADS Innovation Works[14], an in-cabin network architecture alternative to AFDX. A detailed description of this network as well as a RTC model that we use for algNC analysis can be found in [81]. Figure 7.5 depicts its topology and Figure 7.6 provides the details

---

[14] Now Airbus Group Innovations.



(a) Backbone model shown for topline 3.



(b) Model of an end device daisy-chain of topline 1.

Figure 7.6: Case study network model

of its NC model: two abstraction levels, the topline and the device daisy-chain. The network consists of one central server, 16 network access controllers (NAC), 232 end-devices (DEV) such as speakers, buttons, etc. as well as 6 video cameras for either high quality (HQ) or low quality (LQ) video streaming. Among the network traffic, time synchronization messages of the IEEE Precision Time Protocol (PTP) has the highest real-time demands as it, e.g., synchronizes local times at speakers which is crucial to prevent distorted playback of audio messages. In our case study, we will focus on the PTP traffic from the end-devices to the server. The network for this analysis is a sink tree, i.e., we can apply the sink-tree PMOOA of Section 4.3 and thus we can compact the curve domains.

For the experiments, we used the MPA-toolbox [58], adapted to work on compact domains according to the results from Section 7.3.2. The experiments were carried out with Matlab R2013a running on a 2.2GHz Intel Core $i7$ with 8GB DRAM and MacOS 10.7.5. Table 7.2 presents the run-times when using the MPA standard implementation of piece-wise linear (PWL), pseudo-periodic curves, overapproximated curves ($\mathcal{F}_{\mathrm{TB}}$, $\mathcal{F}_{\mathrm{RL}}$) and when using the compact domains for the analysis. The second column presents the rounded computation time for one analysis run. Column three reports the worst-case delay of a PTP message sent from an end-device of NAC 4 in topline 1 to the

Curve Model	Analysis Time	PTP Delay Bound
PWL, pseudo-periodic curves	282.33s	6.1250ms
Approximated curves	0.36s	17.2979ms
Compact domains	1.13s	6.1250ms

Table 7.2: PMOOA run-time and delay bounds for PTP messages sent to the server.

server. Table 7.2 illustrates that the use of compact domains drastically decreases the computation run-time, getting very close to the linear overapproximation, while retaining the delay bound accuracy of the precise modeling approach using PWL, pseudo-periodic curves.

8 The New State of the Art in Network Calculus

We conclude the presentation of contributions made in this thesis with a comparison of the previous state of the art in Network Calculus (NC) and our new algebraic NC (algNC) with optimization principles. The former is demarcated by the optimization-based NC analysis (optNC) applicable to feed-forward networks, i.e., the Unique Linear Programming Analysis (ULPA), and the algNC analysis from the same literature, the Separate Flow Analysis (SFA) that applies a segregate PBOO cross-traffic arrival bounding (segrPBOOAB) [10].

The eventual new state of the art contributed by this thesis is the Tandem Matching Analysis (TMA) for the derivation of the analyzed foi's delay bound, accompanied by the Tandem Matching Arrival Bounding (TMAB) with TFA-assistance. It belongs to the newly established category of compositional, algebraic analyses that implement optimization principles. With the burstiness reduction aspect added to the TMAB, we conclude the integration of optimization principles, i.e., extension of the search space for of algNC. It can improve intermediate arrival bounds derived by the TMAB as the matching of multiple sub-tandems onto another tandem introduces the explicit derivation of burstiness – similar to the aggrPBOOAB. Executing this TFA assistance with the TMAB itself extends the search space of our analysis into the direction of larger cross-traffic aggregates. This method counteracts the problem all compositional algNC analyses and the ULPA share: Violation of the newly established PSOO principle. Therefore, the presented delay bound benchmarks evaluate the accuracy gap between global optimization applied upon the NC network description and the composition of tandem-local, algNC analyses – the so-called composition penalty (Section 3.2). We express this gap by the improvement factor [12]: a value of 1.0 denotes parity, a value in $(0, 1)$ indicates a worsened network delay bound and values > 1.0 measure the improvement.

In this thesis, we also shifted the attention to the computational effort of NC, to be precise, to the user-facing metric of experienced analysis computation time. We saw that this is the very reason to depart from the only analysis that can theoretically derive tight bounds for the NC network of a feed-forward network[15]. In order to provide a self-contained comparison, let us briefly repeat the common analysis setting assumed to obtain the following results. They are especially crucial for the presented analysis run-times.

For topology generation, we employed existing tool support. The result is a device graph that we load into the Disco Deterministic Network Calculator (DiscoDNC, [7]) for further processing: First, the device graph is transformed into a server graph. Next, we

---

[15] If strict service curves are decomposable into the maximum of a set of rate latencies and arrival curves can be decomposed into the minimum of token buckets

applied turn prohibition [34, 79] to enforce the feed-forward property by removing links from the server graph. The device graph and its corresponding turn-prohibited server graph are used to route flows: We randomly choose source and sink devices in the device graph and then route the flow defined by it via the shortest path in the server graph. Note, that the server graph's feed-forward property prevents cyclic dependencies between flows and that none of the sink device's servers is crossed. These steps to create a network for analysis, i.e., a server graph with flows, are common to all analyses. Therefore, we do not measure the time they take in our reported computation times. These times comprise the following parts that the respective analyses take to calculate results:

- For algNC analyses, we measure the entire time to execute all steps of the compositional feed-forward analysis with the DiscoDNC – our integrated cross-traffic arrival bounding that backtracks dependencies through the network, transforms them into an algebraic equation and solves it. Then, the cross-traffic arrival bounds are used to derive the foi's delay bound.
- For the optimization-based ULPA, we use the DiscoDNC for the backtracking of dependencies and the conversion into a linear program, formatted to be either solved with the open-source LpSolve or IBM CPLEX.

Our experiments were carried out on identical servers, each equipped with an Intel Xeon E5420 CPU with four physical cores and 12GB of RAM, running Ubuntu 14.04 LTS. The DiscoDNC is a Java-based library that requires a Java Runtime Environment. We employed the DiscoDNC version 2.2.3 and Oracle Java SE Runtime Environment 8. Further, we used LpSolve version 5.5.2.0 and IBM CPLEX version 12.6.2. Note, that the DiscoDNC was programmed to run single-threaded only. LpSolve exhibits the same behavior whereas CPLEX can be configured to run with up to a specified number of threads. This does, however, introduce thread synchronization overhead. In multi-threaded mode, CPLEX offers two different types of thread synchronization. As we are concerned with the deterministic analysis of potentially safety-critical real-time systems, we chose the deterministic parallel search mode over the opportunistic one.

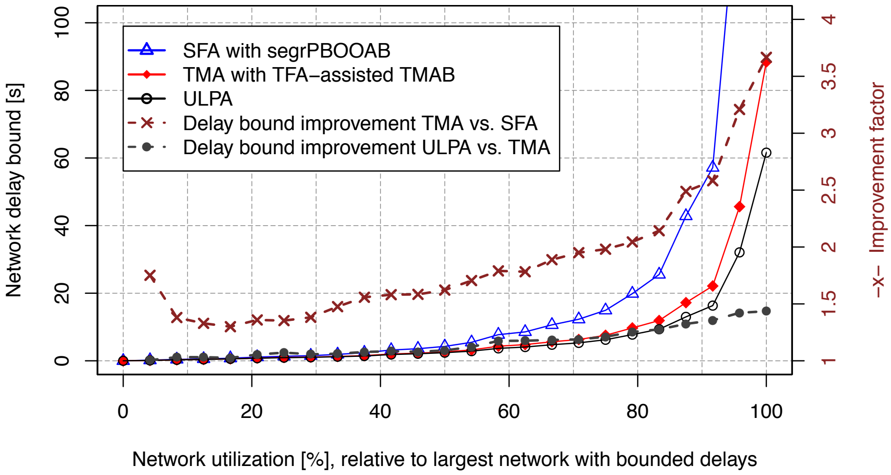
In this setting, we obtained the following benchmark results between the previous state of the art in NC and the new state of the art contributed by this thesis.

8.1 Numerical Experiments I: Erdős-Rényi Networks

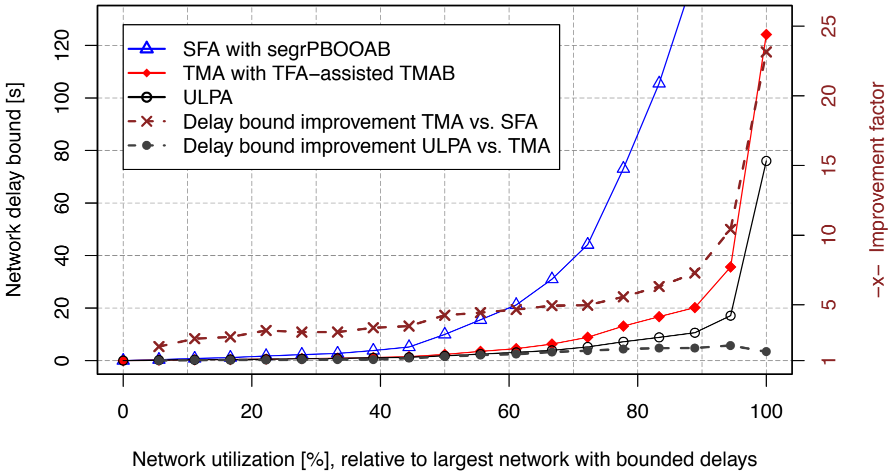
In our first numerical experiments, we randomly create Erdős-Rényi device graphs following the $G(n,p)$-model with $p = 0.1$. We evaluate the impact of improved arrival bounding in flat and hierarchical[16] network topologies with $n = 32$ devices, resulting in

---

[16] Hierarchy retrofitted by the aSHIIP tool.



(a) Flat network.



(b) Hierarchical network.

Figure 8.1: Delay bounds and improvement factors in Erdős-Rényi networks of size 32.

114 servers (flat) as well as 73 servers (hierarchical). The amount of flows is continuously increased in steps of 50. With this approach, bottlenecks build up randomly in the flat network whereas the hierarchical network possesses a set of predefined bottleneck links – those links connecting the levels of the hierarchy. Each flow has an unit sized arrival curve $\alpha = \gamma_{1,1}$. The network capacity in our experiments were 1200 flows (flat) and 900 flows (hierarchical). A further increase of 50 flows resulted in flows whose delay could not be bounded. We evaluate the network delay bound, i.e., the maximum delay bound among all analyzed flows. Each flow's delay is bounded with the three analyses mentioned above: SFA with segrPBOOAB, TMA with TFA-assisted TMAB and ULPA. Delay bounds are decreasing in this order. Therefore, we defined two improvement factors. The first one

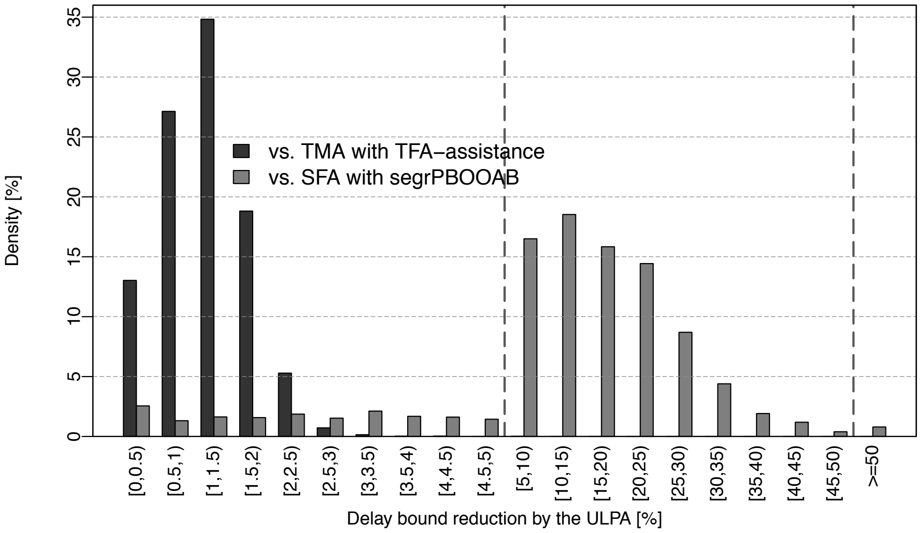
illustrates the contribution made in this thesis, relative to the previous algebraic analysis. It is defined by $\frac{D(\text{SFA with segrPBOOAB})}{D(\text{TMA with TFA-assisted TMAB})}$. The second improvement factor is defined from the ULPA's point of view and compares its delay bounds with the TMA delay bounds, i.e., we derive $\frac{D(\text{TMA with TFA-assistance})}{D(\text{ULPA})}$.

Figure 8.1 depicts the results of numerical experiments with Erdős-Rényi networks. All delay bounds show an asymptotic growth when the network's capacity is approached. However, the SFA with segrPBOOAB is considerably outperformed by the TMA and the ULPA. In the flat ER network of Figure 8.1a, we improved algNC delay bounds with the TMA by a factor of at least 1.3 and at most 3.6. In contrast, the improvement from TMA to ULPA always resided between 1.026 and 1.435, i.e., the TMA brought algNC delay bounds close to those of optNC. In the hierarchical ER network, results are similar, yet, more pronounced due to the set of predefined bottleneck links. In this network (Figure 8.1b), the improvement factor that TMA achieves over SFA is always above 2.0, it even exceeds 5.0 for many utilizations and eventually reaches a factor of 23. Departing from algNC and applying the ULPA of optNC results in a nearly inverted situation. Compared to the TMA, improvements of a factor larger than 2.0 can only be observed once (factor 2.084). For most of the utilizations we evaluated, the improvement factor even stays below 1.5. These observations reveal that the TMA we contribute in this thesis can derive delay bounds within a small deviation from the ULPA bounds. We will further investigate this aspect in the following.

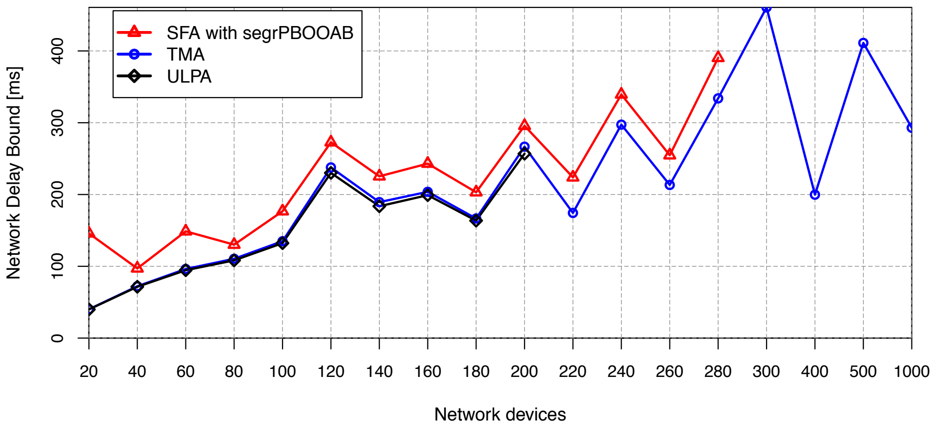
8.2 Numerical Experiments II: General Linear Preference Networks

For our second set of numerical experiments, we decided to investigate large networks and depart from the previous ER topology generation. Instead, we created Internet-like topologies the with the aSHIIP tool, according to the general linear preference (GLP) model [22]. We used the default GLP parameter setting $m_0 = 20$, $m = 1$, $p = 0.4695$, $\beta_{\text{GLP}} = 0.6447$ to generate these device graphs, independently for each network size. Traffic was created with a fixed server-to-flow ratio of 1:4 to generate load in all networks. As before, flows are routed between randomly selected devices and their arrival curves are uniformly shaped to token buckets with rate 5Mbps and bucket size 5Mb. Table 8.2c shows the amount of devices, servers and flows for all of the GLP networks we evaluate in this thesis. Service curves resemble full-duplex links with a transmission rate of 10Gbps.

Delay Bound Accuracy Assessment of delay bound accuracy is performed with respect to the most accurate results that are computationally feasible to derive with NC. We analyze networks of sizes 20 to 180 with the ULPA, i.e., our evaluation is based on the analysis of 12376 flows across 9 independently created GLP networks of different sizes. For these flows, Figure 8.2a illustrates the reduction of delay bounds achieved by



(a) Flow Delay Bound Deviation from the ULPA.



(b) Network Delay Bounds.

Devices	Servers	Flows
20	38	152
40	118	472
60	164	656
80	282	1128
100	364	1456
120	398	1592

Devices	Servers	Flows
140	512	2048
160	572	2288
180	646	2584
200	740	2960
220	744	2976
240	882	3528

Devices	Servers	Flows
260	976	3904
280	994	3976
300	1124	4496
400	1478	5912
500	1876	7504
1000	3626	14504

(c) Network sizes.

Figure 8.2: Delay Bounds in Internet-like GLP networks.

the ULPA over the two algebraic analyses, SFA with segrPBOOAB and the TMA with TFA-assistance. This measure is similar to the improvement factor, yet, it is depicted on a percentage scale instead of $(0, \infty)$. We show the density of reduction to provide an unified depiction of the relative delay bound performance across different networks.

The SFA variant of [10] can be considerably improved by the optimization-based ULPA. We observe, that only 17.32% of delay bounds are close to the ULPA ones, that is able to reduce them by at most 5%. The mean reduction is at 15.63% but more than 8.68% of delay bounds can even be improved by at least 30%. The maximum is as large as 73%. This high variation is caused by the different network sizes as well as the randomness in their creation. Flow entanglements and therefore the backtracking of dependencies can become a significant factor for the SFA, especially in this variant that was proposed in the literature as it enforces cross-flow segregation during arrival bounding. These insights also explain the inferior performance of the SFA with respect to the network delay bound. The most important observations in Figure 8.2b are:

- The absolute network delay bounds are smaller than in Figure 8.1, i.e., the network is operating at a low network utilizations, especially at large sizes.
- Between 20 and 100 devices, the network delay pattern of the SFA and the ULPA crucially differ. Whereas the ULPA shows a nearly linear growth with the network size, the SFA oscillates. It would rank these five networks differently than the ULPA. The cause for this behavior is the higher network utilization in the small networks.
- With smaller utilizations (networks $\geq$ 120 devices), the SFA can rank networks equal to the ULPA but its delay bounds remain worse. It may be required to analyze the networks ranked best by SFA with a more accurate analysis in order to validate against strict deadlines.

The TMA we contribute in this thesis performs well compared to the ULPA. It is depicted mostly on the left side of Figure 8.2a where we adapted the step size accordingly. Except for a single outlier at 7.57%, none of the delay bounds could have been improved by more than 4.2% by the ULPA. In fact, the mean is at 1.142%, the median is at 1.146% and the 99th quantile at 2.48%. Overall, the TMA shows that, in contrast to previous belief, the algNC branch is able to derive competitively accurate delay bounds. This conclusion is confirmed by the evaluation of network delay bounds depicted in Figure 8.2b. The TMA results are a close match to the ULPA ones. Neither the second nor the third observations we made concerning the SFA is valid for the TMA. It is not significantly impacted by network size or utilization and thus ranks the different GLP networks equal to the ULPA – a result that agrees with the flow delay evaluation in ER networks of

Section 6.2.3. They show that, for individual flows, the TMA derives delay bounds that are within a bounded deviation from the ULPA.

In our numerical accuracy evaluations for different network types, sizes and utilizations, we demonstrated that our new algNC analysis, the TMA with TFA-assisted TMAB, derives delay bounds that are highly competitive with those of the ULPA, the representative of NC's optimization branch. Next, we turn to the computational effort required to execute these alternatives.

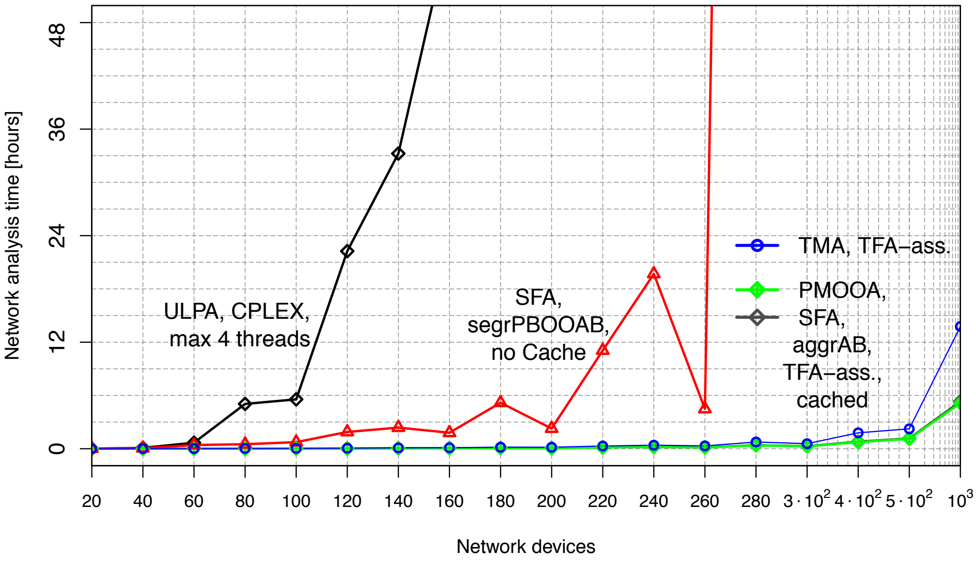
Computational Effort The computational effort required to execute the NC analyses that define the state of the art in the literature and the newly proposed analysis are depicted in Figure 8.3. Ranking the three major alternatives with respect to this metric yields a very different result than the accuracy evaluation. The ULPA represents the optimization branch of NC as it is the most efficient analysis of it. However, relative to the algNC analyses, it does not perform competitively. At a network size of 180 devices, the ULPA already took $\sim$ 13 days. The larger the network, the more effort is required to trace the influence of flows on each other, yet, remember that the fixed server to flow ratio of 1:4 seemingly resulted in small utilizations in larger networks (the SFA's network delay bound did not oscillate compared to the other analyses). With higher utilizations and thus a more flows in networks exceeding 100 devices, tracking their entanglement and optimizing the assumed impact on each other becomes more complex, rendering the ULPA computationally infeasible earlier. Therefore, we rely on this small server to flow ratio in order to provide computation run-times for our investigation.

The SFA with segrPBOOAB performs better, yet, it also reaches its limit soon after the ULPA. Figure 8.3 shows that the SFA becomes computationally infeasible at 280 network devices. Analyzing the 3976 flows in this network with 994 servers takes more than 14 days[17]. The major reason for this behavior is its worst-case segregation of cross-flows. Massive computational effort is required to bound all cross-flows' mutual interference. Moreover, the SFA network analysis times oscillate heavily, again. The two independently created networks with 200 and 220 devices do not differ much in terms of their respective server graphs' size; the latter is just 4 servers and 16 flows larger. Analyzing it with the SFA takes $\sim$10.5 hours, i.e., $4\frac{2}{3}$ as much time. Significantly faster analysis of larger networks is also possible: See the 260 devices network that is analyzed in $\sim$4.5 hours albeit consisting of 30% more flows.

Our TMA does not suffer from these problems thanks to the conceptual improvement

---

[17] Note, that we did not facilitate the early termination of backtracking of cross-flows by employing arrival bound caching. In [4], results of the cached version of this SFA variant can be found. They are nearly identical to the shown results. Segregate arrival bounding does not seem to benefit from caching. With it, the analysis of the 280 devices network still takes $\sim$13 days instead of $\sim$14 days.



(a)

Devices	ULPA, CPLEX (max 4 threads)	SFA, segrPBOOAB (1 thread)	TMA, TFA-ass. (1 thread)
20	00:00:13	00:01:16	00:00:13
40	00:06:04	00:03:39	00:00:16
60	00:40:41	00:25:01	00:00:48
80	05:03:00	00:30:42	00:01:34
100	05:33:44	00:44:36	00:02:31
120	22:15:22	01:53:31	00:03:41
140	33:14:00	02:22:59	00:05:58
160	58:20:54	01:48:05	00:07:05
180	∼13 days	05:11:18	00:10:27
200	–	02:16:38	00:10:13
220	–	11:04:04	00:18:23
240	–	19:41:46	00:23:59
260	–	04:28:52	00:19:22
280	–	∼14 days	00:45:32
300	–	–	00:34:10
400	–	–	01:47:39
500	–	–	02:14:31
1000	–	–	13:45:52

(b)

Figure 8.3: Computational Effort in Internet-like GLP networks.

of this thesis: aggregate bounding of cross-traffic as well as the convolution and caching of its intermediate arrival bounds. Implementing them, we achieve more resilience to changing factors of the analysis, e.g., where the SFA effort increased by a factor $4\frac{2}{3}$, the TMA only takes 1.8 times as long. The analysis scales much better than the others, such that it is possible to accurately analyze networks that could not be analyzed with NC before. For example, the largest network of our evaluation has 1000 devices. Bounding the delay of all its 14504 flows with the TMA was faster than the ULPA analysis of the network with 120 devices or the SFA in 240 or 280 devices networks.

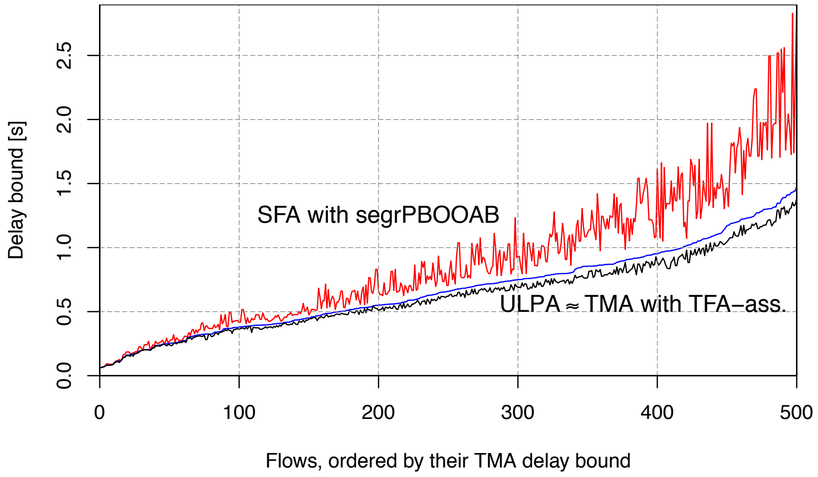
Last, we also provide the impact of the efficiency improvements on other algNC analyses. Namely, the SFA and the PMOOA with aggrAB. Their arrival bounding implements a different optimization principle that results in a smaller search space than the TMA, i.e., they potentially derive less accurate delay bounds. However, the smaller search space also allows them to derive delay bounds faster than the TMA. Growing the network size even larger, a tradeoff might be necessary to obtain delay bounds at all; the TMA effort grows to $13\frac{3}{4}$ hours in the 1000 devices network and sacrificing the TFA-assistance only decreases this time to $11\frac{1}{4}$ hours (see Table 7.1). The results shown on the far right of Figure 8.3a suggest that compromising accuracy for less computational effort should be done incrementally:

1. From TMA to SFA and PMOOA (they have nearly identical analysis times),
2. from SFA and PMOOA to only executing the PMOOA with aggrAB that is usually more accurate than SFA and then
3. finally to a less sophisticated cross-traffic arrival bounding like aggrPBOOAB, but not to segrPBOOAB.

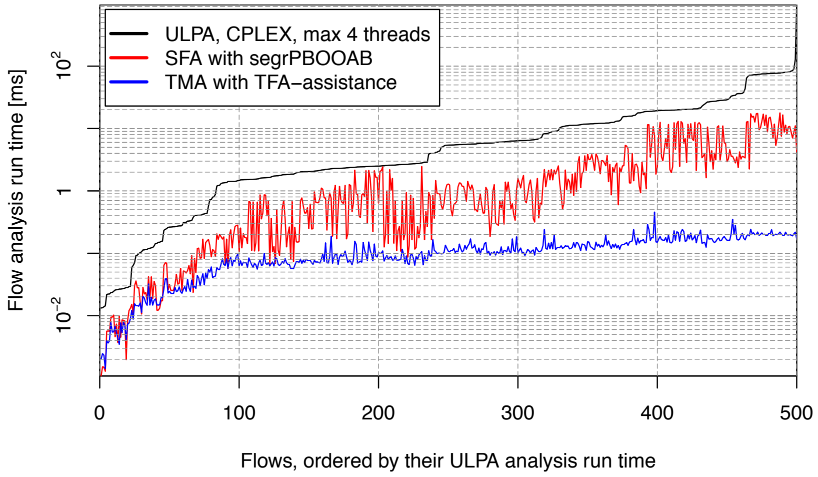
The exact run-time of a NC analysis depends on many factors and predicting it is still an open research topic. However, relative comparisons reveal significant differences; between optNC and algNC and among the algNC analyses. Analyzing large feed-forward networks in practice remains solely possible with algebraic, compositional NC.

8.3 Case Study: Avionics Full-Duplex Ethernet

We conclude with a final evaluation of an AFDX topology. It is dimensioned similar to backbone network in the Airbus A380 aircraft. Device graph creation followed the algorithm presented in [20] in order to result in a representative AFDX topology. It has a dense core of 16 switches that connect 125 end-systems. All links resemble full-duplex Ethernet connections with a speed of 100Mbps. For the transmission of data, the current



(a) Flow Delay Bounds.



(b) Computational Effort.

Figure 8.4: Case Study Results for an AFDX Topology.

AFDX specification defines so-called virtual links (VLs) that connect a single source end-system to multiple sink end-systems (in the device graph). They reserve a fixed amount of resources on the path between these systems that is, in turn, used to serve flows. For NC, VLs correspond to multicast flows that reserve large resource shares. This coarse granularity introduces problems to the network [55]. Moreover, NC does not provide a specialized analysis for multicast flows that implements the PBOO principle or even the PMOO principle – both are essential for accurate performance bounds. Therefore, we skip the VL abstraction layer in our evaluation. Consistently with our previous evaluations, we focus on flow delays in order to evaluate the impact of our findings on the immutable AFDX topology. We added 500 flows with arrival curve $\alpha = \gamma_{1,1}$ to this topology for a flow delay evaluation.

Figure 8.4a depicts the individual flow delays. Our observations from the ER and GLP networks also apply to the AFDX topology. Delay bounds of the TMA with TFA-assistance are in very close range of the optimization-based results of the ULPA. The SFA with segrPBOOAB, on the other hand, performs worse. Its delay bounds show a gap to the other analyses that tends to grow on average. Additionally, the SFA delay bounds oscillate compared to the ULPA and TMA bounds, such that this analysis is not suitable to confidently rank AFDX design alternatives regarding their network delay bounds. The SFA would also rank flows vastly different regarding their individual delay bound, i.e., could result in a falsely assumed delay bound violation for some flows.

From the perspective of the computed delay bounds, TMA and ULPA both seem suited for a design space exploration. We next turn to the effort aspect of analysis effort. Figure 8.4b shows the time it takes to analyze each flow in the network. This per-flow effort can differ more than four orders of magnitude within the SFA and within the ULPA. For the same set of analyses, the TMA per-flow computation times only differ two orders of magnitude. This fact may seem surprising at first as the AFDX topology has such a small network diameter. However, the reason can be found in our efficiency improvements: The backtrackings of SFA and ULPA terminate at the sources of flows whereas the TMA benefits from caching. Therefore, the TMA's effort stays within a much smaller range. In absolute terms, the TMA outperforms ULPA and SFA run times by multiple orders of magnitude – an advantage that can become decisive in large design space explorations.

9 Conclusion

In Section 8, we benchmarked to the previous state of the art in NC and the new one defined by the contributions of this thesis to provide a conclusive evaluation. In this section, we provide some concluding remarks on the contributions and an outlook to future research directions.

9.1 Concluding Remarks

This thesis contributes a fast and accurate NC solution for the derivation of end-to-end delay bounds in general feed-forward networks. Moreover, it can be easily employed to derive the backlog bound of servers. We followed a compositional approach which turned out key for the computational efficiency and, in contrast to previous belief, did not lead to noncompetitive results.

OptNC's source for tightness that has not been exploited by algNC before seems to be its attempt to attain a global view on the network. The optNC analysis that makes use of the maximum amount of global knowledge to derive its constraints, the LPA, was already shown to be NP-hard with no algorithm known to solve the underlying problem efficiently. Reducing the amount of constraints leads to the more feasible, yet not tight, approach of the ULPA. We showed that the ULPA relies on worst-case assumptions that are similar in impact to those of algNC. Moreover, we showed that the ULPA is nonetheless still more computationally demanding than the analyses of algNC. Therefore, this thesis takes one step back in order to take two steps forward. AlgNC follows a divide-and-conquer scheme that computes global performance bounds in a feed-forward network by composing tandem-local results. Whereas the "conquer"-part has been investigated in the literature, leading to different tandem analysis principles as well as analyses implementing them, the "divide"-aspect has not seen much treatment. By shifting our focus to this part of a compositional feed-forward analysis, we could show that considerable accuracy improvements are indeed possible. The Tandem Matching Analysis (TMA) contributed in this thesis is greatly influenced by the insights gained from optNC. With a more comprehensive decomposition of the feed-forward network into individual tandems to analyze, we attain a vast increase of the search space for the analysis – resulting in a significant gain in delay bound accuracy. The delay bounds we derive are very competitive to those obtained with ULPA.

The second step forward we take is w.r.t. computational efficiency of NC. We provide evidence that optNC's ULPA is computationally infeasible even for moderate network sizes. In contrast, the TMA can be improved such that it scales to network sizes previously out of reach for a NC analysis – a contribution we demonstrate with the open-source tool that was developed during work on this thesis, the DiscoDNC [7]. Building on the fact that all tandem matchings of the TMA deliver valid bounds, we can actually scale algNC to much larger networks than presented here by trading off the TMA's accuracy.

Although we restricted our presentation to arbitrary multiplexing analysis, the framework defined by the TMA can be combined with other NC tandem analyses. For instance, the algebraic FIFO tandem analyses of [53, 3] and the optimization-based ones of [74] seem suitable. Integrated into the TMA procedure, they can benefit from improved accuracy and the computational effort reduction that we contribute in this thesis.

9.2 Future Research Directions

One direction of future work is the investigation of real hybrid analyses. Whereas the TMA incorporates optimization principles, both branches of NC can be interfaced as well. For instance, algebraic cross-traffic arrival bounding such as TMAB can virtually shrink the network to be analyzed by the (U)LPA. The opposite direction is not yet possible as neither the LPA nor the ULPA derives an output bound. This prevents the integration of these analyses into the TMA framework. In contrast, the seminal optimization analysis of [74] derives a left-over service curve and thus can bound all three performance characteristics: Delay, backlog and output. It can be embedded into the TMA framework to replace the PBOO and the PMOO procedures on each tandem. I.e., we can replace the two competing analyses that constitute the potential combinatorial explosion with a single one. While this enables for better evaluation of the composition penalty in theory, the optimization allowing for it was shown to suffer from computational infeasibility as well [45, 46]. Nonetheless, in practice it could be started alongside the algebraic tandem analyses, yet, equipped with a time-out mechanism that enforces termination soon after the algNC analyses computed their results on the respective tandem. This scheme requires the employed tool to spread the computation for a specific tandem over multiple threads; tool support is another direction of future research.

Besides the multi-threaded compositional analysis, a promising research direction for tool support is the applied caching strategy. In this thesis, we only cache arrival bounds. For the computation of a left-over service curve, all the required arrival bounds need to be identified, requested from the cache, aggregated and subtracted from the original service curve. Extending the cache to also hold left-over service curves circumvents this effort, yet, it also increases the memory demand – a potential bottleneck not affecting the TMA paired with the current caching scheme.

Last, let us depict a potential research direction for optNC. The LPA is computationally infeasible and thus it cannot be used to exactly evaluate how close the accurate analyses, either algNC or optNC, are to the tight delay bound that can be computed from a given NC network model. What can be done instead, is deriving a lower bound on the delay in order to demarcate the region where the tight bound must reside – similar to the lower bounds used for compact domain computation. A narrow region simply means that the derived upper delay bound can be assumed close to the tight one, i.e., accurate in absolute terms. Remember, that the LPA approaches the tight delay bound from the region of invalid results – each of the individual LPs it derives computes a lower bound on the delay. Extension of the partial order of backlogged periods to a single compatible total order should thus, in principle, allow for the derivation of an optimization-based lower bound on the delay. However, a total order consists of more relations than the partial order, i.e., the derived linear program will possess more constraints and will therefore most probably require more computational effort to solve. A combination of simulation, best-case or average-case assumptions and optimization might, nonetheless, lead to a feasible approach to lower bounds on the delay. Although this thesis provides an algNC analysis that is fast and accurate, optNC can still play a role in the future of Network Calculus.

Nomenclature

α	Arrival curve
α'	Output arrival curve
α^f, $\alpha^{\mathbb{F}}$	Arrival curve of flow f, flow aggregate $\mathbb{F}$
α_s^f, $\alpha_s^{\mathbb{F}}$	Arrival bound of flow f, flow aggregate $\mathbb{F}$ at server s
α_s	Abbreviation for $\alpha_s^{F(s)}$
β	Service curve
$\beta_s^{\mathrm{l.o.}f}$, $\beta_s^{\mathrm{l.o.}\mathbb{F}}$	Left-over service curves for flow f, flow aggregate $\mathbb{F}$ at server s
$\beta_{R,T}$	Rate-latency curve with rate R and latency T
β_s, $\beta_{\mathcal{T}}$	Service curve of server s, tandem $\mathcal{T}$
$\gamma_{r,b}$	Token-bucket curve with rate r and bucket size b
$\langle s_x, \ldots, s_y \rangle$	Tandem of consecutive servers s_x to s_y
$\mathbb{F}$, $\mathbb{G}$	Flow aggregates, denoted by blackboard bold letters
$\mathcal{F}_0$	Set of non-negative, wide-sense increasing functions that pass through the origin
$\mathcal{F}_{\mathrm{mRL}}$	Set of multi-rate-latency curves, a subset of $\mathcal{F}_0$
$\mathcal{F}_{\mathrm{mTB}}$	Set of multi-token-bucket curves, a subset of $\mathcal{F}_0$
$\mathcal{F}_{\mathrm{RL}}$	Set of rate-latency curves, a subset of $\mathcal{F}_{\mathrm{mRL}}$
$\mathcal{F}_{\mathrm{TB}}$	Set of token-bucket curves, a subset of $\mathcal{F}_{\mathrm{mTB}}$
$\mathcal{T}$	Tandem of servers
$\ominus$	Non-decreasing upper closure, a binary operator
$\oslash$	(min,+)-deconvolution, a binary operator, also used for output bounding
$\otimes$	(min,+)-convolution, a binary operator
$\overline{\alpha}$	Upper bound on arrival curve α
$\overline{\beta}$	Upper bound on service curve β
$\overline{\otimes}$	(max,+)-convolution, a binary operator
$\overline{B}$	Upper bound on the backlog bound
$\overline{D}$	Upper bound on the delay bound
$\underline{\alpha}$	Lower bound on arrival curve α
$\underline{\beta}$	Lower bound on service curve β
$\underline{B}$	Lower bound on the backlog bound
$\underline{D}$	Lower bound on the delay bound
B	Backlog bound
D	Delay bound
$F(s)$	Set of flows at server s
$F_{\mathrm{src}}(s)$	Set of flows entering the network at server s
$L(f,s)$	Location (index) of server s on f's path $P(f)$

$P(f, i)$	Server at location (index) i on f's path $P(f)$
$up(s)$	Set of servers 1 hop upstream of s
$x(f)$, $x(\mathbb{F})$	Cross-traffic of flow f, flow aggregate $\mathbb{F}$
AFDX	Avionics Full-Duplex Switched Ethernet
aggrAB	Aggregate arrival bounding
aggrPBOOAB	Aggregate PBOO arrival bounding
aggrPMOOAB	Aggregate PMOO arrival bounding
algNC	Algebraic Network Calculus
CLPEX	A commercial linear program solver developed by IBM
compFFA	Compositional Feed-Forward Analysis
DiscoDNC	Disco Deterministic Network Calculator
EADS	European Aeronautic Defence and Space company, now Airbus Group
ER	Erdős-Rényi graph
FIFO	First In, First Out multiplexing
foi	The flow of interest analyzed by the analysis
GLP	General Linear Preference topology generation
HCS	Heterogeneous Communication System
LP	Linear Program
LPA	Linear Programming Analysis
LpSolve	An open-source linear program solver
LUDB	Least Upper Delay Bound
NC	Network Calculus
Network	A graph connecting servers and routing flows
optNC	Optimization-based Network Calculus
PBOO	Pay Bursts Only Once principle
PMOO	Pay Multiplexing Only Once principle
PMOOA	Pay Multiplexing Only Once Analysis
PSOO	Pay Segregation Only Once principle
PWL	Piecewise linear
segrPBOOAB	Segregated PBOO arrival bounding
SFA	Separate Flow Analysis
SP	Strict priority multiplexing
TDMA	Time Division Multiple Access
TFA	Total Flow Analysis
TM	Tandem Matching
TMA	Tandem Matching Analysis
TMAB	Tandem Matching Arrival Bounding
ULPA	Unique Linear Programming Analysis

List of Figures

2.1 NC network model: Device graph and server graph 10
2.2 Sample tree network 15
3.1 Sample network for compFFA illustration 24
3.2 PSOO violation enforced by compFFA 26
3.3 PSOO violation enforced by SFA and PMOOA with aggrPBOOAB 27
3.4 PSOO violation enforced by SFA with segrPBOOAB 30
3.5 Minimal network illustrating different analysis proceedings of segrPBOOAB and aggrPBOOAB 38
3.6 Network delay bound comparison of SFA with segrPBOOAB or aggrPBOOAB in ER networks 42
3.7 Sample network for beneficial TFA-assisted aggrPBOOAB 47
3.8 Comparison of server backlog and flow burstiness under increasing utilization, absolute difference 48
3.9 Comparison of server backlog and flow burstiness under increasing utilization, relative difference 50
3.10 Flow delay bound comparison of SFA with plain aggrPBOOAB and SFA with aggrPBOOAB with cross-traffic burstiness improvement, tailored network 52
3.11 Network delay bound comparison of SFA with segrPBOOAB, aggrPBOOAB or TFA-assisted aggrPBOOAB in ER networks 54
4.1 SensorNC network model 58
4.2 Cross-traffic arrival bounding in SensorNC 59
4.3 New cross-traffic arrival bounding in SensorNC 64
4.4 Flow delay bound comparison of conventional and new aggrPBOOAB in SensorNC 68
4.5 Computation time comparison of conventional and new aggrPBOOAB in SensorNC, relative improvements 69
4.6 Computation time comparison of conventional and new aggrPBOOAB in SensorNC, absolute improvements 70
4.7 SensorNC task admission control, conventional aggrPBOOAB 73
5.1 Tandem network with overlapping interference 79
5.2 Delay bounds in the tandem with overlapping interference 81
5.3 Delay bounds in the square network 84
5.4 Delay bound reduction, ULPA vs. SFA and PMOOA with aggrPBOOAB in GLP networks 86

5.5 Computation time comparison of ULPA, PMOOA and SFA with aggrPBOOAB in GLP networks . . . 87
6.1 PSOO violation enforced SFA with segrPBOOAB with aggrPMOOAB . . 90
6.2 Delay bound scaling of different analyses and cross-traffic arrival bounding alternatives . . . 92
6.3 Network delay bound comparison of SFA with TFA-assisted aggrPBOOAB and the exhaustive bounding procedure in ER networks . . . 95
6.4 Network delay bound comparison of TFA-assisted aggrPBOOAB and exhaustive delay bounding in an AFDX topology . . . 96
6.5 Number of left-over curves in sample ER networks . . . 97
6.6 Tandem matching alternatives in a sample network . . . 99
6.7 Delay bound scaling of different tandem matching alternatives . . . 100
6.8 Network delay bound comparison of SFA and TMA with TFA-assisted aggrPBOOAB in ER networks . . . 103
6.9 Flow delay bounds in the GLP network with 180 devices. . . . 104
6.10 Delay bound reduction, ULPA vs. TMA, SFA and PMOOA with aggrPBOOAB in GLP networks . . . 105
6.11 Sample search tree for the TMA . . . 106
6.12 Network delay bound comparison of TMA and the exhaustive bounding procedure in ER networks . . . 107
7.1 Computation time comparison of ULPA, SFA, PMOOA (aggrPBOOAB) and TMA in GLP networks . . . 113
7.2 Share of LP creation on the ULPA computation time in GLP networks . . 115
7.3 Sample tree network and its SFA and PMOOA . . . 118
7.4 Results for the sample tree network . . . 119
7.5 HCS network model . . . 125
7.6 HCS network model . . . 126
8.1 Network delay bound comparison of old and new state of the art, ER networks . . . 131
8.2 Flow and network delay bound comparison of old and new state of the art, GLP networks . . . 133
8.3 Computation time comparison of old and new state of the art, GLP networks 136
8.4 Flow delay bound and computation time comparison of old and new state of the art, AFDX topology . . . 138

List of Tables

2.1 NC notation scheme 12
4.1 SensorNC, additional notation 58
5.1 Amount of the total orders in trees 77
5.2 Size of GLP networks 85
7.1 Computation times of ULPA, SFA, PMOOA (aggrPBOOAB) and TMA in GLP networks 114
7.2 HCS evaluation results 127

References

[1] A. Biondi, G. Buttazzo, and S. Simoncelli. Feasibility analysis of engine control tasks under edf scheduling. In *Proc. Euromicro ECRTS*, pages 139–148, July 2015.

[2] L. Bisti, L. Lenzini, E. Mingozzi, and G. Stea. DEBORAH: A Tool for Worst-case Analysis of FIFO Tandems. In *Proc. ISoLA*, 2010.

[3] L. Bisti, L. Lenzini, E. Mingozzi, and G. Stea. Numerical Analysis of Worst-Case End-to-End Delay Bounds in FIFO Tandem Networks. *Springer Real-Time Systems Journal*, 48(5):527–569, 2012.

[4] S. Bondorf, P. Nikolaus, and J. B. Schmitt. Delay bounds in feed-forward networks – a fast and accurate network calculus solution. *arXiv:1603.02094 [cs.NI]*, Mar. 2016.

[5] S. Bondorf and J. Schmitt. The discodnc v2 – a comprehensive tool for deterministic network calculus. *EAI Endorsed Transactions on Internet of Things*, 15(4), dec 2015.

[6] S. Bondorf and J. B. Schmitt. Statistical Response Time Bounds in Randomly Deployed Wireless Sensor Networks. In *Proc. IEEE LCN*, 2010.

[7] S. Bondorf and J. B. Schmitt. The DiscoDNC v2 – A Comprehensive Tool for Deterministic Network Calculus. In *Proc. ValueTools*, 2014.

[8] S. Bondorf and J. B. Schmitt. Boosting Sensor Network Calculus by Thoroughly Bounding Cross-Traffic. In *Proc. IEEE INFOCOM*, 2015.

[9] S. Bondorf and J. B. Schmitt. Calculating Accurate End-to-End Delay Bounds – You Better Know Your Cross-Traffic. In *Proc. ValueTools*, 2015.

[10] A. Bouillard. *Algorithms and Efficiency of Network Calculus.* Habilitation thesis, ENS, 2014.

[11] A. Bouillard, L. Jouhet, and E. Thierry. Service curves in Network Calculus: dos and don'ts. Research Report RR-7094, INRIA, 2009.

[12] A. Bouillard, L. Jouhet, and E. Thierry. Tight Performance Bounds in the Worst-Case Analysis of Feed-Forward Networks. In *Proc. IEEE INFOCOM*, 2010.

[13] A. Bouillard and A. Junier. Worst-case Delay Bounds with Fixed Priorities Using Network Calculus. In *Proc. ValueTools*, 2011.

[14] A. Bouillard and G. Stea. Exact Worst-Case Delay for FIFO-Multiplexing Tandems. In *Proc. ValueTools*, 2012.

[15] A. Bouillard and G. Stea. Exact Worst-Case Delay in FIFO-Multiplexing Feed-Forward Networks. *IEEE/ACM Transactions on Networking*, 2014.

[16] A. Bouillard and G. Stea. *Worst-Case Analysis of Tandem Queueing Systems Using Network Calculus* in *Quantitative Assessments of Distributed Systems*, chapter 6, pages 129–173. Wiley, April 2015.

[17] A. Bouillard and E. Thierry. An Algorithmic Toolbox for Network Calculus. *JDEDS*, 2008.

[18] M. Boyer. NC-maude: a rewriting tool to play with network calculus. In *Proceedings of the 4th international conference on Leveraging applications of formal methods, verification, and validation - Volume Part I*, ISoLA'10, pages 137–151, Berlin, Heidelberg, 2010. Springer-Verlag.

[19] M. Boyer and C. Fraboul. Tightening end to end delay upper bound for afdx network calculus with rate latency fifo servers using network calculus. In *IEEE International Workshop on Factory Communication Systems (WFCS)*, pages 11–20, 2008.

[20] M. Boyer, N. Navet, and M. Fumey. Experimental Assessment of Timing Verification Techniques for AFDX. In *Proc. ERTS*, 2012.

[21] M. Boyer, N. Navet, X. Olive, and E. Thierry. The pegase project: precise and scalable temporal analysis for aerospace communication systems with network calculus. In *ISoLA*, ISoLA'10, pages 122–136, 2010.

[22] T. Bu and D. Towsley. On Distinguishing between Internet Power Law Topology Generators. In *Proc. IEEE INFOCOM*, 2002.

[23] C.-S. Chang. *Performance Guarantees in Communication Networks.* Springer, 2000.

[24] D. B. Chokshi and P. Bhaduri. Performance Analysis of FlexRay-based Systems Using Real-time Calculus, Revisited. In *Proc. ACM SAC*, 2010.

[25] R. L. Cruz. A Calculus for Network Delay, Part I: Network Elements in Isolation. *IEEE Transactions on Information Theory*, 1991.

[26] R. L. Cruz. A Calculus for Network Delay, Part II: Network Analysis. *IEEE Transactions on Information Theory*, 1991.

[27] R. Cummings, K. Richter, R. Ernst, J. Diemer, and A. Ghosal. Exploring Use of Ethernet for In-Vehicle Control Applications: AFDX, TTEthernet, EtherCAT, and AVB. In *SAE 2012 World Congress*, 2012.

[28] J. A. R. De Azua and M. Boyer. Complete modelling of avb in network calculus framework. In *Proc. of the 22nd International Real-Time Networks and Systems Conference (RTNS)*, 2014.

[29] J. Echagüe and V. Cholvi. Tight Arrival Curve at the Output of a Work-Conserving Blind Multiplexing Server. *Informatica*, 2010.

[30] M. Fidler. Extending the Network Calculus Pay Bursts Only Once Principle to Aggregate Scheduling. In *Proc. QoS-IP*, 2003.

[31] M. Fidler. Survey of deterministic and stochastic service curve models in the network calculus. *Communications Surveys Tutorials*, 2010.

[32] F. Frances, C. Fraboul, and J. Grieu. Using Network Calculus to Optimize AFDX Network. In *ERTS*, 2006.

[33] F. Geyer and G. Carle. Network engineering for real-time networks: comparison of automotive and aeronautic industries approaches. *IEEE Communications Magazine*, 54(2):106–112, February 2016.

[34] C. J. Glass and L. M. Ni. The turn model for adaptive routing. In *Proc. of the 19th ACM International Symposium on Computer Architecture (ISCE)*, ISCA '92, 1992.

[35] N. Gollan and J. B. Schmitt. Energy-Efficient TDMA Design Under Real-Time Constraints in Wireless Sensor Networks. In *Proc. IEEE/ACM MASCOTS*, pages 80–87, October 2007.

[36] J. Grieu. *Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques.* PhD thesis, INPT, 2004.

[37] N. Guan and Y. Wang. Finitary real-time calculus: Efficient performance analysis of distributed embedded systems. In *Proc. of IEEE RTSS*, 2013.

[38] T. Hamza, J.-L. Scharbarg, and C. Fraboul. Priority Assignment on an Avionics Switched Ethernet Network (QoS AFDX). In *Proc. IEEE WFCS*, 2014.

[39] P. Heise, N. Tobeck, O. Hanka, and S. Schneele. Safdx: deterministic high-availability ring for industrial low-cost networks. In *Proc. of 7th International Workshop on Communication Technologies for Vehicles (Nets4Cars-Fall)*, pages 40–44, Oct 2014.

[40] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *ACM SenSys*, pages 81–94, November 2004.

[41] B. Jonsson, S. Perathoner, L. Thiele, and W. Yi. Cyclic Dependencies in Modular Performance Analysis. In *Proc. ACM EMSOFT*, 2008.

[42] L. Jouhet. *Algorithmique du Network Calculus.* PhD thesis, École Normale Supérieure de Lyon, November 2012.

[43] P. Jurcik, A. Koubâa, R. Severino, M. Alves, and E. Tovar. Dimensioning and Worst-Case Analysis of Cluster-Tree Sensor Networks. *ACM Transactions on Sensor Networks*, pages 14:1–14:47, September 2010.

[44] S. Kerschbaum, K.-S. J. Hielscher, U. Klehmet, and R. German. A Framework for Establishing Performance Guarantees in Industrial Automation Networks. In *GI/ITG MMB & DFT*, 2014.

[45] A. Kiefer. *Computation of Tight Bounds in Networks of Arbitrary Schedulers.* Diploma (Master's) thesis, University of Kaiserslautern, March 2009.

[46] A. Kiefer, N. Gollan, and J. Schmitt. Searching for Tight Performance Bounds in Feed-Forward Networks. In *GI/ITG MMB and DFT*, 2010.

[47] H. Kim and J. Hou. Network Calculus Based Simulation for TCP Congestion Control: Theorems, Implementation and Evaluation. In *Proc. IEEE INFOCOM*, pages 2844–2855, March 2004.

[48] A. Koubâa, M. Alves, and E. Tovar. Modeling and Worst-Case Dimensioning of Cluster-Tree Wireless Sensor Networks. In *Proc. IEEE RTSS*, pages 412–421, December 2006.

[49] K. Lampka, S. Perathoner, and L. Thiele. Component-based system design: analytic real-time interfaces for state-based component implementations. *Software Tools for Technology Transfer (STTT)*, 2012.

[50] J.-Y. Le Boudec and P. Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet.* Springer, 2001.

[51] L. Lenzini, E. Mingozzi, and G. Stea. Delay bounds for FIFO aggregates: a case study. *Computer Communications*, 28(3):287 – 299, 2005.

[52] L. Lenzini, E. Mingozzi, and G. Stea. End-to-end Delay Bounds in FIFO-multiplexing Tandems. In *Proc. ValueTools*, 2007.

[53] L. Lenzini, E. Mingozzi, and G. Stea. A Methodology for Computing End-to-End Delay Bounds in FIFO-Multiplexing Tandems. *Elsevier Performance Evaluation*, 65(11–12):922–943, 2008.

[54] X. Li, J.-L. Scharbarg, and C. Fraboul. Improving End-to-End Delay Upper Bounds on an AFDX Network by Integrating Offsets in Worst-Case Analysis. In *Proc. IEEE Conference on Emerging Technologies Factory Automation (ETFA) 2009*, 2010.

[55] R. Mancuso, A. V. Louis, and M. Caccamo. Using Traffic Phase Shifting to Improve AFDX Link Utilization. In *Proc. ACM EMSOFT*, 2015.

[56] A. Maxiaguine, S. Kunzli, S. Chakraborty, and L. Thiele. Rate analysis for streaming applications with on-chip buffer constraints. In *Design Automation Conference, 2004. Proceedings of the ASP-DAC 2004. Asia and South Pacific*, pages 131–136, 2004.

[57] A. Mifdaoui and H. Ayed. Wopanets: A tool for worst case performance analysis of embedded networks. In *Computer Aided Modeling, Analysis and Design of Communication Links and Networks (CAMAD), 2010 15th IEEE International Workshop on*, pages 91–95, Dec 2010.

[58] Modular Performance Analysis Framework. www.mpa.ethz.ch.

[59] S. Perathoner, K. Lampka, and L. Thiele. Composing heterogeneous components for system-wide performance analysis. In *Proc. of Design, Automation and Test in Europe (DATE)*, 2011.

[60] W. Y. Poe, M. A. Beck, and J. B. Schmitt. Achieving High Lifetime and Low Delay in Very Large Sensor Networks using Mobile Sinks. In *Proc. IEEE DCOSS*, pages 17–24, May 2012.

[61] Y. Qian, Z. Lu, and W. Dou. Applying network calculus for worst-case delay bound analysis in on-chip networks. In *Proc. of 4th International Conference on Design Technology of Integrated Systems in Nanoscal Era (DTIS)*, pages 113–118, 2009.

[62] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian. A software defined networking architecture for the internet-of-things. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–9, May 2014.

[63] R. Queck. Analysis of ethernet avb for automotive networks using network calculus. In *Vehicular Electronics and Safety (ICVES), 2012 IEEE International Conference on*, pages 61–67, July 2012.

[64] G. Rizzo and J.-Y. Le Boudec. “Pay Bursts Only Once” does not hold for non-FIFO Guaranteed Rate Nodes. *Elsevier Performance Evaluation*, 62(1–4):366 – 381, 2005.

[65] U. Roedig, N. Gollan, and J. B. Schmitt. Validating the Sensor Network Calculus by Simulations. In *Proc. Performance Control in Wireless Sensor Networks Workshop at WICON*, October 2007.

[66] F. Ruskey. *Combinatorial Generation.* 2003.

[67] A. Saggio, G. Du, X. Zhao, and Z. Lu. Validating Delay Bounds in Networks on Chip: Tightness and Pitfalls. In *Proc. IEEE ISVLSI*, 2015.

[68] H. Schiøler, J. J. Jessen, J. D. Nielsen, and K. G. Larsen. Network Calculus for Real Time Analysis of Embedded Systems with Cyclic Task Dependencies. In *Computers and Their Applications*, 2005.

[69] M. Schmidt, S. Veith, M. Menth, and S. Kehrer. DelayLyzer: A Tool for Analyzing Delay Bounds in Industrial Ethernet Networks. In *GI/ITG MMB & DFT*, 2014.

[70] J. B. Schmitt, N. Gollan, S. Bondorf, and I. Martinovic. Pay Bursts Only Once Holds for (Some) non-FIFO Systems. In *Proc. IEEE INFOCOM*, 2011.

[71] J. B. Schmitt and U. Roedig. Sensor Network Calculus – A Framework for Worst Case Analysis. In *Proc. IEEE DCOSS*, 2005.

[72] J. B. Schmitt and U. Roedig. Worst Case Dimensioning of Wireless Sensor Networks under Uncertain Topologies. In *Proc. Workshop on Resource Allocation in Wireless Networks at IEEE WiOpt*, April 2005.

[73] J. B. Schmitt and F. A. Zdarsky. The DISCO Network Calculator – A Toolbox for Worst Case Analysis. In *Proc. ValueTools*, October 2006.

[74] J. B. Schmitt, F. A. Zdarsky, and M. Fidler. Delay Bounds under Arbitrary Multiplexing: When Network Calculus Leaves You in the Lurch ... In *Proc. IEEE INFOCOM*, 2008.

[75] J. B. Schmitt, F. A. Zdarsky, and I. Martinovic. Improving Performance Bounds in Feed-Forward Networks by Paying Multiplexing Only Once. In *Proc. GI/ITG MMB*, 2008.

[76] J. B. Schmitt, F. A. Zdarsky, and U. Roedig. Sensor Network Calculus with Multiple Sinks. In *Proc. of the Performance Control in Wireless Sensor Networks Workshop at the IFIP NETWORKING*, pages 6–13, May 2006.

[77] J. B. Schmitt, F. A. Zdarsky, and L. Thiele. A Comprehensive Worst-Case Calculus for Wireless Sensor Networks with In-Network Processing. In *Proc. IEEE RTSS*, pages 193–202, December 2007.

[78] H. She, Z. Lu, A. Jantsch, D. Zhou, and L.-R. Zheng. Performance Analysis of Flow-Based Traffic Splitting Strategy on Cluster-Mesh Sensor Networks. *International Journal of Distributed Sensor Networks*, 2012.

[79] D. Starobinski, M. Karpovsky, and L. A. Zakrevski. Application of Network Calculus to General Topologies Using Turn-Prohibition. *IEEE/ACM Transactions on Networking*, 2003.

[80] U. Suppiger, S. Perathoner, K. Lampka, and L. Thiele. Modular performance analysis of large-scale distributed embedded systems: An industrial case study. Technical Report 330, ETH Zurich, Nov 2010.

[81] U. Suppiger, S. Perathoner, K. Lampka, and L. Thiele. A simple approximation method for reducing the complexity of modular performance analysis. Technical Report 329, ETH Zurich, Aug 2010.

[82] L. Thiele, S. Chakraborty, M. Gries, and S. Künzli. Design Space Exploration of Network Processor Architectures. In *In Network Processor Design: Issues and Practices, Volume 1*, pages 30–41. Morgan Kaufmann Publishers, 2002.

[83] L. Thiele, S. Chakraborty, and M. Naedele. Real-Time Calculus for Scheduling Hard Real-Time Systems. In *Proc. ISCAS*, March 2000.

[84] J. Tomasik and M.-A. Weisser. Internet Topology on AS-level: Model, Generation Methods and Tool. In *Proc. IEEE IPCCC*, 2010.

[85] Y. L. Varol and D. Rotem. An Algorithm to Generate all Topological Sorting Arrangements. *The Computer Journal*, 24(1), 1981.

[86] E. Wandeler, L. Thiele, M. Verhoef, and P. Lieverse. System Architecture Evaluation Using Modular Performance Analysis - A Case Study. *Software Tools for Technology Transfer (STTT)*, 8(6):649 – 667, 2006.

[87] D. Wrege and J. Liebeherr. Video traffic characterization for multimedia networks with a deterministic service. In *Proc. IEEE INFOCOM*, volume 2, pages 537–544 vol.2, Mar 1996.

[88] D. Wu, D. I. Arkhipov, E. Asmare, Z. Qin, and J. A. McCann. UbiFlow: Mobility management in urban-scale software defined IoT. In *Proc. of IEEE INFOCOM*, pages 208–216, April 2015.

[89] T. Zhu, A. Tumanov, M. A. Kozuch, M. Harchol-Balter, and G. R. Ganger. PriorityMeister: Tail Latency QoS for Shared Networked Storage. In *ACM SOCC*, pages 1–4, 2014.

List of Publications

This thesis revises and extends on the following previous publications:

Conference Proceedings

2016 Achieving Efficiency Without Sacrificing Model Accuracy:
Network Calculus on Compact Domains
Kai Lampka, Steffen Bondorf and Jens B. Schmitt
In Proc. to the IEEE International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS ‘16), September 2016. Section 7.3.

2016 Should Network Calculus Relocate? An Assessment of
Current Algebraic and Optimization-based Analyses
Steffen Bondorf and Jens B. Schmitt
In Proc. of the International Conference on Quantitative Evaluation of Systems (QEST ‘16), August 2016. Sections 3.2 and 5.

2016 Improving Cross-traffic Bounds in Feed-Forward Networks –
There is a Job for Everyone
Steffen Bondorf and Jens B. Schmitt
In Proc. of the 18th International GI/ITG Conference on Measurement, Modelling and Evaluation of Dependable Computer and Communication Systems (MMB & DFT ’16), April 2016. Section 3.4.

2015 Calculating Accurate End-to-End Delay Bounds –
You Better Know Your Cross-Traffic
Steffen Bondorf and Jens B. Schmitt
In Proc. of the 9th EAI International Conference on Performance Evaluation Methodologies and Tools (ValueTools ‘15), December 2015. Sections 3.3 and 6.1.

2015 Boosting Sensor Network Calculus by
Thoroughly Bounding Cross-Traffic
Steffen Bondorf and Jens B. Schmitt
In Proc. of the 34th IEEE International Conference on Computer Communications (INFOCOM ‘15), April 2015. Section 4.

2014 The DiscoDNC v2 –
A Comprehensive Tool for Deterministic Network Calculus
Steffen Bondorf and Jens B. Schmitt
In Proc. of the 8th EAI International Conference on Performance Evaluation Methodologies and Tools (ValueTools '14), December 2014. Section 3.1.

2011 Pay Bursts Only Once Holds for (Some) Non-FIFO Systems
Jens B. Schmitt, Nicos Gollan, Steffen Bondorf and Ivan Martinovic
In Proc. of the 30th IEEE International Conference on Computer Communications (INFOCOM '11), April 2011. Section 2.

2010 Statistical Response Time Bounds in
Randomly Deployed Wireless Sensor Networks
Steffen Bondorf and Jens B. Schmitt
In Proc. of the 35th IEEE Conference on Local Computer Networks (LCN), October 2010. Section 4.

Web-Based Publications

2016 Delay Bounds in Feed-Forward Networks –
A Fast and Accurate Network Calculus Solution
Steffen Bondorf, Paul Nikolaus and Jens B. Schmitt
In arXiv 1603.02094 [cs.NI], March 2016. Sections 6.2, 7.1 and 7.2.

2015 On the Potential to Improve Accuracy of Network Calculus Analyses
Steffen Bondorf and Jens B. Schmitt
Technical report 392/15, University of Kaiserslautern, October 2015. Sections 3.2 and 5.

Curriculum Vitae

PERSONAL INFORMATION

Name	Steffen Bondorf
Nationality	German

EDUCATION

PhD Computer Science, University of Kaiserslautern, 2016,
Thesis title: Worst-Case Performance Analysis of Feed-Forward Networks –
An Efficient and Accurate Network Calculus
Supervisor: Jens B. Schmitt (University of Kaiserslautern)
Additional Reviewers: Roland Meyer (University of Kaiserslautern),
Marc Boyer (ONERA, France)

M.Sc. Computer Science, University of Kaiserslautern, 2012,
Thesis title: Performance Modeling of
Energy-Harvesting Wireless Sensor Networks
Supervisor: Jens B. Schmitt

B.Sc. Computer Science, University of Kaiserslautern, 2009,
Thesis title: Statistical Performance Bounds in
Wireless Sensor Networks with Random Topology
Supervisor: Jens B. Schmitt

HONORS AND GRANDS

2016	Postdoctoral scholarship, Carl Zeiss Foundation.
2016	ACM SIGMETRICS / IFIP Performance student travel grant.
2016	Best Presentation Award, 3rd Workshop on Network Calculus.
2012	PhD scholarship, Computer Science Department, University of Kaiserslautern.
2011	IEEE INFOCOM student travel grant.
2010	PhD graduate scholarship, Computer Science Department, University of Kaiserslautern.
2010	Admissions to the PhD program for talented students.
2006	Freshman scholarship, University of Kaiserslautern.
2005	Admission to B.Sc. studies (CS) before finishing secondary education (Frühstudium), first student at the University of Kaiserslautern.

TALKS

2016 Bounding Flow Arrivals in Feed-forward Networks
3rd Workshop on Network Calculus, GI/ITG MMB & DFT Workshops, April 2016.

2015 Network Calculus: The Quest for Tight Delay Bounds and the Struggle with Computational Effort
Uppsala University, Programming for Multicore Architectures Research Center, UpMarc Seminar, October 2015.

2015 Network Calculus Tool Support – Expectations and Reality
Leibniz Center for Informatics, Dagstuhl Seminar 15112, March 2015.

2015 Worst-Case Performance Analysis with Network Calculus,
CS department talk series, University of Kaiserslautern, January 2015.

2014 Cross-Traffic Arrival Bounds
2nd Workshop on Network Calculus, GI/ITG MMB & DFT Workshops, March 2014.

2013 The DISCO Network Calculator: Design, Capabilities and Future
Airbus Group Innovations, Munich, June 2013.

RESEARCH EXPERIENCE

2016 Integrated End-to-End Modeling and Analysis of Heterogeneous Distributed Systems with Real-time Constraints,
lead researcher,
2-year project funded by the Carl Zeiss Foundation.

2015 Research visit to Kai Lampka, Embedded Systems Group,
Uppsala University, Sweden.

2012 A Calculus for Networks with Flow Transformations,
researcher,
3-year project funded by the German Research Foundation, ended in 2015.

2010 PhD Program, Computer Science Department,
University of Kaiserslautern.

2010 Ambient Systems, federal state main research field, researcher.

2008 Developing a Calculus for Performance Analysis of Wireless Sensor Networks,
undergraduate research assistant,
3-year project funded by the German Research Foundation, ended in 2009.