Non-linear Programming for the Network Calculus Analysis of FIFO Feedforward Networks

Lukas Herll
Ruhr University Bochum
Bochum, Germany

Steffen Bondorf
Ruhr University Bochum
Bochum, Germany

Abstract

System designs for bounded communication latencies often employ a rather basic concept at their core: First-In First-Out (FIFO) queueing. Network Calculus (NC) can compute delay bounds for the end-to-end communication of data flows crossing potentially large feedforward networks of such First-In First-Out (FIFO) systems. Analysis complexity stems from the need to keep track of the interactions between flows when they compete for resources, i.e., multiplex in shared queues.

Network Calculus (NC) has an elegant solution to this: an open, so-called First-In First-Out (FIFO) parameter is introduced every time a (worst-case) First-In First-Out (FIFO) interaction occurs in the analysis. At the end of the analysis stands a (min,plus)-algebraic term with interdependent First-In First-Out (FIFO) parameters. We aim at finding an optimal setting for all open parameters. When employing standard optimization techniques, we cannot work with a parameterized (min,plus)-algebraic term directly. Thus, we show how to derive a minimum size (plus,times)-algebraic term that we can use with Non-Linear Program (NLP) solvers efficiently. Additionally, we show how to differentiate this term to open our approach to gradient-based Non-Linear Program (NLP) algorithms.

In numerical evaluations, we show that our approach outperforms the complexity/accuracy tradeoff of existing heuristics to set the First-In First-Out (FIFO) parameters. With a slight increase of analysis runtime, we reduce the gap to the optimal setting by a factor of 4.4, to 0.15% on average.

CCS Concepts

• **Networks** → **Network performance evaluation**; *Network performance analysis*; Network performance modeling; • **Computing methodologies** → **Symbolic and algebraic algorithms**; *Symbolic calculus algorithms*; Optimization algorithms.

Keywords

Delay bounds, network calculus, FIFO systems, LUDB

ACM Reference Format:
Lukas Herll and Steffen Bondorf. 2025. Non-linear Programming for the Network Calculus Analysis of FIFO Feedforward Networks. In *Proceedings of the 16th ACM/SPEC International Conference on Performance Engineering (ICPE '25), May 5–9, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3676151.3719360

ICPE '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1073-5/2025/05
https://doi.org/10.1145/3676151.3719360

1 Introduction

Bounded end-to-end delay of a data flow communication is essential for many applications, e.g., on factory floors and in the avionics sector. Thus, modern networking standards such as IEEE Time-Sensitive Networking (TSN) and IETF Deterministic Networking (DetNet) aim for system designs that guarantee such bounds.

One of the core design features of these systems is the use of First-In First-Out (FIFO) queues where multiple flows may multiplex in FIFO order. E.g., in IEEE 802.1Qbv, part of the TSN standard, output ports of switches are composed of eight such FIFO queues of different priorities whose forwarding is additionally restricted by a time-based transmission selection. Analyzing worst-case delay bounds in a network of such queues is challenging, yet, Network Calculus (NC) results exist. Foremost, [26] provides the first model for the worst-case forwarding guarantee of a FIFO queue in such a configuration. The results in [26] already outperform previous ones [10] significantly regarding the derived delay bounds. However, the advanced model was used in a less sophisticated NC analysis, leaving room for improvement. Another restriction compared to the previous work is the lack of derivation of configuration parameters for such a TSN switch. Regarding NC, both these aspects are related: the advanced FIFO analysis called Least Upper Delay Bound (LUDB) requires finding the setting for free parameters as does any other form of network configuration synthesis. Therefore, we aim at applying a method recently proposed to synthesize network configurations under delay constraints, called Differential NC (DiffNC) [14]. Our work thus takes the next step towards efficiently synthesizing TSN configurations subject to accurately bounded delays in FIFO queueing networks.

Analyzing such FIFO queueing networks is not trivial. To do so, NC derives a parameterized (min,plus)-algebraic terms with free parameters that "encode" the impact of each FIFO interaction of flows on each other. The challenge is then to efficiently find a setting for each parameter such that the computed delay bound is minimized, see for example [2, 24]. We propose to apply the DiffNC idea to it which entails to solve these complex, parameterized operations in (plus,times)-algebra. [23] provides empirical evidence that the problem shape is convex such that we also differentiate the term to test gradient-based algorithms. Somewhat surprisingly, we show that differentiation and gradient-based optimization need not be superior to optimization algorithms that do not make use of the gradient. Overall, our contributions are:

(1) derivation of a minimal, parameterized (plus,times)-algebraic term bounding the analyzed flow of interest (foi) end-to-end delay under FIFO assumptions, to be used in a Non-Linear Program (NLP) solver,
(2) differentiation of this term to allow for use of gradient-based algorithms,

(3) an open-source extension to the NetworkCalculus.org Deterministic Network Calculator (NCorg DNC) for automated NLP LUDB analysis[1], and
(4) extensive numerical evaluation where we show that using NLP algorithms, even non-gradient-based ones, can outperform previous approaches by a significant margin. Compared to the heuristic recommended in [24], we reduce the gap to the optimal setting by a factor of 4.4, to 0.15% on average while only adding slightly to the computation time.
This may even come as a surprise as the term from contribution 1 is actually a piecewise linear function, not a non-linear one.

The remainder of the paper is organized as follows: Section 2 provides the background on NC and building on it, Section 3 gives an overview on the history of the NC FIFO analysis, i.e., surveys the related work. In Section 4, we present the derivation of the minimal, parameterized (plus,times)-algebraic term bounding the end-to-end delay, its utilization in an NLP formulation, how to differentiate it for gradient-based NLP, and our open-source toolchain. Section 5 benchmarks our analysis with respect to existing ones before Section 6 concludes the paper.

2 Deterministic Network Calculus

2.1 Network Calculus Modeling

NC is based on the notion of networks of queueing locations. These locations consist of a queue for buffering incoming data of potentially multiple flows that is served by some aggregate scheduler. In the NC model, these two parts are generally depicted as a single entity, a so-called server. Servers can model rather simple constructs like a single FIFO queue served by a constant rate link [6] but also parts of more involved architectures such as a single priority queue in a TSN design that is controlled by an IEEE 802.1Qbv Time-Aware Shaper (TAS) [26].

Queueing locations of a network form a graph, the so-called server graph $\mathcal{G} = (\mathcal{S}, \mathcal{E})$ where $\mathcal{S}$ denotes the set of servers and $\mathcal{E}$ the set of directed edges between servers (see Figure 1a for an example). We assume that the server graph possesses the feedforward property, i.e., flows that cross the server graph cannot form cyclic dependencies. All further characteristics are modeled by functions of the set

$$\mathcal{F}_0^+ = \left\{ f : \mathbb{R}^+ \to \mathbb{R}^+ \mid f(0) = 0,\ \forall s \le t\ :\ f(t) \ge f(s) \right\}.$$

These non-decreasing functions allow modeling of cumulative system behavior. That is, traffic regulation as well as forwarding service guarantees. The effort to work with, e.g., staircase functions [17] or pseudo-periodic functions [27] can, however, easily make an analysis intractable. In practice, piecewise-linear functions from $\mathcal{F}_0^+$ are therefore most commonly found to model regulation and service [3, 19, 26].

Flows cross the server graph and data put into it by them is assumed to be upper bounded at the ingress server in $\mathcal{G}$, by an arrival curve:

[1]Our code and data can be accessed at https://github.com/Lukasssssssssss/ICPE2025-Non-linear-Programming-for-the-Network-Calculus-Analysis-of-FIFO-Feedforward-Networks.

Definition 1 (Arrival Curve). *Assume a flow f puts data into the network as per function $A \in \mathcal{F}_0^+$ that cumulatively counts that data. Then a function $\alpha \in \mathcal{F}_0^+$ is an arrival curve for f if and only if*

$$\forall\, 0 \le d \le t\ :\ A(t) - A(t-d) \le \alpha(d). \tag{1}$$

Servers $s \in \mathcal{S}$ offer forwarding of queued data. NC models the minimum forwarding guarantee as service curves:

Definition 2 (Service Curve). *If a server receives a cumulative data input described by $A \in \mathcal{F}_0^+$ and produces an output $A' \in \mathcal{F}_0^+$, then it is said to offer service curve $\beta \in \mathcal{F}_0^+$ if and only iff*

$$\forall t : A'(t) \ge \inf_{0 \le d \le t} \{A(t-d) + \beta(d)\}. \tag{2}$$

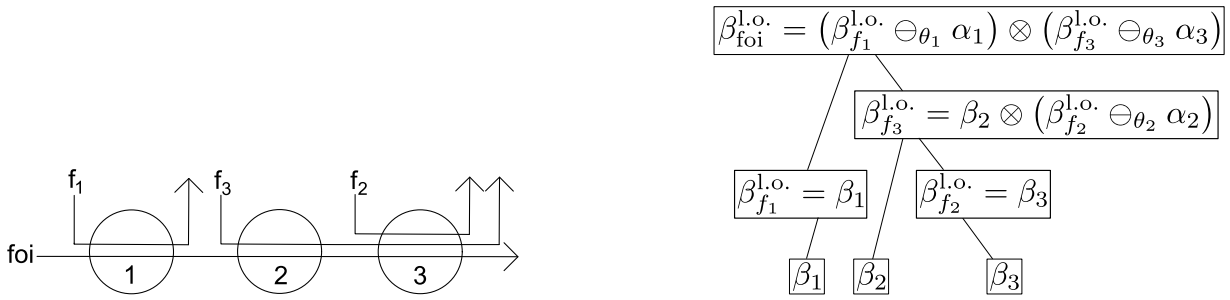
Some behavioral aspects of the queueing locations, , i.e., the edges $\mathcal{E}$ in the server graph $\mathcal{G}$, are not captured by the input/output-centric model explicitly but are assumed instead: While a service curve gives a forwarding guarantee to queued data, it does not define the order in which this data is forwarded. Put simply, assuming that the service is always provided to the head of the queue, NC additionally requires assumptions on a) the behavior of the queue and b) the resulting order when multiple flows multiplex into a single queue. For a), order-preserving FIFO queues are assumed as they are easily and commonly implemented. An important side-effect is that FIFO queues retain the order of data within a single flow. Only few works in the literature deal with the alternative non-FIFO systems [25] that do not preserve any queueing order. The impact on multiplexing different flows into one FIFO queue on the overall service guarantee is, however, a a more widespread research area in NC. Two regimes have been researched extensively: i) arbitrary multiplexing where multiplexing can result in any oder between flows and ii) the order-preserving FIFO multiplexing. The former, arbitrary multiplexing, constructs the worst-case interaction between the analyzed data flow (the foi) every time it multiplexes with another flow. This simplifies the NC analysis tremendously as the worst case is simple to model: the analyzed flow only receives service when all other flows have been served. FIFO multiplexing of ii) is in stark contrast to this as it requires the analysis to trace the multiplexing flows across multiple servers and find the worst case for this under FIFO. Yet, modern systems such as TSN are FIFO systems forming a FIFO network [26] and thus, it is worthwhile to pursue the FIFO analysis that delivers more accurate bounds for them.

2.2 Network Calculus Analysis

An NC analysis aims to compute an upper bound on the end-to-end delay of an foi. In this paper, we use the "classical" (min,plus)-algebraic analysis. It is based on multiple operations that need to be arranged to a delay-bounding term reflecting the mutual impact of flows crossing the server graph $\mathcal{G}$.

In (min,plus)-algebraic NC, systems are modeled in terms of data rates (cf. arrival curves and service curves in Definition 1 and Definition 2), i.e., functions of time. A dual approach using (max,plus) algebra exists where systems are modeled as functions of data, i.e., the timestamp of the data arrival or departure is returned by the functions. We refer the interested reader to [21] for details.

The following binary infix operations work independently of the FIFO assumption:



(a) A nested three-server tandem with four flows, the foi and cross-flows f_1, f_2 and f_3. Edges are implicit by flows traversing between servers.

(b) The nesting tree representing the (min,plus)-algebraic left-over service curve computation $(\beta_1 \ominus_{\theta_1} \alpha_1) \otimes ((\beta_2 \otimes (\beta_3 \ominus_{\theta_2} \alpha_2)) \ominus_{\theta_3} \alpha_3)$.

Figure 1: Nested tandem and nesting tree example (©A. Scheffler, J. Schmitt, S. Bondorf [24]).

Definition 3 (NC (min,plus) Operations [6, 8]). *Let $\alpha_1, \alpha_2 \in \mathcal{F}_0^+$ be arrival curves and $\beta_1, \beta_2 \in \mathcal{F}_0^+$ be service curves. Then*

$$\textit{aggregation:} \quad (\alpha_1 + \alpha_2)(d) = \alpha_1(d) + \alpha_1(d) \tag{3}$$

$$\textit{concatenation:} \quad (\beta_1 \otimes \beta_2)(d) = \inf_{0 \le u \le d} \{\beta_1(d-u) + \beta_2(u)\} \tag{4}$$

$$\textit{output bounding:} \quad (\alpha_1 \oslash \beta_1)(d) = \sup_{u \ge 0} \{\alpha_1(d+u) - \beta_1(u)\} =: \alpha_1'(5) \tag{5}$$

Equation 3 allows for handling two flows as if they were a single one (concatenation), Equation 4 allows for working with a tandem of servers as if the servers offered a single service curve, and Equation 5 allows to compute arrival curves within a network, i.e., after a flow (or flow aggregate) crossed a server (or concatenation of servers).

The FIFO assumption becomes crucial if we want to compute a lower bound on the remaining worst-case left-over service in presence of a crossflow (or flow aggregate) at a server (or concatenation of servers). That is, the remaining guarantee after another arrival curve was deducted from a service curve.

Theorem 1 (FIFO left-over Service Curve [12]). *Assume flow f with arrival curves α is served by a FIFO server with service curve β. Then, the left-over service curve considering the forwarding of f is*

$$\beta^{l.o.}(t) = [\beta(t) - \alpha(t-\theta)]^{\uparrow} \cdot \mathbb{1}_{\{t>\theta\}} =: \beta \ominus_\theta \alpha, \quad \forall \theta \ge 0 \tag{6}$$

where $[g(x)]^{\uparrow} = \sup_{0 \le z \le x} g(z)$, and $\mathbb{1}_{\{condition\}}$ is the indicator function evaluating to 1 if the condition is met and 0 otherwise. We call $\theta \in \mathbb{R}^+$ the FIFO parameter.

For a graphical representation, refer to Figure 2b.

The theorem provides a binary infix operation that is comparable to those of Definition 3. A single service curve and a single arrival curve are used, despite the FIFO assumption that generates interdependency between all flows at a server, i.e., including the flow/s that use this guaranteed left-over service. The trick is that the FIFO parameter θ "encodes" all potential worst-case FIFO interactions. That is, $\beta^{\text{l.o.}}$ represents an infinite set of left-over service curves, θs must be set when all arrival curves are known. Then, a delay bound can be computed. Any setting of θs will lead to a valid bound[2], yet, only one is optimal in the sense that it will allow for computation of the tightest delay bound.

Eventually, the (min,plus)-algebraic, delay-bounding term for an foi consists of the foi's end-to-end left-over service curve and its arrival curve.

Theorem 2 (FIFO Delay Bound [6]). *Assume flow f with arrival curve α that crosses a server (or sequence of servers) guaranteeing (end-to-end / left-over) service curve β to it. In FIFO systems, the end-to-end delay of f is bounded by the horizontal deviation between both curves:*

$$hDev(\alpha, \beta) := \inf \{d \ge 0 \mid (\alpha \oslash \beta)(-d) \le 0\}. \tag{7}$$

Note, that Theorem 2 assumes the FIFO parameter to be set to compute the horizontal deviation between α and one fully specified β. The delay bound is sketched in Figure 2a.

3 Related Work

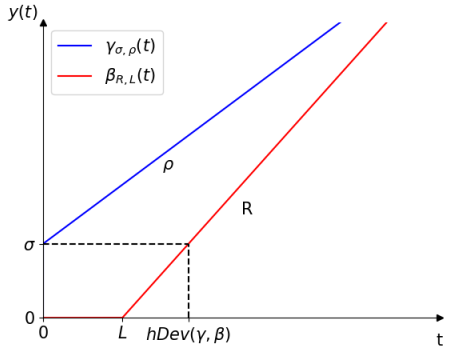
FIFO behavior is a natural assumption and a relatively straightforward one to implement, for multiplexing as well as for forwarding of data. However, the analysis of FIFO systems with NC turned out to have its intricacies. We provide an overview of its history, to survey the work related to our paper, followed by a more in-depth depiction of the NC analysis we base our contribution on.

3.1 FIFO Analysis in NC

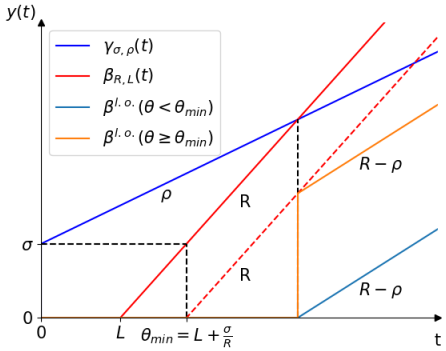
Before Cruz proposed the FIFO left-over service curve in [12], the analysis had to aggregate all flows crossing a server. Given the FIFO property, the horizontal deviation as shown in Theorem 2 bound the delay for each flow as well as the whole aggregate, i.e., the totality of all flows. All delay bounds on the foi path are then summed up to its end-to-end delay [11]. This analysis, known as Total Flow Analysis (TFA), still sees applications today, e.g., in [26].

The so-called Separate Flow Analysis (SFA) applies the FIFO left-over service curve theorem to separate the foi from its cross-traffic at each server [6]. All left-over service curves are then concatenated to a single end-to-end left-over service curve as mentioned in Theorem 2, yet with one θ parameter per server. The SFA is known to

[2]For all values of θ smaller than the sum of inherent service latency and the time to work off the interfering burstiness, there will always be a larger superior θ value. Thus, we lower bound our θ with this value, yet, without introducing an explicit syntax. There is no upper bound for θ, i.e., the set remains infinite. In the literature, this is known as the difference between left-over service curve sets E and $\overline{E}$ [1]. For a sketch of the left-over service curve operation and the set reduction, see Figure 2b.



(a) A token-bucket arrival curve $\gamma_{\sigma,\rho}$ and a rate-latency service curve $\beta_{R,L}$. The arrival curve depicts a constant (accumulative) data influx with rate ρ after an initial burst σ. The service curve offers a constant (cumulative) service with rate R after an initial latency L.
We can bound the delay of arrival curve $\gamma_{\sigma,\rho}$ after being serviced according to service curve $\beta_{R,L}$ by computing the time it takes for $\beta_{R,L}$ to service the initial burst σ of $\gamma_{\sigma,\rho}$, as depicted by the dashed line. Computing the delay bound, $hDev(\gamma_{\sigma,\rho}, \beta_{R,L})$, is generally equivalent to computing the horizontal distance at the height of the arrival curve burst.



(b) This is a sketch of the left-over operation. The forwarding guarantee for flow f is represented by a set of functions. Assuming a token-bucket constraint on f and a rate-latency service curve, the left-over service curve set results from subtracting the former, right-shifted by $\theta \geq 0$, from the latter while mapping negative results as well as $t \geq \theta$ to zero. Finding the optimal curve for delay bounding from this infinite set of service curves equals finding the optimal θ. It is possible to reduce the search space: $\theta > \theta_{min}$, defines curves with a constantly lower service guarantee (e.g. the light-blue curve) than others, $\theta \geq \theta_{min}$ (e.g. the orange curve). The remaining θ values define our optimization problem. For a more rigorous derivation, refer to [1].

Figure 2: Arrival and service curves, as well as the delay bounding and FIFO left-over operations.

result in more accurate delay bounds than the TFA as it implements to so-called Pay Bursts Only Once (PBOO) principle: separating the foi and concatenating the left-over service curve means the foi arrival curve will not be required at each server. That is, its burstiness need not be explicitly bounded at each server (by output bounding of Definition 3) for the delay bound computation. Instead, it only appears once in the derivation, when applying Theorem 2.

A further improvement of the delay bound was found to be possible by extending the PBOO principle to the cross-flows [13]. The approach attempts to concatenate servers before the left-over operation. In this analysis, the θ parameters are set to a fixed value a priori. This was shown to be inferior to leaving the θ parameters open and optimizing them after the symbolic, delay-bounding term was derived. The analysis doing so is called LUDB [20]. Details, including the known attempts to set the interdependent θ parameters, are presented in Section 3.2 and Section 3.3.

The LUDB requires to decompose a feedforward network into tandems of servers without overlapping interference. This decomposition may require to bound flows that are found on adjacent tandems, i.e., PBOO for cross-traffic commonly known as Pay Multiplexing Only Once (PMOO) is not fully achieved. A countermeasure that can further improve accuracy of the LUDB was recently proposed in [16].

3.2 The LUDB for Feedforward Networks

Conceptually, Least Upper Delay Bound (LUDB) for Feedforward Networks (LUDB-FF), an extension of LUDB for analysis of tandems, consists of two parts:

a) the feedforward (FF)-part decomposes the server graph into a sequence of tandems (see [3, 23] for details) and then
b) the LUDB-part decomposes the tandem under analysis into sequences of tandems without overlapping interference and computes a bound, i.e., delay on it or output from it.

We focus on the individual tandems at the end of both decompositions. These tandems are called nested tandems. All flows on a nested tandem have either disjunct paths or a flow path is a subpath of a different flow path. The subpath relationship allows for subsequent application of the binary infix operations defined in Definition 3 and the left-over service curve Theorem 1 that introduces the free θ. This order of operations can be encoded as a binary tree, the so-called nesting tree. An example is shown in Figure 1. The nesting tree also depicts the interdependencies between FIFO parameters of the analysis.

LUDB proposes to use optimization algorithms to find the values for the θs in the nesting tree. It is restricted to rate-latency service curves $\beta_{R,L}(t) = R[t - L]^+, \forall t \geq 0$, and token-bucket arrival curves $\gamma_{\sigma,\rho}(t) = \sigma + \rho \cdot t, \forall t \geq 0$ (see Figure 2a). The authors identify that its objective function will then be a Piecewise Linear Program (PLP) and state that "A closed-form solution for a generic nested tandem is still missing as of today" [2]. Therefore, the PLP is decomposed into a set of LPs by a recursive procedure that traverses the nesting tree. The LPs are then solved, i.e., the optimal θs are found, for delay or output bounding.

It was shown in [23] that solving the LPs (using IBM CPLEX) requires the majority of the computation time, usually above 95%. The authors propose to use a directed search as a fast yet accurate

heuristic instead. This search, a directed search called DS-FF, evaluates the nesting tree for a setting of the θs without the need to derive a closed-form solution. Moreover, its termination criterion can be freely set in order to steer the tradeoff between runtime and accuracy. The authors show that the effort of the search scales superlinearly with this termination criterion. Thus, it is to be expected that the addition of further parameters to the objective would only be feasible if traded against delay bound accuracy. Unfortunately, while this previous work identified the LUDB problem shape as most probably convex (not proved), it did not evaluate gradient-based algorithms.

3.3 Optimization in NC and Differential NC

Optimization tools have been integrated into the NC in various ways. The one that deviates to the greatest extend from the background theory in Section 2 is the holistic optimization. It converts the NC system model of Section 2.1 into a Mixed-Integer Linear Program (MILP) [9], making the implicit FIFO assumption explicit by adding FIFO ordering constraints. While this model theoretically allows for always computing the tight bound – in contrast to the (min,plus)-algebraic theorems of Section 2.2 – it strongly depends on the tool support to solve the MILP formulation. For this reason, latest research on the branch of holistic optimization shifted towards finding a good tradeoff between tractability and accuracy of the analysis [7]. It involves adding TFA and SFA results as constraints as they are fast to compute and can be sufficiently accurate to make up for other removed constraints.

In this paper, we integrate NLP into LUDB to derive the θ values. That is, we use optimization tools after having applied the theorems of Section 2.2, the "classical" (min,plus)-algebraic NC analysis. This "classical" analysis is, in principle, a symbolic analysis that proceeds in three steps: i) derive a (min,plus)-algebraic term that lower bounds the forwarding service that the current foi is guaranteed to get, ii) fill in the constraints: upper bounds for the amount of data that flows can put into the network and lower bounds for the forwarding service offered at queueing locations, iii) evaluate the (min,plus)-algebraic term, i.e., compute an end-to-end delay bound for the foi.

Step i) itself constitutes a compositional analysis of interfering cross-traffic. It requires backtracking of flow dependencies which has seen much treatment in the literature [4, 5, 16]. Steps ii) and iii) constitute the (min,plus) core of NC. Constraints are modeled as cumulative functions in interval time, the so-called curves. The use of (min,plus)-algebraic operations results in new, intermediate curves that bound the result of interactions between data flows and forwarding service at queueing locations. Examples are flow aggregation or left-over service guarantees. It was shown that this three-step approach for the network analysis exposes better scalability than the holistic optimization [3]. Regarding the derived delay bounds, it can be competitive with the holistic analysis. Overall, it provides a better and somewhat steerable trade-off between complexity and quality. For that reason, that the holistic analysis was even amended to use algebraic results as additional constraints [7].

While this speaks for a (min,plus)-algebraic analysis, the approach has its limitations, too. So far, step ii) requires to fully specify all parameters in the (min,plus)-algebraic term as there did not exist an analysis for step iii) that could handle open parameters. The flexibility to do so is a key strength of the holistic optimization and a potential motivation to invest the computational effort into solving an NC-derived optimization problem. However, recent work called DiffNC [14] broke ground for a different analysis in step iii). Conceptually, the proposal is to take the symbolic, (min,plus)-algebraic term with open parameters, convert it into a (plus,times)-algebraic term and use that term as objective function in an NLP. Depending on the shape of bounding functions, this conversion can be relatively simple. The running example of that work uses linear approximations and illustrates how the search for delay-bound-optimizing flow paths can be solved efficiently. The name DiffNC is derived from the fact that gradient-based NLP algorithms performed best on this problem, i.e., differentiation of the (plus,times)-algebraic term was performed in a step preceding the optimization. DiffNC was theorized to be extended to synthesize further network-defining parameters as they are commonly found in standards like TSN. Examples given in [14] are priority assignment and TDMA schedule optimization. The former has been presented in [15].

Our paper presents another direction of advancement towards efficiently synthesizing TSN configurations: the integration of NLP into LUDB to derive the θ values. In that, it outperforms the search-based approaches presented in Section 3.2.

4 NLP LUDB

In contrast to the original LUDB analysis we aim at directly bounding the delay of each nested tandem, thereby providing the missing closed-form solution for generic nested tandems. To achieve this, we need to find closed-form expressions in (plus,times) algebra for the results of the (min,plus)-algebraic NC operations defined in Definition 3. The key to this is to sensibly restrict the input curve shapes. We use token-bucket and rate-latency curves as LUDB traditionally does, and abstract from them to a more general curve shape, the class of pseudoaffine curves, which we show are closed under the NC operations. We then derive a (plus,times)-algebraic term bounding an foi end-to-end delay by recursively applying the NC operations along the foi path in the tandem. We strive for a minimal expression where we pre-compute as much as possible to allow for the NLP solver to evaluate efficiently. This step entails to exhaustively distinct all cases that can occur during computations with pseudoaffine curves and reduce to the relevant cases.

4.1 The Delay Bounding Term in (plus,times) Algebra

Bounding the service at each server in the server graph by a rate-latency service curve $\beta_{R,L}$ and each flow by a token-bucket arrival curve $\gamma_{\sigma,\rho}$, we obtain the following (plus,times) expressions of the (min,plus) operations:

Lemma 3 (Closed-form expression of NC operations [6]). *With the assumption of using rate-latency service curves and token-bucket arrival curves, the NC operations listed in Section 2.2 have the*

following closed-form solutions:

$$\textit{aggregation:} \quad \gamma_{\sigma_1,\rho_1} + \gamma_{\sigma_2,\rho_2} = \gamma_{\sigma_1+\sigma_2,\rho_1+\rho_2} \tag{8}$$

$$\textit{concatenation:} \quad \beta_{R_1,L_1} \otimes \beta_{R_2,L_2} = \beta_{\min(R_1,R_2),L_1+L_2} \tag{9}$$

$$\textit{output bounding:} \quad \gamma_{\sigma,\rho} \oslash \beta_{R,L} = \gamma_{\sigma+\rho\cdot L,\rho} \tag{10}$$

$$\textit{left-over [2]:} \quad \beta_{R,L} \ominus_\theta \gamma_{\sigma,\rho} = \gamma_{R[\theta-(L+\frac{\sigma}{R})],R-\rho}(t-\theta), \quad \theta \geq L + \tfrac{\sigma}{R}$$
$$\overset{s:=\theta-(L+\frac{\sigma}{R})}{=} \delta_{s+L+\frac{\sigma}{R}} \otimes \gamma_{Rs,R-\rho}$$
$$=: \beta_{R,L} \ominus_s \gamma_{\sigma,\rho}, \quad s \geq 0 \tag{11}$$

$$\textit{delay bound:} \quad hDev(\gamma_{\sigma,\rho}, \beta_{R,L}) = \sigma/R + L \tag{12}$$

under the condition that $\rho \leq R$ to not overload the system.

See Section 2.2 and Figure 2b for conditions on the above θ.

The FIFO left-over service curve operation is not closed in the set of rate-latency curves, i.e., the infinite set of resulting curves does not only consist of rate latencies. Instead, we obtain a so-called pseudoaffine curve (see Figure 3a) for each setting of θ [18]):

$$\pi = \delta_D \otimes \left[\bigotimes_{1 \leq x \leq n} \gamma_{\sigma_x,\rho_x} \right] \tag{13}$$

where parameter D, the initial latency, follows from the computation in Equation 11 and the γ functions are called token-bucket stages. That is, with the open θ, we have an infinite set of pseudoaffine FIFO left-over service curves. We denote the curve's long-term rate $\rho^*_\pi = \min_{x=1,\ldots,n}(\rho_x)$ and demand that $\rho^* \leq R$.

The set of pseudoaffine curves is indeed a common generalization of token-bucket and rate-latency curves: $\beta_{R,L} = \delta_L \otimes \gamma_{R,0}$ and $\gamma_{\sigma,\rho} = \delta_0 \otimes \gamma_{\sigma,\rho}$.

Pseudoaffine curves are closed under convolution and FIFO left-over (first stated in [18]), allowing to eventually compute a delay bounding term consisting of pseudoaffine, potentially θ-parameterized, curves:

Theorem 4 (Closeness under convolution [18]). *Assume two pseudoaffine curves*

$$\pi_n = \delta_{D_n} \otimes \left[\bigotimes_{1 \leq x \leq n} \gamma_{\sigma_x,\rho_x} \right] \quad \textit{and} \quad \pi_m = \delta_{D_m} \otimes \left[\bigotimes_{n+1 \leq x \leq m} \gamma_{\sigma_x,\rho_x} \right],$$

then their convolution produces another pseudoaffine curve with combined token-bucket stages:

$$\pi_n \otimes \pi_m = \delta_{D_n+D_m} \otimes \left[\bigotimes_{1 \leq x \leq n+m} \gamma_{\sigma_x,\rho_x} \right] \tag{14}$$

The convolution of two one-stage pseudoaffine curves can be seen in Figure 3b.

Theorem 5 (Closeness under left-over [18]). *For a pseudoaffine service curve $\pi = \delta_D \otimes \left[\bigotimes_{1 \leq x \leq n} \gamma_{\sigma_x,\rho_x}\right]$, and token-bucket arrival curve $\alpha = \gamma_{\sigma,\rho}$ with $\rho \leq \rho^*$, the set of relevant left-over service curves is given by*

$$\pi \ominus_\theta \alpha = \pi \ominus_s \alpha = \delta_{D^{l.o.}} \otimes \left[\bigotimes_{1 \leq x \leq n} \gamma_{\sigma_x^{l.o.},\rho_x^{l.o.}} \right] \textit{ with} \tag{15}$$

$$D^{l.o.} = hDev(\alpha,\pi) + s \tag{16}$$

$$\sigma_x^{l.o.} = \rho_x \{s + hDev(\alpha,\pi) - D\} - (\sigma - \sigma_x) \tag{17}$$

$$\rho_x^{l.o.} = \rho_x - \rho \tag{18}$$

where $s := (\theta - hDev(\alpha,\pi)) \in \mathbb{R}^+$.

Similarly to Equation 11, we can implicitly lower bound our θ by introducing a new parameter s. For any $\theta < hDev(\alpha,\pi)$, there is a $\theta \geq hDev(\alpha,\pi)$ whose corresponding left-over service curve is superior. Analogously to left-over operations on rate-latency curves, this is denoted in literature as the difference between left-over service curve sets E and $\overline{E}$ [18]. We provide an exhaustive proof of this in Section B.

The output bound computation remains closed in the set of token-bucket curves, even for the infinite set of pseudoaffine FIFO left-over service curves.

Theorem 6 (Closeness under output bounding [19]). *Assume a pseudoaffine left-over service curve $\pi \ominus_s \gamma_{\sigma_x,\rho_x}$ and a token-bucket arrival curve $\alpha = \gamma_{\sigma,\rho}$, where $\rho^*_\pi \geq \rho$ holds. Then, the output bound is given by*

$$\gamma_{\sigma,\rho} \oslash (\pi \ominus_s \gamma_{\sigma_x,\rho_x}) = \gamma_{\sigma,\rho} \oslash \left(\delta_{D^{l.o.}} \otimes \left[\bigotimes_{1 \leq x \leq n} \gamma_{\sigma_x^{l.o.},\rho_x^{l.o.}} \right] \right)$$
$$= \gamma_{\sigma+\rho\cdot D^{l.o.},\rho} \tag{19}$$

This means, when decomposing the server graph into nested tandems and bounding the output of flows crossing them, the flows' arrival curves will always have a token-bucket shape. That is, all arrival curves – even the intermediate output bounds – to be removed from pseudoaffine service curves by the left-over operation will be token buckets, as assumed in Theorem 5. Note, that Theorem 6 presents the most generic service curve, resulting from subsequent FIFO left-over operations. Special, commonly found instantiations are:

$$\gamma_{\sigma,\rho} \oslash (\beta_{R,L} \ominus_s \gamma_{\sigma_x,\rho_x}) = \gamma_{\sigma,\rho} \oslash (\delta_{s+L+\frac{\sigma_x}{R}} \otimes \gamma_{Rs,R-\rho_x})$$
$$= \gamma_{\sigma+\rho\cdot(L+\frac{\sigma_x}{R}),\rho} \tag{20}$$

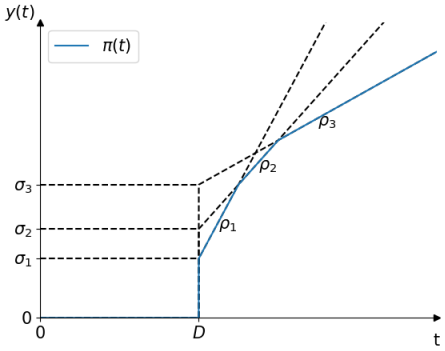
$$\gamma_{\sigma,\rho} \oslash \pi = \gamma_{\sigma,\rho} \oslash \left(\delta_D \otimes \left[\bigotimes_{1 \leq x \leq n} \gamma_{\sigma_x,\rho_x}\right]\right)$$
$$= \gamma_{\sigma+\rho\cdot D,\rho} \tag{21}$$

Finally, for the delay bound, we get the following expression for each element of the infinite set, such that the least one among them is the Least Upper Delay Bound (LUDB):

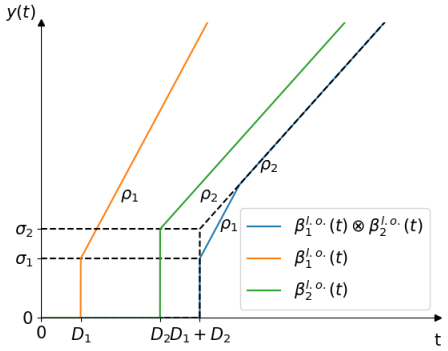
Theorem 7 (Delay Bound [18]). *Let π be a pseudoaffine curve $\pi = \delta_D \otimes \left[\bigotimes_{1 \leq x \leq n} \gamma_{\sigma_x,\rho_x}\right]$ and let α be a token-bucket arrival curve $\gamma_{\sigma,\rho}$ with $\rho^*_\pi \geq \rho$. Then, the delay bound $hDev(\alpha,\pi)$ can be calculated using the pseudo-inverse $\pi^{-1}(\cdot)$ of $\pi(\cdot)$ defined in [6]:*

$$hDev(\alpha,\pi) = \pi^{-1}(\sigma) = D + \left[\bigvee_{1 \leq x \leq n} \frac{\sigma - \sigma_x}{\rho_x} \right]^+ \tag{22}$$

From Theorem 3 and Theorems 4 to 7 we now know the shape of curves in our analysis as well as the delay bounding. By recursively applying these theorems along the path of the foi in the



(a) A pseudoaffine curve $\pi(t)$ consists of many token-bucket "stages". The curve form is given by an initial latency D and the convolution of its stages.



(b) The origin of a pseudaffine curve lies in the FIFO left-over operation resulting in a function that resembles a shifted token-bucket curve (the orange curve in figure 2b). These functions are already pseudoaffine curves. Convolving such functions then introduces more stages.

Figure 3: The structure and origin of pseudoaffine curves.

tandem, we arrive at a closed-form (plus,times)-algebraic term for the end-to-end delay bound of said foi. This closed-form expression forms a parameterized piecewise-linear function, with one FIFO parameter for each crossflow on the foi path. A graphical sketch of an exemplary computation for the network in Figure 1a is given in Appendix A.

4.2 Using Non-linear Optimization

The closed-form solution, our parameterized delay-bounding term in (plus,times) algebra, can now directly be used as the objective function of an optimization formulation without decomposition into LPs as proposed by the original LUDB. We can confirm that the function is piecewise linear and while it is not proven that our problem space is convex, results in [24] suggest it. A convex problem space opens the way for gradient-based optimization strategies.

Inspired by the efficiency of DiffNC [14] that uses NLP, we opted to replicate that approach. Even though NLP solvers do not guarantee for optimality – their result depends on the applied algorithm and the specific optimization problem – we will show in Section 5 that the derived delay bounds are competitive and even outperform existing analyses. Additionally, NLP solvers are applicable to a wider range of optimization problems. This makes our analysis more adaptable to possible future changes and extensions.

In [14], a different problem is formulated as a relaxed mixed-integer nonlinear programming (MINLP). This class of optimization problems include mixed-integer linear problems, which are NP-hard, and inherits this property from them. The authors showed that gradient-based NLP algorithms not only outperform non-gradient-based algorithms in solving the NLP, they are also seem sufficiently efficient to be extended to more complex problem formulations. Lastly, DiffNC found that a gradient-based NLP delivered the best tradeoff between analysis runtime and accuracy. It is therefore reasonable to include gradient-based NLP algorithms into our set of solvers. Thus, we will show how to differentiate the delay-bounding term to open our NLP approach up to these algorithms.

4.3 Differentiating Network Calculus Terms with FIFO left-over Operations

The parameterized delay-bounding term is obtained by applying Theorems 4 to 5 as we progress through the nesting tree while keeping all FIFO parameters open. To use gradient-based optimization algorithms for finding the optimal settings of the FIFO parameters, we need to derive the gradient of the delay-bounding term w.r.t to the FIFO parameters.

Theorem 8 (Differentiability of the Delay Bound). *Assuming pseudoaffine service curves and token-bucket arrival curves, the parametrized NC end-to-end delay bound is differentiable w.r.t the open FIFO parameters.*

Proof. Using the operations defined in Theorems 4 to 5, the symbolic term for the delay bound uses the following basic operations: addition, subtraction, multiplication, division, max, and $[\cdot]^+$. For the max operator, we define the following partial derivatives for $x \neq y$ [14]

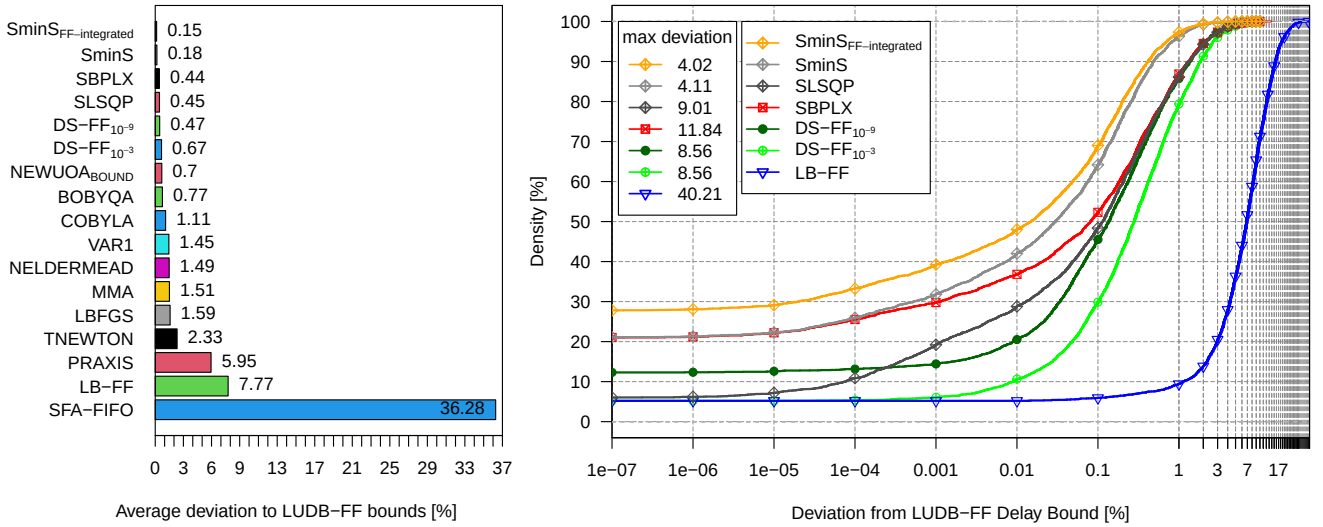
$$\frac{\partial \max(x,y)}{\partial x} = \begin{cases} 0, \text{ if } x < y \\ 1, \text{ if } x > y \end{cases} \qquad \frac{\partial \max(x,y)}{\partial y} = \begin{cases} 1, \text{ if } x < y \\ 0, \text{ if } x > y \end{cases}$$

The $[\cdot]^+$ operator can be rewritten in terms of the max operator as $[x]^+ = \max(x, 0)$, proving its differentiability. Then, Equations 16 and 17, and in particular the delay bound in Equation 22 are differentiable w.r.t to the FIFO parameters. □

A note on the differentiability of the max operation: To avoid evaluation errors of the solver, it was necessary to define a derivative for the undefined case $\max(x, y)$ with $x = y$. We use

$$\left.\frac{\partial \max(x,y)}{\partial x}\right|_{y=x} = 0 \qquad \left.\frac{\partial \max(x,y)}{\partial y}\right|_{x=y} = 0$$

which is, strictly mathematically speaking, incorrect. However, every setting of θs leads to a valid delay bound and as we show



(a) Average delay bound deviation.

(b) CDF of most competitive NLP algorithms, Directed θ-Search for Feedforward Analysis (DS-FF) and LB-FF.

Figure 4: Delay bound deviations from LUDB-FF. SFA-FIFO, LB-FF, and Directed θ-Search for Feedforward Analysis (DS-FF) are proposed in [24]. Sequential Least Squares Quadratic Programming (SLSQP), SBPLX based on Subplex (SBPLX), SminS (SminS), and feedforward integrated SminS (SminS) (SminS$_{\text{FF-integrated}}$) are our most competitive NLP algorithms.

in our numerical evaluation, the quality of our derived parameter settings and delay bounds still vastly outperform other analyses.

4.4 The NLP LUDB Toolchain

The proposed NLP LUDB analysis is fully integrated into the NetworkCalculus.org Deterministic Network Calculator (NCorg DNC)[3]. Differentiation of the NC terms has been implemented in JAutoDiff[4], an open-source Java tool for automatic differentiation. For solving the non-linear optimization problem, we use the NLP library NLopt. NLopt offers a rich set of algorithms, including gradient-based ones. We interface using the open-source Java bindings for nlopt4j[5].

5 Numerical Evaluation

5.1 Evaluated Networks

For numerical evaluation, we used the set of randomly generated feedforward networks following the Erdős-Rényi model described in [24]. The dataset consists of 31 networks with a total of 4479 flows. The full dataset is available online[6].

5.2 Evaluation Setup

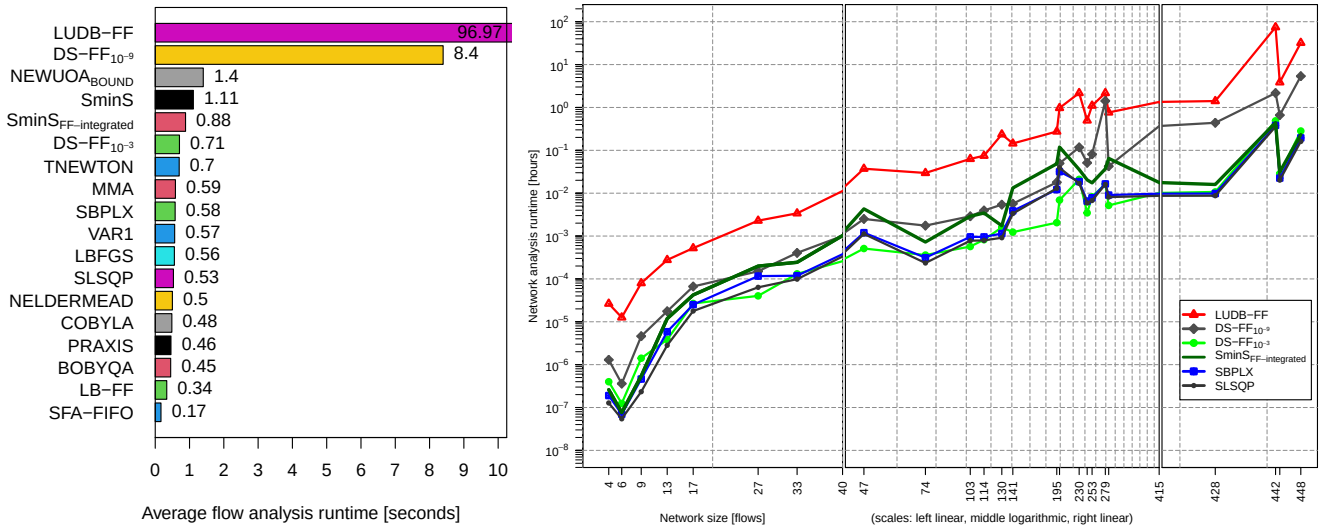
Extending the code base of the open-source NCorg DNC, see Section 4.4, allows us to provide benchmarks against the results shown in [24]. These are the search-based algorithm called DS-FF$_\epsilon$ with the recommended termination criterion alternatives $\epsilon \in \{10^{-3}, 10^{-9}\}$, as well as the SFA. All runtimes were measured on a Lenovo ThinkStation P620 with an AMD Ryzen Threadripper PRO 3955WX CPU clocked at (max) 4.30 GHz, running Ubuntu 22.04.1 LTS and OpenJDK 17.

5.3 Delay Bound Comparison

To find the optimal θ parameters, we need to solve the optimization problem resulting from each nested tandem in LUDB-FF (see Section 3.2, decompositions *a)* and *b)*). For that purpose we used the NLopt library, which provides several different algorithms for non-linear optimization, each yielding different results in terms of delay bound quality and runtime. We lay our focus on the most competitive algorithms:

(1) Sequential Least Squares Quadratic Programming (SLSQP) uses this algorithm to solve each optimization problem. It has already been found to work well for finding delay-bound-optimizing flow paths [14].
(2) SBPLX based on Subplex (SBPLX) [22] is a bound-constrained gradient-free optimization algorithm. Solving all optimization problems with SBPLX produced results comparable to SLSQP.
(3) Both above algorithms achieve comparable overall results, but either one may considerably beat the other for specific flow analyses. For each flow, we thus also present the minimum delay bound and the sum of computation times of both above analyses as SminS.
(4) We define another new combination of SLSQP and SBPLX, one that is integrated into the LUDB-FF feedforward analysis: feedforward integrated SminS (SminS$_{\text{FF-integrated}}$). This algorithm executes SminS on each nested tandem found in the feedforward network decomposition of LUDB-FF.

[3] dnc.networkcalculus.org, github.com/NetCal/DNC
[4] github.com/uniker9/JAutoDiff
[5] github.com/dibyendumajumdar/nlopt4j
[6] github.com/alexscheffler/dataset-rtns2022



(a) Average of all flow analyses. **(b) Total runtime for the analysis of all flows per network size.**

Figure 5: Runtime evaluation based on our analysis of 4479 flows across 31 networks.

We start with initial values of zero for all parameters. Each algorithm is set to terminate after an optimization step causes a relative change in the parameter value by less than 0.01%. To evaluate the quality of the delay bounds, we compare the bounds achieved with the algorithms mentioned above and some other algorithms from the NLopt library[7] to existing analyses, such as Separate Flow Analysis (SFA) under First-In First-Out (FIFO) assumptions (SFA-FIFO), Lower θ-Bound for Feedforward Analysis (LB-FF), DS-FF [24]. We measure the competitiveness of the achieved bounds as the deviation to the delay bounds obtained by an LUDB-FF analysis, defined by

$$delay_{\text{analysis,LUDB-FF}} = \frac{delay_{\text{analysis}} - delay_{\text{LUDB-FF}}}{delay_{\text{LUDB-FF}}} \tag{23}$$

Figure 4 shows that all of our most competitive algorithms, SLSQP, SBPLX, SminS, and $\text{SminS}_{\text{FF-integrated}}$, achieve equal or better results than the established analyses. SLSQP and SBPLX already achieve delay bounds that compare to $\text{DS-FF}_{10^{-9}}$, the most competitive of the established algorithms, at 0.44% and 0.45% average deviation to LUDB-FF, respectively. Their maximum deviation from LUDB-FF slightly exceeds that of $\text{DS-FF}_{10^{-9}}$, at 9.01% for SLSQP and 11.84% for SBPLX, compared to 8.56% for $\text{DS-FF}_{10^{-9}}$. Yet note, that [24] actually proposes to use $\text{DS-FF}_{10^{-3}}$ due to the dismal runtime performance of $\text{DS-FF}_{10^{-9}}$ as we will see later. SminS drastically improved the achieved delay bounds. It achieved an average deviation from LUDB-FF by 0.18%, about one third of the result from $\text{DS-FF}_{10^{-9}}$. Its maximum deviation of 4.11% lies at less than half of that of $\text{DS-FF}_{10^{-9}}$. Using our $\text{SminS}_{\text{FF-integrated}}$ algorithm, we could improve the delay bound quality even further, to 0.15% deviation to LUDB-FF on average and 4.02% at most.

Additionally, Figure 4b shows the ECDF of delay bound deviations. Our NLP algorithms all have a considerably larger share of bounds that deviate at most 0.1% from LUDB-FF. SminS approaches a density of 1 faster than SLSQP and SBPLX individually, showing the motivation behind executing both. $\text{SminS}_{\text{FF-integrated}}$ performs slightly better than SminS w.r.t. delay bound deviation.

[7] See nlopt.readthedocs.io/en/latest/NLopt_Algorithms/ for information about the algorithms.

5.4 Runtime Comparison

To explore the practicability of our analyses, we inspect their runtime. We focus on the runtime over an entire network analysis, that is the total time to bound all flows' end-to-end delay in one network size, and compare our results to LUDB-FF, LB-FF, and DS-FF.

First, we observe in Figure 5b that all our most competitive algorithms are consistently faster than LUDB-FF. Figure 5a shows that the average per-flow runtime of SLSQP and SBPLX also beat both DS-FF algorithms we chose for comparison, $\text{DS-FF}_{10^{-3}}$ and $\text{DS-FF}_{10^{-9}}$. Remember that they simultaneously achieved delay bounds that are comparable to $\text{DS-FF}_{10^{-9}}$. In LUDB-FF analyses, more than 95% of the total runtime was spent on the solver. The runtimes of SLSQP and SBPLX suggest that this is not the case anymore when using an NLP solver instead of the LP solver CPLEX. Unsurprisingly, SminS takes as long as SLSQP and SBPLX combined, as it analyses any given network with both algorithms and combines the results. Doing so, it runs about 87% faster than $\text{DS-FF}_{10^{-9}}$ and about 56% slower than $\text{DS-FF}_{10^{-3}}$. $\text{SminS}_{\text{FF-integrated}}$, our best algorithm in terms of the quality of the delay bounds, runs almost 90% faster than $\text{DS-FF}_{10^{-9}}$ and only about 24% slower than $\text{DS-FF}_{10^{-3}}$. The median relative runtime $\frac{\text{runtime}^{\text{LUDB-FF}}}{\text{runtime}^{\text{SminS}_{\text{FF-integrated}}}}$ of $\text{SminS}_{\text{FF-integrated}}$ w.r.t LUDB-FF is 110. $\text{SminS}_{\text{FF-integrated}}$ is therefore two orders of magnitude faster than LUDB-FF, whilst achieving delay bounds that are on average within a deviation of only 0.15% of LUDB-FF.

Note that while $\text{SminS}_{\text{FF-integrated}}$ and SminS each use both, SLSQP and SBPLX, to find the optimal θ settings, $\text{SminS}_{\text{FF-integrated}}$ is about 21% faster than SminS. This can be explained by a slight difference in the procedure. SminS runs two independent analyses for each flow, SLSQP and SBPLX, and compares the results at the end. This

way, some of the required analysis steps are executed twice: backtracking of flow dependencies, decomposition into nested tandems, the derivation of the optimization problems. $\text{SminS}_{\text{FF-integrated}}$, on the other hand, does all of this only once per analysis, i.e., it runs both, SLSQP and SBPLX, on the same optimization problem. The speedup of $\text{SminS}_{\text{FF-integrated}}$ compared to SminS is therefore not within the time to run the solver, but in the time it takes to derive the optimization problem.

6 Conclusion

In this paper, we tackle a main challenge within the (min,plus)-algebraic NC FIFO analysis called LUDB [2]. In the delay-bounding (min,plus)-algebraic term, there are free, interdependent parameters that need to be set efficiently and such that the analysis still gives an accurate delay bound. We show how to do so by non-linear programming, an idea derived from recent work on a different optimization problem within NC [14] called DiffNC. In order to do so, we derive a minimal closed-form solution of the delay-bounding term in (plus,times) algebra that is parameterized with the FIFO parameters mentioned above. This term can be used as the objective function in an optimization formulation in our NLP FIFO analysis. Additionally, we show how to differentiate the term in order to open up the opportunity to use gradient-based algorithms as they were used in DiffNC. We provide a new open-source toolchain for this analysis, allowing for benchmarking against existing heuristics for this problem [24] as well as the optimization that was proposed by the original authors. Compared to the heuristic recently recommended in [24], we reduce the gap to the optimal setting by a factor of 4.4, to 0.15% on average while only adding 24% to the computation time. This computation time is well below the one for finding the optimal setting.

6.1 Future Work

One of the contributions of our paper is the closed-form solution in (plus,times) algebra. Previous work either worked by decomposition into partial optimization problems or operated on the (min,plus)-algebraic NC term that cannot be used as the objective function in a standard solver. This contribution opens up a wide range of future work directions. Potential ways to extend the objective function include, for example, delay bound improving techniques

- decomposition, a.k.a. "cutting", of non-nested tandems into nested tandems (see Section 3.2, decomposition *b*)).
- flow prolongation that tightens delay bounds and was so far only feasible by using machine learning techniques [23].

as well as a full synthesis of the TSN 802.1Qbv scheduler configurations [26], i.e., the parameters of service-curve-defining priority FIFO queues and forwarding times, subject to bounds derived with an accurate LUDB FIFO analysis.

Last, we aim to extend our toolchain such that we can validate the DiffNC evaluation results.

Acknowledgements

We thank Fabien Geyer for his valuable feedback on an early draft of our work as well as our anonymous shepherd for his guidance.

References

[1] Luca Bisti, Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. 2008. Estimating the Worst-case Delay in FIFO Tandems Using Network Calculus. In *Proc. of the ICST International Conference on Performance Evaluation Methodologies and Tools (ValueTools)*.

[2] Luca Bisti, Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. 2012. Numerical Analysis of Worst-Case End-to-End Delay Bounds in FIFO Tandem Networks. *Springer Real-Time Systems Journal* 48, 5 (2012), 527–569.

[3] Steffen Bondorf, Paul Nikolaus, and Jens B. Schmitt. 2017. Quality and Cost of Deterministic Network Calculus – Design and Evaluation of an Accurate and Fast Analysis. *Proceedings of the ACM on Measurement and Analysis of Computing Systems (POMACS)* 1, 1 (2017), 16:1–16:34. ACM SIGMETRICS 2017 full papers.

[4] Steffen Bondorf and Jens B. Schmitt. 2015. Boosting Sensor Network Calculus by Thoroughly Bounding Cross-Traffic. In *Proc. of IEEE INFOCOM*.

[5] Steffen Bondorf and Jens B. Schmitt. 2016. Improving Cross-Traffic Bounds in Feed-Forward Networks – There is a Job for Everyone. In *Proc. of the 18th International GI/ITG Conference on Measurement, Modelling and Evaluation of Dependable Computer and Communication Systems (MMB & DFT 2016)*.

[6] Jean-Yves Le Boudec and Patrick Thiran. 2001. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer-Verlag.

[7] Anne Bouillard. 2022. Trade-off between accuracy and tractability of network calculus in FIFO networks. *Elsevier Performance Evaluation* 153 (2022), 102250.

[8] Anne Bouillard, Marc Boyer, and Euriell Le Corronc. 2018. *Deterministic Network Calculus: From Theory to Practical Implementation*. John Wiley & Sons, Ltd.

[9] Anne Bouillard and Giovanni Stea. 2015. Exact Worst-Case Delay in FIFO-Multiplexing Feed-Forward Networks. *IEEE/ACM Transactions on Networking* 23, 5 (2015), 1387–1400.

[10] Silviu S. Craciunas, Ramon Serna Oliver, Martin Chmelík, and Wilfried Steiner. 2016. Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks. In *Proc. of the 24th International Conference on Real-Time Networks and Systems*.

[11] Rene L. Cruz. 1991. A Calculus for Network Delay, Part I: Network Elements in Isolation" and "A Calculus for Network Delay, Part II: Network Analysis. *IEEE Trans. Inf. Theory* 37, 1 (1991), 114–141.

[12] Rene L. Cruz. 1998. SCED+: Efficient Management of Quality of Service Guarantees. In *Proc. of IEEE INFOCOM*.

[13] Markus Fidler. 2003. Extending the Network Calculus Pay Bursts Only Once Principle to Aggregate Scheduling. In *Proc. of the International Workshop on Quality of Service in Multiservice IP Networks (QoS-IP)*.

[14] Fabien Geyer and Steffen Bondorf. 2022. Network Synthesis under Delay Constraints: The Power of Network Calculus Differentiability. In *Proc. of IEEE INFOCOM*.

[15] Fabien Geyer and Steffen Bondorf. 2023. Differentiable Programming & Network Calculus: Configuration Synthesis under Delay Constraints. *arXiv:2307.14280 [cs.NI]*. https://arxiv.org/abs/2307.14280

[16] Fabien Geyer, Alexander Scheffler, and Steffen Bondorf. 2021. Tightening Network Calculus Delay Bounds by Predicting Flow Prolongations in the FIFO Analysis. In *Proc. of the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*.

[17] Kai Lampka, Steffen Bondorf, Jens B. Schmitt, Nan Guan, and Wang Yi. 2017. Generalized finitary real-time calculus. In *Proc. of IEEE INFOCOM*.

[18] Luciano Lenzini, Linda Martorini, Enzo Mingozzi, and Giovanni Stea. 2006. Tight end-to-end per-flow delay bounds in FIFO multiplexing sink-tree networks. *Performance Evaluation* 63, 9-10 (2006), 956–987.

[19] Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. 2007. End-to-End Delay Bounds in FIFO-Multiplexing Tandems. In *Proc. of the 2nd International Conference on Performance Evaluation Methodologies and Tools (ValueTools)*.

[20] Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. 2008. A methodology for computing end-to-end delay bounds in FIFO-multiplexing tandems. *Performance Evaluation* 65, 11-12 (2008), 922–943.

[21] Jörg Liebeherr. 2017. *Duality of the Max-Plus and Min-Plus Network Calculus*.

[22] Thomas Harvey Rowan. 1990. *Functional Stability Analysis of Numerical Algorithms*. Ph. D. Dissertation.

[23] Alexander Scheffler and Steffen Bondorf. 2021. Network Calculus for Bounding Delays in Feedforward Networks of FIFO Queueing Systems. In *Proc. of the 18th International Conference on Quantitative Evaluation of Systems (QEST 2021)*.

[24] Alexander Scheffler, Jens B. Schmitt, and Steffen Bondorf. 2022. Searching for Upper Delay Bounds in FIFO Multiplexing Feedforward Networks. In *Proc. of the 30th International Conference on Real-Time Networks and Systems (RTNS 2022)*.

[25] Jens Schmitt, Nicos Gollan, Steffen Bondorf, and Ivan Martinovic. 2011. Pay Bursts Only Once Holds for (Some) Non-FIFO Systems. In *Proc. of IEEE INFOCOM*.

[26] Luxi Zhao, Paul Pop, and Silviu S. Craciunas. 2018. Worst-Case Latency Analysis for IEEE 802.1Qbv Time Sensitive Networks Using Network Calculus. *IEEE Access* 6 (2018).

[27] Raffaele Zippo, Paul Nikolaus, and Giovanni Stea. 2023. Extending the network calculus algorithmic toolbox for ultimately pseudo-periodic functions: pseudo-inverse and composition. *Discrete Event Dynamic Systems* 33, 3 (2023).

A Step-by-step Analysis of an Example Network

Here, we briefly sketch the step-by-step process of deriving the delay bound in the network of Figure 1a. For the sake of simplicity, we limit our computation to only two FIFO parameters. We therefore assume that flow f_2 is zero. As such, the final computation step changes to

$$\begin{aligned}\beta_{foi}^{l.o.} &= \left(\beta_{f_1}^{l.o.} \ominus_{\theta_1} \alpha_1\right) \otimes \beta_{f_3}^{l.o.} \\ &= \left(\beta_{f_1}^{l.o.} \ominus_{\theta_1} \alpha_1\right) \otimes \left(\beta_{f_2}^{l.o.} \ominus_{\theta_2} \alpha_2\right)\end{aligned}$$

We will apply the computations in Theorem 3 and Theorems 4 to 7 following the computation steps in Figure 1b taking into account our additional assumption. We start by computing the left-over service, the foi experiences over its path from server 2 to server 3 considering crossflow f_2. To compute the left-over service, we apply Theorem 3 and Theorem 4, which is sketched in Figure 6a.

We then compute the left-over service available at server 1 considering crossflow f_1 and finally we compute the total left-over service available to the foi over its entire path. This is displayed in Figure 7b.

With the total left-over service available to the foi known, we can now search for the optimal delay bound and the corresponding optimal values of the introduced parameters θ_1 and θ_2. The delay bound w.r.t. to the introduced parameters is shown in Figure 8. Our strategy to solve this optimization problem is described in Section 5.

B Set Reduction of the FIFO Left-over Operation on Pseudoaffine Curves

In this section we will take a closer look at the FIFO left-over operations and why a subset of the set of valid left-over service curves can be excluded. In particular we will explore why we can substitute $s = \theta - hDev(\alpha, \pi), s \geq 0$ as our new FIFO parameter, thereby bounding θ by $hDev(\alpha, \pi)$ from below, as was done in Theorem 5. Note that this insight was first introduced in [18], the statement itself therefore constitutes no new contribution. However, we provide a more complete proof of it that is independent of the decomposition of the PLP into LPs.

To understand, why this excluded set of curves is irrelevant for delay bounding, we first need to understand how delay bounding works.

B.1 Delay Bounding

The delay bound $hDev(\alpha, \pi)$, see Theorem 2, is defined as the time it takes before an arriving flow is being processed. Mathematically, it can be seen as the horizontal deviation between the arrival curve at the point of its initial burst and the curve of the service offered to it. Assuming a token-bucket arrival curve $\alpha = \gamma_{\sigma,\rho}(t)$ and a pseudoaffine service curve $\pi(t)$, the delay bound $hDev(\alpha, \pi)$ can then be derived from solving $\pi(hDev(\alpha, \pi)) = \sigma$.

B.2 Left-over Service Curve Assuming FIFO Multiplexing

Assume flow f with arrival curve α is serviced by a server with service curve π. In the following we will assume the arrival curves to be a token-bucket curve $\alpha = \gamma_{\sigma,\rho}, \sigma, \rho \in \mathbb{R}^+$ and all service curves to be pseudoaffine curves of the form $\pi = \delta_D \otimes \left[\bigotimes_{1 \leq x \leq n} \gamma_{\sigma_x,\rho_x}\right], D, \sigma_x, \rho_x \in \mathbb{R}^+$. After servicing flow f, the server can offer only a reduced service. This service is described by the left-over service curve and can be computed from the initial service curve and flow f arrival curve α. However, f might arrive with a delay which is encoded in the FIFO parameter $\theta \in \mathbb{R}^+$.

Mathematically, we subtract α from π, but allow for α to be shifted to the right by θ. Note that the left-over service expressed by a service curve is lower bounded by zero. In this case, the server is still processing the queued-up data from flow f and cannot offer any left-over service. Similarly, the left-over service curve needs to be set to zero for $t < \theta$.

Due to the flexibility in setting θ to any positive real value, we end up with an infinite set of left-over service curves. Even though they are all valid, some of them are irrelevant for delay bounding. To show this, we will look at the left-over operation in detail and distinguish two cases:

(1) The burst σ of flow f arrival curve α exceeds the initial burst of the service curve:
$\sigma \geq \bigwedge_{1 \leq x \leq n} \sigma_x$.
(2) The burst σ of flow f arrival curve α is smaller than the initial burst of the service curve:
$\sigma < \bigwedge_{1 \leq x \leq n} \sigma_x$.

Each left-over service curve is again a pseudoaffine curve, which we will denote by

$$\begin{aligned}&\left[\pi^{l.o.}\right]_{bound_\theta} \\ &= \delta_{\left[D^{l.o.}\right]_{bound_\theta}} \otimes \left[\bigotimes_{1 \leq x \leq n} \gamma_{\left[\sigma^{l.o.}\right]_{bound_\theta},\left[\rho^{l.o.}\right]_{bound_\theta}}\right]\end{aligned} \tag{24}$$

where $bound_\theta$ refers to a bound on the value of θ.

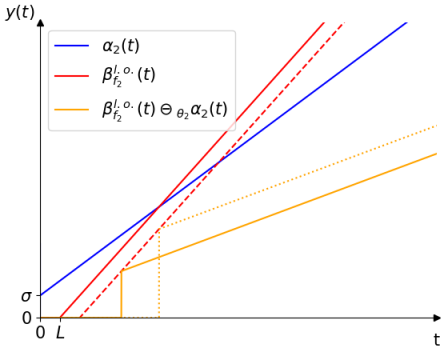
B.2.1 The burst σ of flow f arrival curve α exceeds the initial burst of the service curve. Both, the arrival curve α and the service curve π, start with an initial burst. However, the burst in α exceeds that of the service curve. In other words, flow f arrives with a burst of data that cannot be readily processed by the initial offered service. This only occurs at a later time, when the service has caught up with the initial burst, which is identical to the delay bound $hDev(\alpha, \pi)$. Thus, the left-over service inevitably gets delayed to at least this point. For reasons that will become evident later on, we will refer to this point as θ_{min}:

$$\theta_{min} = hDev(\alpha, \pi) = D + \bigvee_{1 \leq i \leq n} \left\lceil \frac{\sigma - \sigma_i}{\rho_i} \right\rceil^+ \tag{25}$$

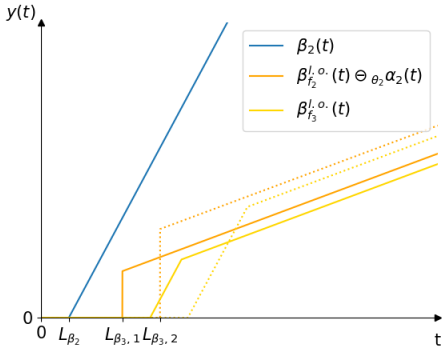
The shape of the left-over service curve π now depends on our choice of the FIFO parameter $\theta \in \mathbb{R}^+$ and we again distinguish two cases:

(1) Data of flow f arrives before the provided service meets its initial demand $\theta < \theta_{min}$. This notation might seem counter-intuitive, and we will see later that this case leads to irrelevant left-over service curves.
(2) The data starts arriving when its initial demand can already be met $\theta \geq \theta_{min}$.

Case a) Data of flow f arrives before the provided service meets its initial demand $\theta < \theta_{min}$: The server needs some time to catch up with the incoming data; left-over service can only be provided as

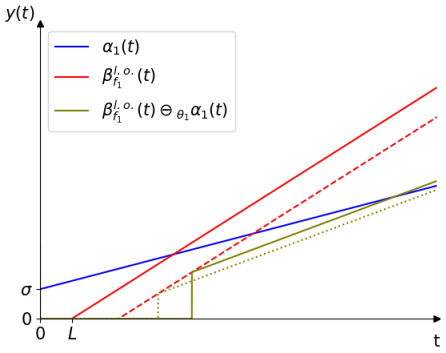


(a) Computing the left-over service at server 3 requires a left-over operation in Theorem 3 $\beta^{l.o.}_{f_2}$, i.e., removing crossflow f_2 arrival curve α_2. Under FIFO multiplexing, this will result in a set of left-over service curves, encoded by the indexed FIFO parameter θ_2. We focus on the left-over service curve shown by the solid orange curve. The dotted orange curve represents another curve from the set, for a different value of θ_2. Since the ideal value of θ_2 is not known at this stage, we continue with the entire set of left-over service curves.

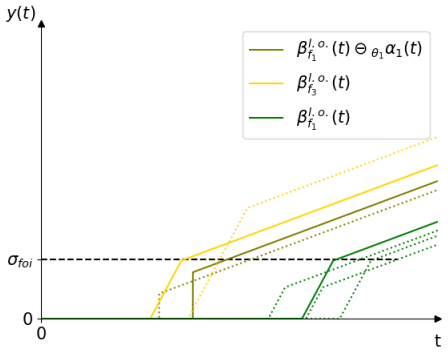


(b) To compute the service available to the foi across servers 2 and 3, we apply Theorem 4 to the service offered by server 2, β_2, and the left-over service at server 3 as a result of crossflow f_2. The result of the convolution operation is shown by the yellow curves. Again, we have plotted one resulting curve in a solid line and indicated the set of curves by including another curve plotted with a dotted line.

Figure 6: Computation of the left-over service curve for flow f_2 at servers 2 and 3 in the example network from Figure 1a.



(a) Analogously to Figure 6a, we apply Theorem 3 to determine the left-over service experienced by the foi at server 1 as a result of crossflow f_1.



(b) We can now compute the total service available to the foi. For this we once again apply Theorem 4 to the results of Figures 6b and 7a. The resulting left-over service (olive-colored graphs) depends on both parameters we introduced in the computation, θ_1 and θ_2.

Figure 7: Computation of the left-over service experienced by the foi over the example network from Figure 1a.

soon as data of flow f has been processed. The latency of the left-over service curve $\pi^{l.o.}$ is therefore determined by the intersection point $t_{intersect}$ of the arrival curve $\gamma_{\sigma,\rho}$ and the initial service curve π:

$$\begin{aligned}\left[D^{l.o.}\right]_{\theta<\theta_{min}} &= t_{intersect} \\ &= \max_{1\le x\le n}\left\{t_x : \gamma_{\sigma,\rho}(t_{intersect}-\theta) = \gamma_{\sigma_x,\rho_x}(t_{intersect}-D)\right\} \\ &= D + \bigvee_{1\le i\le n}\left\lceil \frac{\sigma+\rho(D-\theta)-\sigma_i}{\rho_i-\rho}\right\rceil \end{aligned} \tag{26}$$

The rates of the service curve decrease according to the arrival curve's rate ρ, as the server needs to process further incoming data:

$$\left[\rho^{l.o.}_x\right]_{\theta<\theta_{min}} = \rho_x - \rho, \forall x \in [1,n] \tag{27}$$

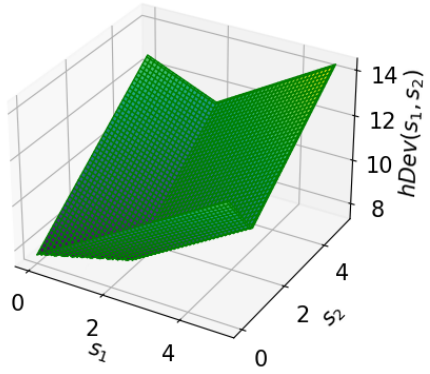


Figure 8: We can now use Theorem 7 to compute the delay bound. Since we introduced the FIFO parameters, θ_1 and θ_2, along the way, the solution now depends on these two parameters. We can find the optimal values of the parameters and subsequently the delay bound by the means described in Section 5.

The bursts $\sigma_x, \forall x \in [1,n]$ of the service curve stages decrease to

$$\begin{aligned}
&\left[\sigma_x^{l.o.}\right]_{\theta<\theta_{min}} \\
&= \gamma_{\sigma_x,\rho_x}\left(\left[D^{l.o.}\right]_{\theta<\theta_{min}} - D\right) - \gamma_{\sigma,\rho}\left(\left[D^{l.o.}\right]_{\theta<\theta_{min}} - \theta\right) && (28)\\
&= \sigma_x - \sigma + \rho\cdot\theta - \rho_x\cdot D + \\
&\quad (\rho_x-\rho)\left[D + \bigvee_{1\le i\le n}\left[\frac{\sigma+\rho(D-\theta)-\sigma_i}{\rho_i-\rho}\right]\right] && (29)\\
&= \rho_x\left\{\frac{\sigma_x-\sigma+\rho\cdot(\theta-D)}{\rho_x} + \left(1-\frac{\rho}{\rho_x}\right)\left[\bigvee_{1\le i\le n}\left[\frac{\sigma+\rho(D-\theta)-\sigma_i}{\rho_i-\rho}\right]\right]\right\} && (30)
\end{aligned}$$

The set of left-over service curves belonging to this case is then given by Equation (24) with $\theta < \theta_{min}$ as the bound on θ.

Case b) The data starts arriving when its initial demand can already be met $\theta \ge \theta_{min}$: In this case, the server can immediately provide ample service to process the demand by flow f. Left-over service is immediately provided at θ. The latency of the left-over service curve therefore is

$$\left[D^{l.o.}\right]_{\theta\ge\theta_{min}} = \theta \tag{31}$$

The rates decrease analogously to case a)

$$\left[\rho_x^{l.o.}\right]_{\theta\ge\theta_{min}} = \rho_x - \rho, \forall x \in [1,n] \tag{32}$$

The bursts of the service curve stages decrease by the arrival curve burst to

$$\left[\sigma_x^{l.o.}\right]_{\theta\ge\theta_{min}} = \gamma_{\sigma_x,\rho_x}(\theta - D) - \gamma_{\sigma,\rho}(0) = \rho_x(\theta-D) + \sigma_x - \sigma \tag{33}$$

This yields a set of left-over service curves $\left[\pi^{l.o.}\right]_{\theta\ge\theta_{min}}$ as specified by Equation (24).

We can reduce the set of relevant left-over service curves with the following theorem.

Theorem 9. *For every curve $\left[\pi^{l.o.}\right]_{\theta_a<\theta_{min}}$ resulting from case a) and an arbitrary $\theta_a \in \mathbb{R}^+$, there is a curve $\left[\pi^{l.o.}\right]_{\theta_b\ge\theta_{min}} \ge \left[\pi^{l.o.}\right]_{\theta_a<\theta_{min}}$ resulting from case b). This makes the set of curves from case a) inferior to that of case b) and therefore irrelevant for delay bounding.*

Proof. First note that since $\left[D^{l.o.}\right]_{\theta_a<\theta_{min}} \ge \theta_{min}$, and $\left[D^{l.o.}\right]_{\theta_b\ge\theta_{min}} = \theta_b \ge \theta_{min}$ we can compare each curve from case a) to a curve from case b) with an equal latency by setting $\theta_b = \left[D^{l.o.}\right]_{\theta_a<\theta_{min}}$.

Further note that the rates of the two curves are identical $\left[\rho_x^{l.o.}\right]_{\theta_a<\theta_{min}} = \left[\rho_x^{l.o.}\right]_{\theta_b\ge\theta_{min}} = \rho_x - \rho, \forall x \in [1,n]$.

The comparison of the two curves therefore depends on the comparison of their bursts. From Equation (28) and Equation (33) we see that

$$\begin{aligned}
&\left[\sigma_x^{l.o.}\right]_{\theta_a<\theta_{min}} \\
&= \gamma_{\sigma_x,\rho_x}\left(\left[D^{l.o.}\right]_{\theta_a<\theta_{min}} - D\right) - \gamma_{\sigma,\rho}\left(\left[D^{l.o.}\right]_{\theta_a<\theta_{min}} - \theta_a\right) && (34)\\
&\ge \gamma_{\sigma_x,\rho_x}\left(\left[D^{l.o.}\right]_{\theta_a<\theta_{min}} - D\right) - \gamma_{\sigma,\rho}(0) && (35)\\
&= \left[\sigma_x^{l.o.}\right]_{\theta_b\ge\theta_{min}}\Big|_{\theta_b=\left[D^{l.o.}\right]_{\theta_a<\theta_{min}}} && (36)
\end{aligned}$$

where the inequality follows from the property that $\gamma_{\sigma,\rho}$ is wide-sense increasing and that $\theta_a \in \mathbb{R}^+$. This proves the theorem. □

B.2.2 The burst σ of flow f arrival curve α is smaller than the initial burst of the service curve. Up until now, our proof is comparable to the proof of the set reduction for the left-over operation with rate-latency curves (Equation (11)) presented in [18]. This section describes a case that is not covered by this proof. We start by noting that in this case $\theta_{min} = D$, since $\sigma < \sigma_i, \forall 1 \le i \le n$. Then, the two cases identified in the previous section become

(1) $\theta < D$
(2) $\theta \geq D$

However, here case a) differs from the previous section in that we can define a new sub-case for it. The initial burst of flow f is small enough that it could be readily processed by the initial burst in the service. In particular circumstances, f could arrive before any service is provided without causing further latency to the left-over service. This is a sub-case of case a) from the previous section. We note that this case occurs when $\theta < \theta_{min} = D$ (the previous case a)), but additionally $\theta < \theta_0$ where we define θ_0 as the point where α exactly intersects with the point $(D, \bigwedge_{1 \leq i \leq n} \sigma_i)$. By setting $\gamma_{\sigma,\rho}(D - \theta_0) = \pi(D) = \bigwedge_{1 \leq i \leq n} \sigma_i$ we obtain an expression for θ_0:

$$\theta_0 = D + \frac{\sigma - \bigwedge_{1 \leq i \leq n} \sigma_i}{\rho} \tag{37}$$

Every $\theta \leq \theta_0$ yield a results analogous to case a) of the previous section. In the same way every $\theta \geq \theta_{min}$ yields a result analogous to case b) of the previous section. By Theorem 9 these curves following from $\theta \leq \theta_0$ are irrelevant for delay bounding.

Every $\theta_0 < \theta < \theta_{min} = D$, however, leads to a left-over service curve with unchanged latency. Since these curves are not covered by the previous section, we will derive them now and then show that they are also irrelevant for delay bounding. The latency of these curves is given by

$$\left[D^{l.o.}\right]_{\theta_0 < \theta < \theta_{min}} = D \tag{38}$$

The rates decrease analogously to the previous cases

$$\left[\rho_x^{l.o.}\right]_{\theta_0 < \theta < \theta_{min}} = \rho_x - \rho, \forall x \in [1, n] \tag{39}$$

The bursts of the left-over service curve stages decrease to

$$\left[\sigma_x^{l.o.}\right]_{\theta_0 < \theta < \theta_{min}} = \gamma_{\sigma_x,\rho_x}(0) - \gamma_{\sigma,\rho}(D - \theta) \tag{40}$$
$$= \sigma_x - \sigma - \rho_x(D - \theta) \tag{41}$$
$$< \sigma_x - \sigma \tag{42}$$

where the inequality follows from $\rho_x \in \mathbb{R}^+$ and $\theta < D$.

Theorem 10. *For every curve $\left[\pi^{l.o.}\right]_{\theta_0 < \theta_a < \theta_{min}}$ resulting from the sub-case of a) and an arbitrary $\theta_a \in \mathbb{R}^+$, there is a curve $\left[\pi^{l.o.}\right]_{\theta_b \geq \theta_{min}} \geq \left[\pi^{l.o.}\right]_{\theta_0 < \theta_a < \theta_{min}}$ resulting from case b). This makes the set of curves from the sub-case of a) inferior to that of case b) and therefore irrelevant for delay bounding.*

Proof. Consider the left-over service curve obtained from $\theta = D$ (case b)) and compare it to the curve derived above. The latencies and rates are identical. However, note that the bursts are larger for the curve derived from $\theta = D$. Using Equation (33) and Equation (42):

$$\left[\sigma_x^{l.o.}\right]_{\theta_b \geq \theta_{min}}\Big|_{\theta_b = D} = \gamma_{\sigma_x,\rho_x}(D - D) - \gamma_{\sigma,\rho}(0)$$
$$= \sigma_x - \sigma > \left[\sigma_x^{l.o.}\right]_{\theta_0 < \theta_a < D} \tag{43}$$

Therefore $\left[\pi^{l.o.}\right]_{\theta_b \geq \theta_{min}}\Big|_{\theta_b = D} \geq \left[\pi^{l.o.}\right]_{\theta_0 < \theta_a < \theta_{min}}$ □

From Theorem 9 and Theorem 10 we conclude that the the set of relevant left-over service curves is given by Equations (31) to (33). Now that we have established that we can discard all cases where $\theta < \theta_{min}$ we can rewrite θ as

$$\theta = \theta_{min} + s, s \in \mathbb{R}^+ \tag{44}$$

Then, we can rewrite the set of relevant left-over service curves using the new s parameter:

$$\left[D^{l.o.}\right]_{\theta \geq \theta_{min}} = \theta_{min} + s = hDev(\alpha, \pi) + s, s \in \mathbb{R}^+ \tag{45}$$
$$\left[\rho_x^{l.o.}\right]_{\theta \geq \theta_{min}} = \rho_x - \rho, \forall x \in [1, n] \tag{46}$$
$$\left[\sigma_x^{l.o.}\right]_{\theta \geq \theta_{min}} = \gamma_{\sigma_x,\rho_x}(\theta_{min} + s - D) - \gamma_{\sigma,\rho}(0) \tag{47}$$
$$= \rho_x \left\{ \bigvee_{1 \leq i \leq n} \left\lceil \frac{\sigma - \sigma_i}{\rho_i} \right\rceil^+ + s \right\} + \sigma_x - \sigma \tag{48}$$
$$= \rho_x \{s + hDev(\alpha, \pi) - D\} - (\sigma - \sigma_x) \tag{49}$$

thus bringing us to the notation from Theorem 5.