Searching for Upper Delay Bounds in FIFO Multiplexing Feedforward Networks

Alexander Scheffler
Faculty of Computer Science,
Ruhr University Bochum
Bochum, Germany

Jens Schmitt
Distributed Computer Systems Lab,
TU Kaiserslautern
Kaiserslautern, Germany

Steffen Bondorf
Faculty of Computer Science,
Ruhr University Bochum
Bochum, Germany

ABSTRACT

Network technologies are being developed to increase not only performance of data communication but also for provision of deterministic guarantees. Such designs foster the development of distributed real-time applications. When networks must provide time-sensitive communication, a commonly found design feature is First-In First-Out (FIFO) – a natural design for multiplexing different data flows into queues and for scheduling queued data. Alongside technological advancements, there is the development of formal tools to reason about timing behavior. Network Calculus is such a methodology. It has been widely adopted already and its modeling capabilities were extended to features found in modern standards as, e.g., in IEEE Time-Sensitive Networking. However, the basic challenge to compute a deterministic bound on a flow's worst-case end-to-end delay in FIFO networks still imposes challenges. Different analyses exist but often offer limited scalability. In this paper, we present an analysis with a readily tunable tradeoff between quality of the delay bound and the computational effort it imposes. We combine a greedy algorithm with an iterative directed search whose termination criterion can be adapted, e.g., subject to the analyzed network size. Our approach provides bounds of competitive quality at smaller computational cost than current alternatives for the analysis of feedforward FIFO networks.

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 multiplexing

ACM Reference Format:
Alexander Scheffler, Jens Schmitt, and Steffen Bondorf. 2022. Searching for Upper Delay Bounds in FIFO Multiplexing Feedforward Networks. In *Proceedings of the 30th International Conference on Real-Time Networks and Systems (RTNS '22), June 7–8, 2022, Paris, France.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3534879.3534894

RTNS '22, June 7–8, 2022, Paris, France

ACM ISBN 978-1-4503-9650-9/22/06.
https://doi.org/10.1145/3534879.3534894

1 INTRODUCTION

Motivation. Providing hard guarantees in communication systems becomes increasingly important as evident from the industrial demands leading to the development of standards like Avionics Full-Duplex Switched Ethernet (AFDX), IEEE Time-Sensitive Networking (TSN), IETF Deterministic Networking (DetNet) or Time-Triggered Ethernet (TTEthernet). A prominent such guarantee of interest is bounded latency, i.e., provision of a deterministic bound on the achievable end-to-end Worst-Case Delay (WCD) of a specific data flow. Any upper bound on the WCD is valid for that purpose, however, to avoid unnecessary overprovisioning of resources, the aim of any network analysis is to either derive the WCD itself or an accurate upper bound on it. An analysis is said to be tight if it computes the WCD. This usually comes at a considerable, if not forbidding, analysis complexity – often even boiling down to prohibitive computational complexity and analysis runtime. E.g., the tight analysis provided in [9] is an NP-hard Mixed-Integer Linear Program (MILP) formulation. Therefore, it may be preferable to relax the result accuracy in favor of significant runtime improvements to achieve a more suitable overall tradeoff between these two factors. A performance evaluation methodology that has the potential for such tuning is the Network Calculus (NC). NC has been applied to the above standards, e.g., AFDX [11], TSN [27] and TTEthernet [26]. These standards usually employ First-In First-Out (FIFO) multiplexing and forwarding in their queues. In this paper, we therefore aim to improve the accuracy that can be achieved with the runtime required for analyzing FIFO feedforward networks. To that end, we provide two new analyses based on the Least Upper Delay Bound (LUDB) for Feedforward Networks (LUDB-FF) [22].

Contribution. LUDB-FF creates a set of interdependent free parameters $\theta_i \in \mathbb{R}^+$ that need to be set for the derivation of the delay bound. Our first proposal for doing so, Lower θ-Bound for Feedforward Analysis (LB-FF), can be described as greedy in nature since it sets the θ_i directly without considering the interdependence of these parameters. While being fast, this naturally decreases delay bound accuracy. Our second analysis, Directed θ-Search for Feedforward Analysis (DS-FF), aims at exploring the interdependence of θ_i and thus closing the gap between LB-FF and LUDB-FF. As its name suggests, it employs a directed search on top of an initial setting for the θ_i – the setting derived with LB-FF. The accuracy of the delay bounds as well as the runtime of DS-FF can be easily tuned by the directed search's termination criterion ϵ. We will explore the tradeoff between accuracy and runtime for different DS-FF$_\epsilon$ in numerical evaluations. In short, we make the following contributions:

- We dissect the LUDB-FF to enable creation of a framework for multiple approaches setting free FIFO analysis parameters θ_i.

- We provide two new instantiations: LB-FF and DS-FF. While LUDB-FF optimizes the θ_i parameters, we aim at tunable, accurate NC analyses. LB-FF is greedy and DS-FF is search-based.
- We show in numerical evaluations that the accuracy of delay bounds is almost always better than the ones obtained from a classical server-by-server analysis and very competitive with LUDB-FF, even for rather large termination criteria ϵ.
- We further show that our runtimes for favorable ϵ settings is at least one order of magnitude below LUDB-FF and that DS-FF$_\epsilon$ scales better than the tight analysis from the literature.

The remainder of this paper is organized as follows: Section 2 presents the relevant NC basics and Section 3 dissects related analyses, in particular LUDB-FF, to generalize from them. Section 4 presents our contribution, the greedy LB-FF analysis and the search-based DS-FF analysis. Section 5 provides extensive benchmarks of the aforementioned analyses. Finally, Section 6 concludes the paper.

2 NETWORK CALCULUS BACKGROUND AND SYSTEM MODEL

Network Calculus Background. A thorough and accessible introduction on (min,plus)-algebraic NC performance modeling can be found in [6, 8].

Definition 1 (Cumulative Data Functions). *Suppose function A describes the cumulative input data arrival of a flow crossing a server $\mathcal{S}$. Then A' describes its cumulative output after $\mathcal{S}$. These functions are in the set $\mathcal{F}_0 := \{f : \mathbb{R}_\infty \to \mathbb{R}_\infty^+ | f(0) = 0 \; \forall s \le t : f(t) \ge f(s)\}$ with $\mathbb{R}_\infty = \mathbb{R} \cup \{\infty\}$ and $\mathbb{R}_\infty^+ = \mathbb{R}^+ \cup \{\infty\}$, respectively.*

Definition 2 (Arrival Curve). *Let A be the cumulative input function of a flow f at server $\mathcal{S}$. Then we call $\alpha \in \mathcal{F}_0$ an arrival curve of f at $\mathcal{S}$ if*

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

Definition 3 (Service Curve). *Suppose A is an input to a server $\mathcal{S}$ and A' is the respective output function. Then we say that $\beta \in \mathcal{F}_0$ is a service curve for $\mathcal{S}$ if*

$$\forall t \ge 0 : A'(t) \ge \inf_{0 \le s \le t} \{A(t-s) + \beta(s)\} =: A \otimes \beta(t).$$

Definition 4 (NC Operations). *The (min,plus)-algebraic aggregation, convolution and deconvolution of two functions $g, h \in \mathcal{F}_0$ are defined as*

$$\textit{aggregation: } (g+h)(d) = g(d) + h(d), \tag{1}$$

$$\textit{convolution: } (g \otimes h)(d) = \inf_{0 \le u \le d} \{g(d-u) + h(u)\}, \tag{2}$$

$$\textit{deconvolution: } (g \oslash h)(d) = \sup_{u \ge 0} \{g(d+u) - h(u)\}. \tag{3}$$

Theorem 1 (Performance Bounds). *Consider a server $\mathcal{S}$ that offers a service curve β. Assume flow f has arrival curve α. Then we obtain the following bounds:*

$$\textit{Output Bound: } \alpha'(t) = \alpha \oslash \beta(t) := \sup_{u \ge 0} \{\alpha(t+u) - \beta(u)\}$$

$$\textit{Delay Bound: } hDev(\alpha, \beta) = \inf\{d \ge 0 : (\alpha \oslash \beta)(-d) \le 0\}$$

where α' is a bound on A', the output from server $\mathcal{S}$ (see Definition 3), and the horizontal deviation $hDev$ between arrival curve and service curve bounds the delay experienced by f at $\mathcal{S}$.

Theorem 2 (Convolution of Service Curves). *Consider two servers in tandem $\mathcal{S}_1$ and $\mathcal{S}_2$ with service curves β_1 and β_2 respectively for flow f. Then, $\beta_1 \otimes \beta_2$ is a service curve for f for the system $(\mathcal{S}_1, \mathcal{S}_2)$.*

Theorem 3 (FIFO left-over Service). *Let server $\mathcal{S}$ offer service curve β. Assume flows f_1 and f_2 with arrival curves α_1 and α_2 cross $\mathcal{S}$. Assuming FIFO multiplexing, the left-over service for f_1 is*

$$\beta_{f_1}^{l.o.}(t) = [\beta(t) - \alpha_2(t-\theta)]^\uparrow \cdot \mathbf{1}_{\{t > \theta\}}$$

with $[g(x)]^\uparrow = \sup_{0 \le z \le x} g(z)$, the indicator function $\mathbf{1}_{\{condition\}}$ that is 0 if the condition is not met and 1 otherwise, and $\theta \in \mathbb{R}^+$ is the free FIFO parameter. As abbreviation for $[\beta(t) - \alpha_2(t-\theta)]^\uparrow \cdot \mathbf{1}_{\{t>\theta\}}$ we use $\beta \ominus_\theta \alpha_2$.

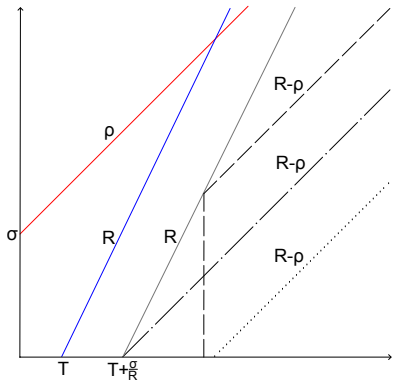

Figure 1: Arrival and service curve (for the aggregate) both colored on the left as well as the FIFO left-over curves with different setting of θ (dotted, dashed, dash-dotted line). Note that the curve with latency $T + \frac{\sigma}{R}$ and rate R is solely a helper curve to indicate the discontinuity of the dashed line. It is not a service curve itself.

Figure 1 depicts the influence of the choice of θ from Theorem 3 on the left-over service curve. Assume the crossflow has token-bucket arrival curve $\gamma_{\rho,\sigma}(t) = \{\sigma + \rho t\} \cdot \mathbf{1}_{\{t>0\}}$ and the service curve being a rate latency $\beta_{R,T}(t) = R \cdot [t-T]^\uparrow$. Setting $\theta = T + \frac{\sigma}{R}$, the left-over service curve is a rate latency curve with latency θ and rate $R - \rho$ (dash-dotted line). It is easy to see that any θ lower than $T + \frac{\sigma}{R}$ yields a worse curve for delay bounding (dotted line). Any setting of $\theta \ge T + \frac{\sigma}{R}$ (dashed line) yields a latency of θ, however the curve continues with $y = R \cdot (\theta - (T + \frac{\sigma}{R}))$. Hence, the curves for $\theta \ge T + \frac{\sigma}{R}$ are not directly comparable and it remains a highly complex task how to set this local value w.r.t. global delay bounds.

Another useful curve shape in NC is the so-called burst-delay function $\delta_T(t)$ that is 0 for $t \le T$ and ∞ otherwise. $\delta_0(t)$ is the neutral element w.r.t. operator $\otimes$.

System Model. Our approach is applicable to FIFO feedforward networks, i.e., the network can be depicted as a graph where nodes represent servers which are connected by links of increasing numbers. The latter makes sure that we have a cycle-free network, i.e., that the feedforward property holds. Each server i works off incoming data of different flows in a FIFO fashion and is constrained by a rate-latency service curve β_{R_i,T_i}. Each flow f has a known path

in the network and the data it inputs at its ingress server is constrained by a token-bucket arrival curve γ_{ρ_f,σ_f}. The token-bucket and rate-latency modeling is required due to a restriction of the LUDB methodology.

3 DISSECTING THE RELATED WORK

Our contribution in Section 4 is based on a lineage of NC achievements. Some of the related previous work on feedforward networks were presented as monolithic, i.e., multiple features were "packaged" into a single analysis. This view prevents further innovation. Therefore, we contribute a related work study that breaks up the monolithic analysis known as LUDB by dissecting it.

Total Flow Analysis (TFA). For completeness of the related work overview, consider the classic TFA that simply aggregates all flows, i.e., the analyzed flow of interest (foi) and its crossflows, for computation of delay and backlog bound at a server. Therefore, TFA is not faced with the challenge to set any θ as it does not make use of Theorem 3. While this derives one valid bound for each of the aggregated flows, it is not the WCD of any flow (in presence of crosstraffic). The inaccuracy grows when per-server delay bounds on the foi's path are summed up to a bound on its end-to-end delay.

3.1 Creating and setting the free parameter/s θ

At the basis of more recent (min,plus)-algebraic NC FIFO analyses is Theorem 3. As already shown in Section 2, visualized by Figure 1, setting θ is a non-trivial task, even for curves restricted to rate-latency and token-bucket shapes. When networks grow in complexity in terms of servers and flows, a multitude of $|F_x|$ interdependent θ-parameters, forming the vector $\Theta = (\theta_1, \ldots, \theta_{|F_x|})$, will have to be set. The actual size of Θ and how its free θ parameters interdepend is an intermediate result of the analysis applied to it.

3.1.1 Separate Flow Analysis under First-In First-Out (FIFO) assumptions (SFA-FIFO). The SFA-FIFO is a server-by-server analysis like TFA. Unlike TFA, it separates the foi from its crossflows by application of Theorem 3. I.e., SFA-FIFO creates at least one θ per server on a tandem, possibly more in case of complex interference patterns as to how flow paths overlap. Hence, for SFA-FIFO the number of FIFO parameters to set is at least equal to $|F_x|$. As a server-by-server analysis, the θ_i can be set in isolation as presented above, in order to compute a locally optimal delay or output bound. For a minimum delay bound, θ has to be chosen as $\theta = T + \frac{\sigma+\sigma_{\text{foi}}}{R}$. Regarding the output bound, [6], Corollary 6.2.2 (Burstiness Increase due to FIFO, General Case) discusses the single server case and shows that $\theta = T + \frac{\sigma}{R}$ results in the minimum output burstiness for token-bucket constrained arrivals and rate-latency service. For bounding the output of arrivals constained by (combinations of) token-bucket(s) crossing a single constant rate server ($\beta_{R,0}$), [12] even provides a tight output bound that may improve the one above without applying Theorem 3 at all. For this setting, [10] proves an output bound equal to applying Theorem 3 and setting $\theta = T + \frac{\sigma}{R}$.

However, the server-by-server proceeding of the analysis is inherently untight, as was first shown for SFA with Arbitrary Multiplexing (ARB MUX) [24], i.e., the SFA analysis whose result holds for any multiplexing behavior. The property increasing bound accuracy was named Pay Multiplexing Only Once (PMOO).

3.1.2 Least Upper Delay Bound (LUDB) [2]. [13, 14] propose a left-over service curve computation for entire tandems, i.e., end-to-end, however the resulting delay bound was shown to not be the WCD [20]. The improvement is achieved by applying the "convolution before subtraction" scheme, i.e., neighboring servers with the same crossflow interference are first convolved using Definition 4 before the left-over service curve is computed with Theorem 3. This is possible for nested interference on tandems (short "nested tandems").

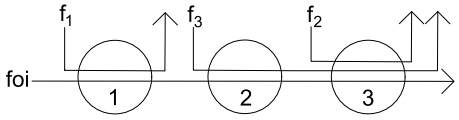

Figure 2: Nested tandem.

Definition 5 (Nested Interference [3]). *A tandem has nested interference iff for every pair of flows either both flows do not have common servers or the path of one flow is completely included in the path of the other. More formally, if we denote the crossed servers of a tandem by* $1, \ldots, N$ *then it is said to be nested iff there are no two flows* f_1, f_2 *with* $Source(f_1) < Source(f_2) \leq Dest(f_2) < Dest(f_1)$ *where the ordering relation* $\mathcal{S}_1 < \mathcal{S}_2$ *(*$\mathcal{S}_1 \leq \mathcal{S}_2$*) on a tandem of servers denotes that server* $\mathcal{S}_1$ *is a predecessor of* $\mathcal{S}_2$ *(or equal to it). Source and Dest are the given flow's source and destination server, respectively.*

For example, the network depicted in Figure 2 is a nested tandem as all crossflows are included in foi and f_2 is included in f_3.

The resulting order of (min,plus)-operations can be visualized in the according nesting tree, see Figure 3. A nesting tree [3] is a convenient data structure that captures the hierarchies of nested flows. Leaves of the tree correspond to the servers on a nested tandem. The remaining nodes represent flows and the need to derive a left-over service curve for that flow on its path. A flow-node representing f_2 is a child of flow-node representing f_3 iff the path of f_2 is fully contained in the path of f_3 and there is no other flow whose path is fully contained in f_3 and f_2. A leaf-node representing server $\mathcal{S}$ is a child of flow-node representing f_1 iff f_1 crosses $\mathcal{S}$ and there are no other flows crossing $\mathcal{S}$ whose paths are fully contained in the path of f_1.

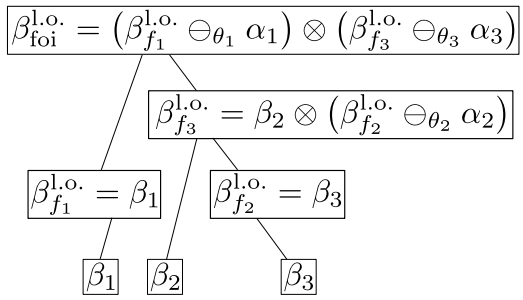

Figure 3: Nesting tree of nested tandem Figure 2. The folded out left-over service curve for the foi is of the form $(\beta_1 \ominus_{\theta_1} \alpha_1) \otimes ((\beta_2 \otimes (\beta_3 \ominus_{\theta_2} \alpha_2)) \ominus_{\theta_3} \alpha_3)$.

Table 1: Nomenclature on DNC and nesting tree analysis.

$\alpha_f^{\mathcal{S}}$	Input arrival curve of f at $\mathcal{S}$
$\beta_f^{\text{l.o.},p}$	Left-over service curve for f on path p
F, F_x	Set of flows, set of crossflows
foi, F_{foi}	foi, set of flows of interest
$S(\{f, F\})$	Set of servers flow f crosses, $S(F) = \bigcup_{f \in F} S(f)$
Children(i)	Child flows/servers of flow i in the nesting tree
$\Theta = (\theta_1, \ldots, \theta_{\lvert F_x \rvert})$	Vector of θ_i values for each crossflow i in the nesting tree

We may omit indices if they are clear from the context.

This sequential procedure of operations implements the PMOO property on the involved servers and for the involved flows.

The Least Upper Delay Bound (LUDB) analysis derives this tree to create Θ as well as the interdependencies between θ_i. I.e., LUDB is not a monolithic analysis, it rather consists of multiple steps towards derivation of delay bounds.

3.2 Dissecting LUDB-FF [22]

On a high level, LUDB-FF, the extension of LUDB for feedforward networks (details are in [22]) proceeds as follows:

Step 1 Take the foi's path as first tandem to analyze.
Step 2 Check for non-nested interference patterns, cut into nested tandems if necessary (is always possible).
Step 3 Compute the bounds on crossflow arrivals to the nested tandems by starting *Step 1* with them as F_{foi}.
Step 4 For each nested tandem, compute the nesting tree that encodes the "convolution before subtraction" scheme according to the nesting of flows. This creates the interdependent $\Theta(i)$, their interdependency is defined by the nesting (see Figure 3 for an example).
Step 5 Start the LUDB analysis for the nesting tree. The LUDB produces a Piecewise Linear Programm (PLP).
Step 6 For each linear decomposition of the PLP, one LP will be formulated and solved. The one with the minimal objective value (representing the delay bound) is the LUDB.

Previous work on LUDB was focused on analyzing tandems, i.e., *Step 3* was not detailed nor optimized. All the steps have been implemented in the NetworkCalculus.org Deterministic Network Calculator (NCorg DNC) [5] according to [22] that provides advanced code for *Step 3*, resulting in LUDB-FF. An important aspect of LUDB-FF is that it maximizes aggregation of flows. This reduces $|F_x|$ which, in turn, leads to less LPs to solve that will also have a smaller number of constraints due to aggregation. Nonetheless, this step can result in up to $O(|F_x|!)$ LPs which have to be solved. Note, that the authors of [3] give hints on how to reduce this number by computing the left-over curves bottom up in the nesting tree while already checking for infeasible constraints – such combinations can be safely skipped.

LUDB is restricted to token-bucket arrival curves and rate-latency service curves due to the conversion of Θ to the LP formulations in *Step 6*. Unfortunately, there is no obvious way to generalize this step to more complex curve shapes and LUDB-FF inherits the restriction.

Another advantage of the maximized aggregation is that it minimizes the potential sources of inaccuracies when setting the θs. Results of the solver employed in *Step 6* may not be optimal after all. This was exemplified for the closed-source IBM ILOG CPLEX Optimizer (CPLEX) in [23] and for open-source LpSolve in [15]. LUDB-FF makes use of CPLEX that is generally faster and more reliable than LpSolve. Yet, inaccuracies may build up with every θ.

This last observation raises the central question of our work: *Given this set of steps and the potential, unavoidable and uncontrollable inaccuracy in Step 6, can we achieve a better tradeoff between delay bound accuracy and computational effort by replacing it with a controllably inaccurate way to set* $\Theta = (\theta_1, \ldots, \theta_{|F_x|})$*?*

To that end, we have measured the share of CPLEX execution time w.r.t. the overall LUDB-FF analysis runtime in the networks presented in Section 5.1. Figure 4 shows the results. With only few exceptions, the share is above 95% of the overall analysis computation runtime, with the only significant drop still remaining above 80%.

In this paper, we contribute a search-based analysis with a controllable termination criterion to provide a significant reduction of analysis runtimes while not compromising much on the bound accuracy.

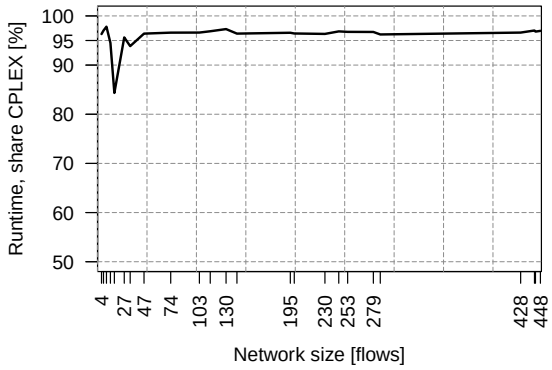

Figure 4: The share of overall computation times for solving the LPs generated within the LUDB-FF analysis.

3.3 Holistic Network Optimization [7, 9]

A tight analysis computing a flow's WCD in FIFO feedforward networks was recently proposed in [9]. It does not use (min,plus) algebra, instead it transforms the entire feedforward network as modeled by NC into a MILP formulation, with the objective set to computing the foi WCD. We call this analysis Feedforward Mixed-Integer Linear Programming Analysis (FF-MILPA). It does not follow the above steps and notably, it is therefore not taking *Step 4* where the free θ parameters are created in LUDB-FF. However, this comes at a high price: the MILP grows exponentially with the number of servers as it has to consider all flows dependencies – dependencies we consider in *Step 5* above but for the entire network. In practice, solving the FF-MILPA is only feasible for relatively small networks – which comes at no surprise as solving the LUDB-FF's

LPs is already very demanding (see Figure 4). Still, an LP heuristic was proposed for FF-MILPA, we call it Feedforward Linear Programming Analysis (FF-LPA). This heuristic computes an upper bound on the actual WCD, by a single LP. FF-LPA remains exponential in size, yet, we will include results we could compute in our numerical evaluation.

For a further improved tradeoff between accuracy and computational cost, the very recent work in [7] takes two steps: *a)* removing some LP constraints and *b)* adding new that were computed with (min,plus)-algebraic server-by-server analyses.

4 FINDING FIFO PARAMETERS $(\theta_1, \ldots, \theta_{|F_x|})$

We have seen that *Step 6* in the LUDB-FF procedure is a major contributor to the computational cost. Naturally, we want to tackle *Step 5* and *Step 6* to improve upon this aspect. In order to devise an appropriate alternative, we first have a closer look at the challenge of finding good values for $\Theta = (\theta_1, \ldots, \theta_{|F_x|})$ introduced in *Step 4*.

Figure 5 depicts the influence of the setting of (θ_1, θ_2) on an foi's delay bound w.r.t. the depicted network. We can observe that the problem space has a convex structure which is in favor of local optimization techniques such as directed search, the following main new analysis of our paper (DS-FF).

Remember that the nesting tree allows to easily identify the order in which to proceed the LUDB-FF analysis by going upwards in the tree and computing the left-over service for a flow-node based on the left-over service of its child-nodes and then storing it in the respective node. For example, Figure 3 depicts the nesting tree of the network Figure 2. Note that the approach creates a parameter θ per application of the FIFO left-over service curve computation Theorem 3. Hence, for a tandem with $|F_x|$ crossflows with pairwise distinct paths, we have $|F_x|$ many FIFO parameters to set. Crossflows with equal paths are aggregated. The values of these variables affect the computed performance bound, delay or output. We are searching for the "best" setting of these parameters w.r.t. the foi delay bound in a feedforward network. This (directed) search will improve upon a greedy algorithm's (initial) solution.

Table 2: Overview on the LB-FF and DS-FF nomenclature.

$\underline{\theta}_f(\beta_f^{\text{l.o.}}, \alpha_f)$	Lower bound on θ w.r.t. service curve $\beta_f^{\text{l.o.}}$ and arrival curve α_f
$\Theta^{\text{LB-FF}}$	$= (\theta_1^{\text{LB-FF}}, \ldots, \theta_{\lvert F_x\rvert}^{\text{LB-FF}})$, i.e., the vector of θ_i values derived by the LB-FF analysis
$\overline{\theta}_i(\Theta^{\text{LB-FF}})$	Upper bound on θ_i w.r.t. the Exploratory Phase of DS-FF if all the crossflows $F_x \ni j \neq i$ are set to $\theta_j^{\text{LB-FF}}$
ϵ	Termination criterion of the DS-FF analysis
sp_i	Step size of flow f_i in the DS-FF analysis
c	Parameter for the initial step size of all flows in the DS-FF analysis
ξ	Factor by which all step sizes get decreased if the Exploratory Phase of DS-FF analysis does not find a better delay bound

We may omit indices if they are clear from the context.

Algorithm 1 LB-FF computation on a nested tandem

Input i (Flow)-node of nesting tree
Output $\beta^{\text{l.o.}}$ left-over service curve for flow i

procedure COMPUTE-LB-LEFTOVERSERVICE(i)
  $\beta^{\text{l.o.}} \leftarrow \delta_0$
  $\forall c \in \text{Children}(i) \backslash F_x$:
    $\beta^{\text{l.o.}} \leftarrow \beta^{\text{l.o.}} \otimes \beta_c$
  $\forall c \in \text{Children}(i) \cap F_x$:
    $\beta_c^{\text{l.o.}} \leftarrow$ COMPUTE-LB-LEFTOVERSERVICE(c)
    $\theta_c \leftarrow \underline{\theta}(\beta_c^{\text{l.o.}}, \alpha_c)$
    $\beta^{\text{l.o.}}(t) \leftarrow \beta^{\text{l.o.}}(t) \otimes ([\beta_c^{\text{l.o.}}(t) - \alpha_c(t - \theta_c)]^{\uparrow} \cdot \mathbf{1}_{\{t > \theta_c\}})$
  return $\beta^{\text{l.o.}}$

4.1 Lower θ-Bound for Feedforward Analysis

Before presenting the greedy algorithm LB-FF we define a lower bound on the FIFO parameter θ (Theorem 3) which our proposed analyses heavily built upon.

DEFINITION 6 (LOWER BOUND ON θ). *Given an arrival curve $\alpha_f := \gamma_{\rho,\sigma}$ and left-over service curve $\beta_f^{l.o.}$, we define $\underline{\theta}_f(\beta_f^{l.o.}, \alpha_f) := \inf\{t \geq 0 : \beta_f^{l.o.}(t) \geq \sigma\}$. Note that $\underline{\theta}_f(\beta_f^{l.o.}, \alpha_f) = hDev(\alpha_f, \beta_f^{l.o.})$.*

An explanation why a lower value for θ is not beneficial can be found at the end of Section 2. Note that the setting of θ is equal to the approach of [13] – but only in case of nested tandems. Last, note that the resulting left-over service curve is a rate-latency curve (see Figure 1).

Next, we present LB-FF that is based on locally choosing a θ_i that matches Definition 6. More precisely, Algorithm 1 iteratively applies Definition 6 whenever a θ_i has to be chosen in the nesting tree which is traversed bottom-up. As an example take the network in Figure 2 and its nesting tree Figure 3 depicting the interdependency between the θ_i's. Applying Definition 6 throughout the nesting tree, we get $\Theta = (\theta_1, \theta_2, \theta_3)$,

$$\theta_1 = T_1 + \frac{\sigma_1}{R_1} \quad (4) \qquad \theta_2 = T_3 + \frac{\sigma_2}{R_3} \quad (5)$$

$$\begin{aligned} \theta_3 &= T_2 + \theta_2 + \frac{\sigma_3}{\min\{R_2, R_3 - \rho_2\}} \\ &= T_2 + T_3 + \frac{\sigma_2}{R_3} + \frac{\sigma_3}{\min\{R_2, R_3 - \rho_2\}} \end{aligned} \quad (6)$$

which results in the following left-over service curve for the foi:

$$\beta_{\text{foi}}^{\text{l.o.}} = \beta_{\substack{\min\{R_1-\rho_1, R_2-\rho_3, R_3-\rho_2-\rho_3\}, \\ T_1+T_2+T_3+\frac{\sigma_1}{R_1}+\frac{\sigma_2}{R_3}+\frac{\sigma_3}{\min\{R_2,R_3-\rho_2\}}}} \quad (7)$$

Although this greedy setting is not always minimizing delay bounds, computed bounds are valid such that LB-FF can be used as a standalone analysis. The benefit compared to LUDB-FF is the reduced runtime since the Θ is not optimized, at the cost of higher but still reasonable delay bounds (see our evaluation in Section 5). For improved delay bounds, we use the LB-FF-derived Θ setting as the starting point of our directed search.

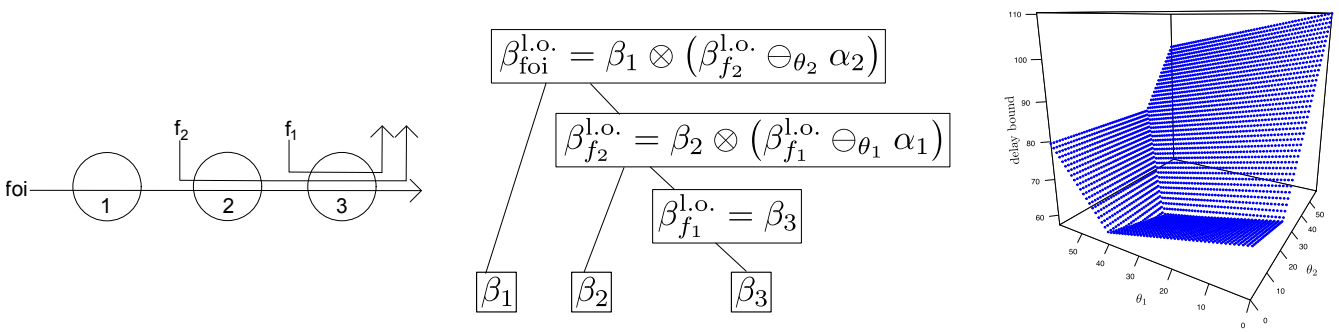

Figure 5: A tandem with two crossflows for illustration of the problem's shape (left) and its nesting tree visualizing the left-over service curve's computation with its interdependency between the two free parameters, $\Theta = (\theta_1, \theta_2)$ (middle). For each choice of $\Theta = (\theta_1, \theta_2)$, the delay bound $hDev(\alpha_{\textbf{foi}}, \beta^{\textbf{l.o.}}_{\textbf{foi}})$ gives one grid point in the plot (right).

4.2 Directed θ-Search for Feedforward Analysis

The directed search, also known as Hooke and Jeeves, is a well-known and understood local search method. Unfortunately, it may find its solution in the nearest local minimum [21] instead of the flobal minimum. Proving that the problem to set the $(\theta_1, \ldots, \theta_{|F_x|})$ for any nesting tree is indeed convex, is out of scope of our work. Our later numerical observation does, however, show that our approach to start from an LB-FF-derived setting results in a resilient DS-FF analysis.

The basic idea of our search scheme is: given an initial combination of the variables, search steps will 1) move each variable, one at a time, according to a variable-specific step size and 2) save the best result, to 3) combine the moves of all the variables. The search continues in the "combined" direction until no significant progress is made anymore. Then, the step size of each variable gets decreased. Termination is triggered when at least one variable's step size becomes smaller than a pre-defined threshold $\epsilon \in \mathbb{R}^+$. In the following we show our adaptation of this local search in the context of the NC FIFO analysis, abbreviated as DS-FF.

Algorithm 2 depicts how to obtain the left-over service curve for the foi: given a setting for the FIFO parameter per crossflow on a tandem by going through the nesting tree and computing the left-over service curves. In essence, it works very similar to Algorithm 1 but with a flexible parameter combination in contrast to the fixed setting for θ based on Definition 6.

The actual search is presented in Algorithm 3. First, a base combination of θs and its respective delay bound has to be found and provided. We use here the combination derived by LB-FF denoted by $\Theta^{\text{LB-FF}} := (\theta_1^{\text{LB-FF}}, \ldots, \theta_{|F_x|}^{\text{LB-FF}})$. Moreover, the (initial) step sizes denoted by sp_i (for crossflow f_i) have to be chosen. To incorporate the nesting hierarchy of crossflows for reasonable starting step sizes, we choose the following setting:

$$sp_i = \frac{\overline{\theta}_i(\Theta^{\text{LB-FF}}) - \theta_i^{\text{LB-FF}}}{c - 1} \tag{8}$$

with $c \in \mathbb{N}^+_{\geq 2}$. For $\overline{\theta}_i(\Theta^{\text{LB-FF}})$ we differentiate between two nesting scenarios:

Algorithm 2 Left-over service computation on a nested tandem given a parameter combination

Input i, Θ (Flow)-node of nesting tree, parameter combination
Output $\beta^{\text{l.o.}}$ left-over service curve for flow i

1: **procedure** computeLeftOverService(i, Θ)
2: $\quad\beta^{\text{l.o.}} \leftarrow \delta_0$
3: $\quad\forall c \in \text{Children}(i) \backslash F_x$:
4: $\quad\quad\beta^{\text{l.o.}} \leftarrow \beta^{\text{l.o.}} \otimes \beta_c$
5: $\quad\forall c \in \text{Children}(i) \cap F_x$:
6: $\quad\quad\beta^{\text{l.o.}}_c \leftarrow$ computeLeftOverService(c, Θ)
7: $\quad\quad\theta_c \leftarrow \Theta(c)$
8: $\quad\quad\beta^{\text{l.o.}}(t) \leftarrow \beta^{\text{l.o.}}(t) \otimes ([\beta^{\text{l.o.}}_c(t) - \alpha_c(t - \theta_c)]^{\uparrow} \cdot \mathbf{1}_{\{t > \theta_c\}})$
9: $\quad$return $\beta^{\text{l.o.}}$

i) $f_i \in$ Children(foi)

$$\overline{\theta}_i(\Theta^{\text{LB-FF}}) := d^{\text{LB-FF}} - \sum_{k \in S(\text{foi}) \backslash S(\text{Children(foi)})} T_k - \sum_{f_i \neq f_k \in \text{Children(foi)}} \theta_k^{\text{LB-FF}}$$

ii) $f_i \notin$ Children(foi). Let $f_p \in F_x$ s.t. $f_i \in$ Children(f_p), then

$$\overline{\theta}_i(\Theta^{\text{LB-FF}}) := \overline{\theta}_p(\Theta^{\text{LB-FF}}) - \sum_{k \in S(f_p) \backslash S(\text{Children}(f_p))} T_k - \sum_{f_i \neq f_k \in \text{Children}(f_p)} \theta_k^{\text{LB-FF}}$$

For example w.r.t. the network in Figure 2 we get

$$\overline{\theta}_1(\Theta^{\text{LB-FF}}) = d^{\text{LB-FF}} - \theta_3^{\text{LB-FF}} \qquad \overline{\theta}_2(\Theta^{\text{LB-FF}}) = \overline{\theta}_3(\Theta^{\text{LB-FF}}) - T_2$$
$$\overline{\theta}_3(\Theta^{\text{LB-FF}}) = d^{\text{LB-FF}} - \theta_1^{\text{LB-FF}}$$

Consider the combination $\Theta^{\text{LB-FF}}$ and fix one crossflow f_i, then $(\theta_1^{\text{LB-FF}}, \ldots, \theta_{i-1}^{\text{LB-FF}}, \overline{\theta}_i(\Theta^{\text{LB-FF}}), \theta_{i+1}^{\text{LB-FF}}, \ldots, \theta_{|F_x|}^{\text{LB-FF}})$ will result in a worse delay bound. For the example above use $\Theta = (\overline{\theta}_1(\Theta^{\text{LB-FF}}), \theta_2^{\text{LB-FF}}, \theta_3^{\text{LB-FF}})$. Then one can show with Theorem 3 and Theorem 2 that the left-over service curve for the foi has a latency that is at least $\overline{\theta}_1(\Theta^{\text{LB-FF}}) + \theta_3^{\text{LB-FF}} = d^{\text{LB-FF}} - \theta_3^{\text{LB-FF}} + \theta_3^{\text{LB-FF}} =$

Algorithm 3 DS search algorithm on a nested tandem

Input Θ, d parameter combination, delay bound (foi)
Output $\beta^{\text{l.o.}}$ left-over service curve for foi

1: **procedure** DS(Θ, d)
2: $d^{\text{old}} \leftarrow d$
3: $\Theta^{\text{old}} \leftarrow \Theta$
4: $\Theta^{\text{best}} \leftarrow \Theta$
5: **while** $\min_{i=1,..,|F_x|}\{sp_i\} \geq \epsilon$ **do**
6: $(d^{\text{new}}, \Theta^{\text{new}}) \leftarrow$ FindBestNearby($\Theta^{\text{old}}, d^{\text{old}}$)
7: $improved \leftarrow false$
8: **while** $d^{\text{new}} < d^{\text{old}}$ **do**
9: $improved \leftarrow true$
10: $\Theta^{\text{best}} \leftarrow \Theta^{\text{new}}$
11: $\Theta^{\text{pot}} \leftarrow \Theta^{\text{new}} + (\Theta^{\text{new}} - \Theta^{\text{old}})$
12: $d^{\text{old}} \leftarrow d^{\text{new}}$
13: $\Theta^{\text{old}} \leftarrow \Theta^{\text{new}}$
14: $\beta^{\text{l.o.}} \leftarrow$ computeLeftOverService(foi, Θ^{pot})
15: $d^{\text{pot}} \leftarrow hDev(\alpha_{\text{foi}}, \beta^{\text{l.o.}})$
16: **if** $d^{\text{pot}} < d^{\text{new}}$ **then**
17: $d^{\text{new}} \leftarrow d^{\text{pot}}$
18: $\Theta^{\text{new}} \leftarrow \Theta^{\text{pot}}$
19: **if** !$improved$ **then**
20: $\forall i \in \{1, \ldots, |F_x|\} : sp_i \leftarrow sp_i \cdot \xi$
21: $\beta^{\text{l.o.}} \leftarrow$ computeLeftOverService(foi, Θ^{best})
22: return $\beta^{\text{l.o.}}$

$d^{\text{LB-FF}}$ which is a Θ combination we can safely skip since it will deliver a delay bound that won't improve upon $d^{\text{LB-FF}}$. Similarly, one can show that neither $\Theta = (\theta_1^{\text{LB-FF}}, \overline{\theta}_2(\Theta^{\text{LB-FF}}), \theta_3^{\text{LB-FF}})$ nor $\Theta = (\theta_1^{\text{LB-FF}}, \theta_2^{\text{LB-FF}}, \overline{\theta}_3(\Theta^{\text{LB-FF}}))$ will result in an improved bound upon $d^{\text{LB-FF}}$. Hence, we start with the respective fraction.

The actual directed search consists of two phases: the Exploratory Phase (Algorithm 4) and the Pattern Move Phase (Algorithm 3, lines 8 to 18). During the Exploratory Phase, it checks for each crossflow f_i in isolation if a lower or higher setting of the variable θ_i w.r.t. the current combination and step size yields a better delay bound. If this is the case, the search continues into that direction in the subsequent Pattern Move Phase. This second phase can be sped up by using the Armijo line search [21], i.e., instead of testing for $\Theta^{\text{start}} + \Delta, \Theta^{\text{start}} + 2 \cdot \Delta, \Theta^{\text{start}} + 3 \cdot \Delta, \ldots$ with $\Delta = \Theta^{\text{new}} - \Theta^{\text{old}}$ (see line 11), it tests for $\Theta^{\text{start}} + 2^0 \cdot \Delta, \Theta^{\text{start}} + 2^1 \cdot \Delta, \Theta^{\text{start}} + 2^2 \cdot \Delta, \ldots$. We use Armijo line search in DS-FF. In the other case, i.e., if a better delay bound is not found during the Exploratory Phase, then all step sizes are decreased by $0 < \xi < 1$ (with $\xi \in \mathbb{R}^+$). The search terminates if the smallest step size (over all crossflows) becomes smaller than a pre-defined threshold $\epsilon > 0$ (with $\epsilon \in \mathbb{R}^+$). This is where we can tune the effort of the directed search: a smaller ϵ potentially yields a more accurate delay bound but will do so by more search steps and thus larger computational costs.

Moreover, before trying a combination, i.e., before calling computeLeftOverService(foi, Θ), it can be checked if any $\theta \in \Theta$ violates the constraints $\theta \geq 0$ (has to hold, see Theorem 3) and $\theta < d$ with d being the lowest delay bound so far found by the search. It can be shown that violating the latter bound automatically yields a worse delay bound than d. Additionally, to save computation cost during computeLeftOverService, DS-FF works with one current nesting tree in order to reuse the left-over service curves of flows that are currently not influenced by a change of a certain parameter setting of some crossflow. For brevity, we omit the details here.

Algorithm 4 Exploratory Phase of the DS algorithm

Input Θ, d parameter combination, delay bound (foi)
Output (d, Θ) lowest delay bound in current environment, respective parameter combination

1: **procedure** FindBestNearby(Θ, d)
2: $\Theta^{\text{new}} \leftarrow \Theta$
3: $d^{\text{new}} \leftarrow d$
4: **for** $i = 1$ to $|F_x|$ **do**
5: $\Theta^{\text{low}} \leftarrow (\theta_1^{\text{new}}, \ldots, \theta_{i-1}^{\text{new}}, \theta_i^{\text{new}} - sp_i, \theta_{i+1}^{\text{new}}, \ldots, \theta_{|F_x|}^{\text{new}})$
6: $\beta^{\text{l.o.}} \leftarrow$ computeLeftOverService(foi, Θ^{low})
7: $d^{\text{low}} \leftarrow hDev(\alpha_{\text{foi}}, \beta^{\text{l.o.}})$
8: $\Theta^{\text{high}} \leftarrow (\theta_1^{\text{new}}, \ldots, \theta_{i-1}^{\text{new}}, \theta_i^{\text{new}} + sp_i, \theta_{i+1}^{\text{new}}, \ldots, \theta_{|F_x|}^{\text{new}})$
9: $\beta^{\text{l.o.}} \leftarrow$ computeLeftOverService(foi, Θ^{high})
10: $d^{\text{high}} \leftarrow hDev(\alpha_{\text{foi}}, \beta^{\text{l.o.}})$
11: $d^{\text{min}} \leftarrow \min\{d^{\text{new}}, d^{\text{low}}, d^{\text{high}}\}$
12: Update $d^{\text{new}}, \Theta^{\text{new}}$ according to d^{min}
13: return $(d^{\text{new}}, \Theta^{\text{new}})$

Note that DS-FF shares the "-FF" suffix with LUDB-FF as it is embedded into the same feedforward analysis presented in [22]. Regarding the effort of the presented algorithms, this means that for a nested tandem both analyses will work with the same number of θ's, i.e., $|F_x|$, which is minimal. For simplicity, consider a single server with an foi and two crossflows. There are two alternatives to compute the left-over service curve (1) consecutive application of Theorem 3, removing the impact of crossflows in an arbitrary order and (2) aggregate the crossflows first (Definition 4) and then remove the impact of the crossflow aggregate. In theory, the left-over service curve of either alternative results in the same bounds. In practice, however, the consecutive application of Theorem 3 will introduce more free θ parameters, will increase the size of Θ without any benefit. The effort of all analyses will grow with the size of Θ, for DS-FF see the simultaneously considered free θ parameters in Algorithm 3. For more details about the aggregation of crossflows in a feedforward analysis, we refer the reader to [22].

Example 1 (Execution of DS-FF Algorithms 3 and 4). As an example we unroll the DS-FF algorithm for the network Figure 2. That is, we compute $\Theta^{\text{LB-FF}}$ (see end of Section 4.1). We use $c = 5$ (as in our evaluation) to compute sp_i according to Equation (8) for $i \in \{1, 2, 3\}$. The call to DS($\Theta^{\text{LB-FF}}, d^{\text{LB-FF}}$) (Algorithm 3) then proceeds as follows:

Exploratory Phase (Algorithm 4), $i = 1$

- Compute delay bound d^{low} for
$$\Theta^{\text{low}} = (\theta_1^{\text{LB-FF}} - sp_1, \theta_2^{\text{LB-FF}}, \theta_3^{\text{LB-FF}})$$
- Compute delay bound d^{high} for
$$\Theta^{\text{high}} = (\theta_1^{\text{LB-FF}} + sp_1, \theta_2^{\text{LB-FF}}, \theta_3^{\text{LB-FF}})$$
- Assume $d^{\text{high}} < d^{\text{new}} = d^{\text{LB-FF}} < d^{\text{low}}$
- Update $d^{\text{new}} = d^{\text{high}}, \Theta^{\text{new}} = \Theta^{\text{high}}$

Exploratory Phase (Algorithm 4), $i = 2$

- Compute delay bound d^{low} for
$$\Theta^{\text{low}} = (\theta_1^{\text{LB-FF}} + sp_1, \theta_2^{\text{LB-FF}} - sp_2, \theta_3^{\text{LB-FF}})$$
- Compute delay bound d^{high} for
$$\Theta^{\text{high}} = (\theta_1^{\text{LB-FF}} + sp_1, \theta_2^{\text{LB-FF}} + sp_2, \theta_3^{\text{LB-FF}})$$
- Assume $d^{\text{new}} < d^{\text{high}} < d^{\text{low}}$ ($\rightarrow$ no update)

Exploratory Phase (Algorithm 4), $i = 3$

- Compute delay bound d^{low} for
$$\Theta^{\text{low}} = (\theta_1^{\text{LB-FF}} + sp_1, \theta_2^{\text{LB-FF}}, \theta_3^{\text{LB-FF}} - sp_3)$$
- Compute delay bound d^{high} for
$$\Theta^{\text{high}} = (\theta_1^{\text{LB-FF}} + sp_1, \theta_2^{\text{LB-FF}}, \theta_3^{\text{LB-FF}} + sp_3)$$
- Assume $d^{\text{high}} < d^{\text{new}} < d^{\text{low}}$
- Update $d^{\text{new}} = d^{\text{high}}, \Theta^{\text{new}} = \Theta^{\text{high}}$

Pattern Move Phase (Algorithm 3, lines 8 to 18)

- $\Theta^{\text{old}} = (\theta_1^{\text{LB-FF}}, \theta_2^{\text{LB-FF}}, \theta_3^{\text{LB-FF}})$
- $\Theta^{\text{new}} = (\theta_1^{\text{LB-FF}} + sp_1, \theta_2^{\text{LB-FF}}, \theta_3^{\text{LB-FF}} + sp_3)$
- Compute delay bound d^{pot} for
$$\Theta^{\text{pot}} = (\theta_1^{\text{LB-FF}} + 2 \cdot sp_1, \theta_2^{\text{LB-FF}}, \theta_3^{\text{LB-FF}} + 2 \cdot sp_3)$$
- Assume $d^{\text{pot}} < d^{\text{new}}$
- Update $d^{\text{new}} = d^{\text{pot}}, \Theta^{\text{new}} = \Theta^{\text{pot}}$
- Compute delay bound d^{pot} for
$$\Theta^{\text{pot}} = (\theta_1^{\text{LB-FF}} + 3 \cdot sp_1, \theta_2^{\text{LB-FF}}, \theta_3^{\text{LB-FF}} + 3 \cdot sp_3)$$
  - in case of Armijo line search
$$\Theta^{\text{pot}} = (\theta_1^{\text{LB-FF}} + 4 \cdot sp_1, \theta_2^{\text{LB-FF}}, \theta_3^{\text{LB-FF}} + 4 \cdot sp_3)$$
- Assume $d^{\text{pot}} \geq d^{\text{new}}$

Exploratory Phase (Algorithm 4), $i = 1, 2, 3$

- Assume that the considered Θ combinations don't improve the delay bound

Decrease all stepsizes: $sp_i = sp_i \cdot 0.5$ ($\xi = 0.5$ as in our evaluation) Assume $sp_1 < \epsilon$, then the DS algorithm terminates with the best found service curve, i.e., the one computed with the Θ setting $(\theta_1^{\text{LB-FF}} + 2 \cdot sp_1, \theta_2^{\text{LB-FF}}, \theta_3^{\text{LB-FF}} + 2 \cdot sp_3)$

A Note on Generalizability. As mentioned in Section 3.2, LUDB is restricted to token-bucket arrival curves and rate-latency service curves due to the conversion of Θ to LP formulations. As we replace this step in our search approaches, we can, in principle, generalize our analysis to more complex curve shapes such as stair-case functions. This might, however, impact the shape of the problem. E.g., its convexity might be lost due to local minima that hinder the progress of DS-FF. In that case, it is also possible to add some random additional move between LB-FF and DS-FF. Our improved runtimes allow for multiple (re-)starts of DS-FF with different seeds for the intermediate random step to reduce the impact of local minima. The same approach was recently used in [17].

5 NUMERICAL EVALUATION

5.1 Network Generation for Evaluations

For numerical evaluation, we generate random feedforward networks following the Erdős-Rényi model. The process is as follows: First, we create a random undirected Erdős-Rényi graph, $\mathcal{G} = G(n, p)$ with n being the number of nodes, and p the probability that a random pair of nodes $\{x, y\}, x \neq y$ has an undirected edge between them. We pick n uniformly at random between 10 and 25 and set the probability $p = 0.1$. Then we find the biggest component in $\mathcal{G}$, i.e., the component with the most number of nodes. If there is more than one, we pick a random one among them. All other components will be deleted from $\mathcal{G}$. This will result in a (possibly reduced) connected undirected graph $\mathcal{G}^{\{c\}} = (V^{\{c\}}, E^{\{c\}})$. For conversion to a feedforward NC network, we first run the full turn prohibition algorithm [25] on $\mathcal{G}^{\{c\}}$ which returns the set of prohibited turns. Turns are pairs of edges $\{a, b\}, \{x, y\} \in E^{\{c\}}$ such that b equals x. Next, we create another representation of $\mathcal{G}^{\{c\}}$ denoted by $\mathcal{G}^{\{c,d,f\}}$. As NC analyzes the behavior of (sequences of) queueing locations, we encode them in $\mathcal{G}^{\{c,d,f\}}$. I.e., a node in $\mathcal{G}^{\{c,d,f\}}$ is created from the pair consisting of a node and its outgoing edge, both from $\mathcal{G}^{\{c\}}$. The not prohibited turns of $\mathcal{G}^{\{c\}}$ will be the (directed) edges in $\mathcal{G}^{\{c,d,f\}}$. Hence, $\mathcal{G}^{\{c,d,f\}}$ will then be a directed graph (d) which is free of cycles, i.e., satisfies the feedforward property (f). Next, we add flows to $\mathcal{G}^{\{c,d,f\}}$ in order to make it the NC network to analyze. We consider all possible paths from $\mathcal{G}^{\{c,d,f\}}$ and pick randomly $|Paths| \cdot density$ many distinct paths for $density = 30\%$. For each selected path we create one flow.

Finally, as vertices in $\mathcal{G}^{\{c,d,f\}}$ represent servers, we set service curves for all elements in $V^{\{c,d,f\}}$, and arrival curves for all flows. We strive for the ability to define the utilization at servers since we want to study the effect of the FIFO property throughout different NC analyses that consider this property to different extents. For the arrival curves, we set token-bucket constraints $\gamma_{\rho,\sigma}$ with burstiness $\sigma = 1$ and rate $\rho = 1$. Service curves are created as rate-latency curves $\beta_{R,T}$ with latency $T = 0$ and a rate R according to a uniformly at random selected utilization between 50% and 99%, independently at each server. We selected latency $T = 0$ since the analyses we contribute in this paper make use of Definition 6, i.e., they consider the latency in the same way and we are not interested in improvements relative to some $T > 0$.

Overall, our dataset consists of 31 randomly created feedforward networks with a total of 4479 flows. Table 3 gives an overview over the most important properties of the created networks. The full dataset is available online[1]. The distribution of network sizes, measured in amount of flows, is included in the x-axes of Figures 4, 8 and 9.

Table 3: Statistics on the created dataset.

Property	Min	Max	Median	Mean
# of servers	6	72	36	35.29
# of flows	4	448	103	144.48
Path length of flows	1	11	4	4.10

5.2 Evaluation Setup

All considered analyses have been implemented in the NCorg DNC[2]: LUDB-FF was released with v2.7.0 and our implementations of LB-FF as well as DS-FF are part of v2.8.0. An implementation of FF-LPA is publicly available separately[3]. For LUDB-FF and FF-LPA we make

[1] https://github.com/alexscheffler/dataset-rtns2022
[2] dnc.networkcalculus.org → https://github.com/NetCal/DNC
[3] https://github.com/bocattelan/DiscoDNC-FIFO-Optimization-Extension v1.0.

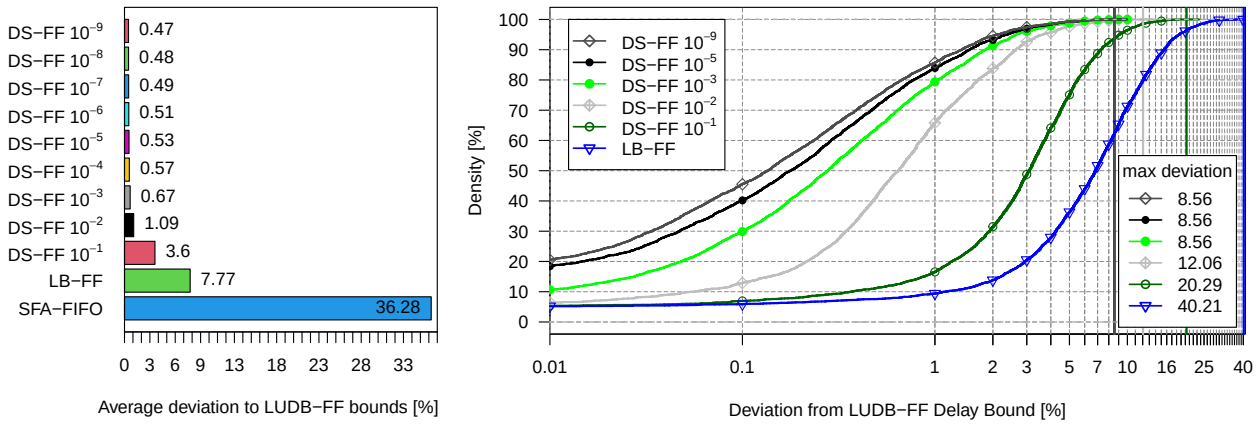

Figure 7: Comparison delay bounds w.r.t. LUDB-FF, average deviations and Cumulative Distribution Function.

use of the IBM CPLEX v20.1 solver that is generally faster than the open-source LpSolve. The runtimes were measured on a server with AMD Ryzen Threadripper PRO 3955WX CPU clocked at (max) 4.30 GHz, running Ubuntu 20.04.2 and OpenJDK 16 .

The NCorg DNC implementation of SFA-FIFO sets each server-local θ to minimize the output bound (see example given in Section 3.1.1). Note that all of these analyses make explicit use of the FIFO multiplexing assumption. We already showed in [22] that analyses which work for more general multiplexing strategies such as the state-of-the-art analysis TMA which was designed for ARB MUX, can be far off compared to LUDB-FF when it comes to delay bounds, especially for high utilizations. Concerning DS-FF$_\epsilon$, we opted for ϵ ranging from 10^{-1} to 10^{-9} since our evaluation shows that further decreasing ϵ only marginally reduces the delay bounds and comes at the price of a much higher runtime, potentially even higher than our main competitor LUDB-FF. Regarding the computation of the initial step sizes of DS-FF$_\epsilon$ we selected $c = 5$ (see Equation (8) for the definition of step size) and for the parameter ξ that determines by which factor the step sizes get decreased if the Exploratory Phase is not able to find a better bound is set to $\xi = 0.5$.

5.3 Comparison of Delay Bounds

First, we compare the bounds on flow delays derived by the different analyses. As a metric, we define the *relative delay bound* of an analysis A w.r.t. to some analysis B:

$$delay_{\text{A,B}} = \frac{delay_{\text{analysis A}} - delay_{\text{analysis B}}}{delay_{\text{analysis B}}}$$

5.3.1 A comparison to SFA-FIFO. We start by a small comparison of delay bounds against SFA-FIFO as analysis B. [22] already presented that in some corner cases, "modern" analyses that follow the "convolution before subtraction" scheme (see Section 3.1.2) can be inferior to SFA-FIFO.

I.e., the main analyses in this paper, analysis A $\in$ {LUDB-FF, LB-FF, DS-FF$_\epsilon$ }, may be beaten by SFA-FIFO. Figure 6 shows a sample comparison for LUDB-FF, LB-FF and DS-FF$_{10^{-1}}$.

These competing bounds are better than SFA-FIFO for almost all flows our networks. This comes at no surprise as the "modern" analyses aim at capturing the beneficial PMOO effect when the

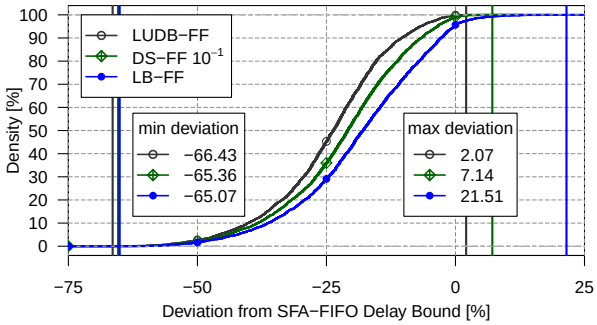

Figure 6: Relative Delay Bounds w.r.t. SFA-FIFO.

analyzed foi and its crossflows share multiple servers in sequence. Still, LUDB-FF yields better delay bounds than SFA-FIFO for 99.84% of the flows. The less precise LB-FF is better than SFA-FIFO for 95.71% of the flows. Delay bounds derived by any DS-FF variant always are in between those of LB-FF and LUDB-FF, converging to the latter with decreasing termination criterion ϵ. Interestingly, even for the largest $\epsilon = 10^{-1}$ in our evaluation, we can see that the delay bounds tend to be actually closer to LUDB-FF than to LB-FF – namely, in 99.20% cases DS-FF$_{10^{-1}}$ beats SFA-FIFO.

In terms of the *relative delay bound* defined above, LUDB-FF can be up to 2.07% worse than SFA-FIFO and LB-FF even up to 21.51%. The search-based approach with the coarsest granularity, DS-FF$_{10^{-1}}$, can be up to 7.14% worse than SFA-FIFO. On the other hand, we can observe that LUDB-FF can deliver delay bounds which are up to 66.43% better than SFA-FIFO. DS-FF$_{10^{-1}}$ (LB-FF) can similarly be better than SFA-FIFO by up to 65.36% (65.07%).

It is possible to achieve delay bounds that are always as good as all the above ones, while still applying the "convolution before subtraction" scheme: decompose the tandem under analysis into subtandems first, ranging from a single end-to-end one (as LUDB-FF, LB-FF, DS-FF do) to individual server decomposition (as SFA-FIFO does). Similar work has been done in the branch of NC without considering the FIFO property [4]. An exhaustive enumeration of subtandem decompositions is expensive such that the use Machine

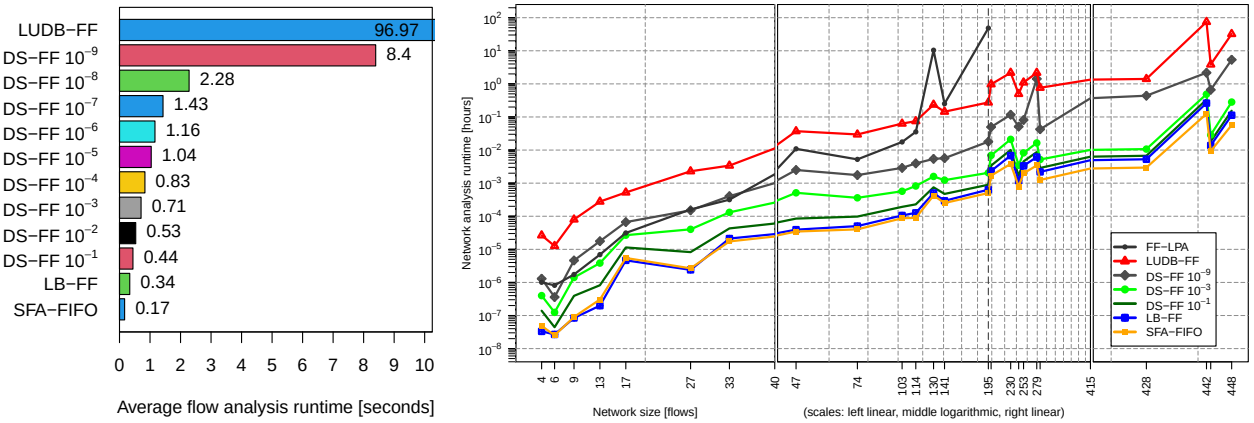

Figure 8: Comparison of runtimes, average runtime per flow and total runtime for analysis of all flows per network size.

Learning predictions [16] is beneficial. However, preventing any decompositions, even those inside LUDB-FF, LB-FF or DS-FF, was recently shown to improve delay bounds considerably [18, 19] and we strive for a combination with this approach in future work.

5.3.2 Comparing our new Analysis to LUDB-FF. Having confirmed that SFA-FIFO is not competitive, we shift our attention to a comparison to LUDB-FF.

This first evaluation confirms our working hypothesis: It is generally more promising to compute delay bounds with an analysis trying to achieve PMOO. For this very reason, we have investigated LUDB-FF and aimed at a faster analysis alternative that can still derive competitive bounds. To better present how far off our new analyses LB-FF and DS-FF are from LUDB-FF, we continue with relative bounds to the latter, i.e, $delay_{\text{LB-FF,LUDB-FF}}$ and $delay_{\text{DS-FF,LUDB-FF}}$. Figure 7 presents our observations for different sets of contenders. First of all, we can validate that LB-FF and DS-FF never deliver better bounds than LUDB-FF. Moreover, we can see that LB-FF can deliver bounds which are up 40.21% off w.r.t. LUDB-FF while $\text{DS-FF}_{10^{-1}}$ is already able to half this potential gap (20.29%). When further lowering the termination criterion ϵ to 10^{-3}, i.e., $\text{DS-FF}_{10^{-3}}$, we can see that we are already getting quite close to LUDB-FF – in fact, further lowering the ϵ only marginally improves the delay bounds. The average relative delay of LB-FF is at 7.77%, the one of $\text{DS-FF}_{10^{-1}}$ is 3.6% and for $\text{DS-FF}_{\epsilon \leq 10^{-4}}$ not much gain is achieved below 0.6% while converging to 0.45%.

5.3.3 A Note on Reproducibility. During repetitions of our evaluations, we noted that DS-FF delay bounds were not entirely reproducible although all employed methods behave deterministically. We could pinpoint the cause to the following: The NCorg DNC makes extensive use of sets and double precision floating point numbers internally. Thus, the order of operations may differ between runs and we observed that the unavoidable rounding errors differ, too. For small termination criteria in our DS-FF, these rounding errors could be decisive for termination. As a result of a continued search, the delay bounds computed with DS-FF in consecutive repetitions could differ notably. At times even more than a multiple of the small machine epsilon for double precision floating point numbers. Even though FF-LPA and LUDB-FF suffer from similar problems due to the employed solver (see Section 3.1.2), we were able to reproduce their results between runs.

5.4 Comparison of Computation Runtimes

The second metric of interest for any NC analysis is the time it takes to execute it, abbreviated with runtime. In our evaluation, we focus on the runtime to analyze an entire network of a certain size, i.e., the sum of all the runtimes for bounding an individual flow's end-to-end delay. Figure 8 depicts the runtime of each network size in our dataset for each of the competing analyses. Note that the methods LUDB-FF, DS-FF and LB-FF follow exactly the same steps to apply their respective tandem analysis to a feedforward network (see Section 3.2 for details). That means, these analyses aggregate the same flows and compute arrival bounds at the same locations in a feedforward network when they backtrack crossflows to their sources. These three analyses therefore only differ in the respective left-over service curve that, in turn, depends on their derived θ settings. In contrast, SFA-FIFO computes a left-over service curve at each server in the network, making it less complex regarding aggregation of crossflows.

First of all, we can observe (in this aggregate network runtime view) that LUDB-FF takes the longest to compute all delay bounds. Moreover, but not surprisingly, we have a clear ordering of runtimes between DS-FF with different termination criteria ϵ. Recall that the search terminates when at least one of the step sizes is lower than ϵ, so potentially more θ combinations will be tested with a decreasing ϵ. Furthermore, since DS-FF uses the θ setting provided by LB-FF as starting point of its search, the latter's runtime will be a lower bound on DS-FF analyses. LUDB-FF takes more time to solve than LB-FF since LB-FF uses one specific θ-setting while LUDB-FF first has to find the best one by solving several LPs. The comparison of LUDB-FF and DS-FF can actually go in both directions, i.e., theoretically if we set the termination criterion ϵ of the search very low, LUDB-FF can be solved in less time than DS-FF. Figure 8 already indicates that for the network with 279 flows $\text{DS-FF}_{10^{-9}}$ already gets close to LUDB-FF by being only about 53% faster.

At this point, it is worthwhile to briefly refine our definition of runtime to the per-flow computation times: Doing so, we observed that only for the two lowest ϵ some of the runtimes were lower with

LUDB-FF. More precisely, DS-FF$_{10^{-7}}$ was slower only for one flow and w.r.t. DS-FF$_{10^{-9}}$ this number increased to 30 – although the latter still only represents about 0.67% of the flows in our dataset.

We have seen previously that DS-FF$_{10^{-3}}$ provides delay bounds which are close to LUDB-FF – losing at most 8.56% in delay bound accuracy – while further decreasing ϵ only results in marginal bound improvements. Regarding the runtime of the analyses, we can now report that the median relative runtime $\frac{\text{runtime}^{\text{LUDB-FF}}}{\text{runtime}^{\text{other}}}$ of DS-FF$_{10^{-3}}$ w.r.t. LUDB-FF is about 120. I.e., DS-FF$_{10^{-3}}$ is about two orders of magnitude faster to compute bounds of almost the same accuracy.

When it comes to the runtimes of SFA-FIFO and LB-FF, we can observe from Figure 8 that the average runtime of LB-FF is twice as high than SFA-FIFO. First note that both analyses set their respective θs in the same static way (instead of trying different values like DS-FF) – however, they differ in the amount of θs to set. The trend that LB-FF takes longer than SFA-FIFO is especially predominant in larger networks where LB-FF follows the more complex procedure to dynamically derive subtandems, cut locations and crossflows aggregates, all based on how they interleave on the tandem under analysis while SFA-FIFO statically follows the principle to remove crosstraffic arrival at each server. In smaller networks (not shown), however, SFA-FIFO can be slower than LB-FF since the latter does not necessarily experience long non-nested tandems during the analysis (if at all) and thus has to compute less arrival bounds. Note that DS-FF and LUDB-FF yield worse runtimes than SFA-FIFO throughout all networks – the "additional" search or optimization negates the benefit of potentially computing less arrival bounds (in small networks) than SFA-FIFO.

5.5 Comparison to FF-LPA

Since the perfect optimization approach, namely FF-LPA and FF-MILPA, has scalability issues as it has to deal with an exponential number of variables and constraints, it was only possible to compute results for a subset of our set of networks: we were able to analyze 21 of our 31 networks in reasonable time. Figure 8 depicts the total runtime to analyze specific networks of our subset. We can observe that DS-FF$_{10^{-3}}$ is faster than FF-LPA for almost all networks. On the other hand, FF-LPA can be faster than LUDB-FF for smaller networks while the tables turn for larger network sizes. In particular, for the biggest network in our subset, FF-LPA needs more than 176 times as long as LUDB-FF does and for the next bigger network FF-LPA gets unpredictably "stuck" for one of the first flows we analyzed and still does not deliver a result after letting it run longer than LUDB-FF needed to analyze all of our networks.

Concerning delay bounds it must be noted that FF-LPA delivered delay bounds that were either the same or often times lower compared to LUDB-FF. To show the biggest extent to which this is true, we depict in Figure 9 the largest delay bound per network size. Although FF-LPA does not always compute the WCD as FF-MILPA does, this can be explained as follows: the algebraic feedforward analysis LUDB-FF always has to (recursively) bound crossflows at interfering servers which leads to more pessimistic delay bounds for the foi. However, the results of [18, 19] are promising in the sense that they may be able to close the gap we show here. All in all, FF-LPA is not suitable for the analysis of large-scale networks

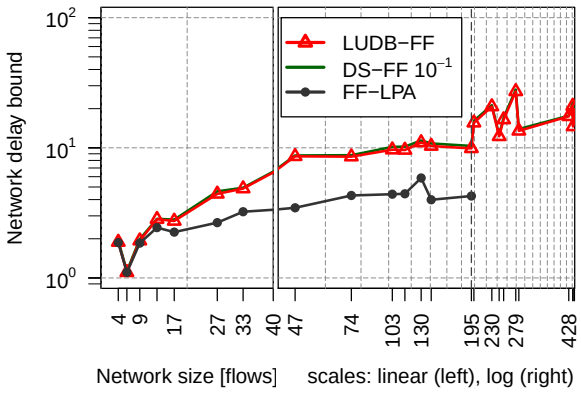

Figure 9: Maximum delay bounds per network size.

which is the aim of this paper since it suffers from the fact that for a certain network size it becomes unpredictable when FF-LPA will return a bound due to the exponentially-sized optimization problem.

6 CONCLUSION

In this paper we have presented novel First-In First-Out (FIFO)-aware analysis techniques for computing delay bounds in feedforward networks. Lower θ-Bound for Feedforward Analysis (LB-FF) is the technique that sets all FIFO parameters to a certain value in a static way, while Directed θ-Search for Feedforward Analysis (DS-FF) does so after considering different settings whose termination criterion can be set in a flexible way. Although the delay bounds of LB-FF and DS-FF are less accurate than the existing Least Upper Delay Bound (LUDB) for Feedforward Networks (LUDB-FF) and the optimization-based method Feedforward Linear Programming Analysis (FF-LPA), we provide a significant improvement in computation runtimes. The search-based method DS-FF$_{10^{-3}}$, DS-FF with termination criterion 10^{-3} as the minimum change of FIFO parameters θ_i in a search step, is on average 120 times faster than LUDB-FF while delivering delay bounds that are worse by only 0.57% on average and maximally 8.56%. Moreover, LUDB-FF only works with specific types of arrival and service curves, namely token-bucket and rate-latency curves while our proposed methods can be applied to network models with more curves, e.g., staircase functions commonly used for modeling periodic, packetized arrivals.

Last, also other work on improving the DNC method as well as applying it may easily benefit from a tailored directed search as we present it. E.g., the aforementioned work on the Flow Prolongation (FP) feature in DNC could be solved with a search instead of a Neural Network [18, 19]. [1] employs a Genetic Algorithm for priority assignment of the connections in an Ethernet-based avionics system (ARINC 664) standard, DNC serves as the fitness function.

Acknowledgements. We thank the anonymous reviewers for the detailed comments that helped improve the paper. Moreover, we thank Lukas Herll for his help in identifying the cause for limited reproducibility (Section 5.3.3).

REFERENCES

[1] Eyüp Can Akpolat, Ömer Faruk Gemici, M Selim Demir, İbrahim Hökelek, Sinem Coleri, and Hakan Ali Çırpan. 2021. Genetic Algorithm Based ARINC 664 Mixed Criticality Optimization Using Network Calculus. In *Proc. of the IEEE International Conference on Communications Workshops (ICC Workshops)*.

[2] 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)*.

[3] Luca Bisti, Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. 2010. *Computation and Tightness Assessment of End-to-end Delay Bounds in FIFO-multiplexing Tandems.* Technical Report.

[4] 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.

[5] Steffen Bondorf and Jens B. Schmitt. 2014. The DiscoDNC v2 – A Comprehensive Tool for Deterministic Network Calculus. In *Proc. of EAI International Conference on Performance Evaluation Methodologies and Tools (ValueTools)*.

[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] Marc Boyer and Christian Fraboul. 2008. Tightening end to end delay upper bound for AFDX network calculus with rate latency FIFO servers using network calculus. In *in Proc of. the IEEE International Workshop on Factory Communication Systems (WFCS)*.

[11] Marc Boyer, Nicolas Navet, Xavier Olive, and Eric Thierry. 2010. The PEGASE project: precise and scalable temporal analysis for aerospace communication systems with network calculus. In *Proc. of the International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA)*.

[12] Vicent Cholvi, Juan Echagüe, and Jean-Yves Le Boudec. 2002. Worst case burstiness increase due to FIFOd multiplexing. *Elsevier Performance Evaluation* 49, 1 (2002), 491–506.

[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] Markus Fidler and Volker Sander. 2004. A parameter based admission control for differentiated services networks. *Elsevier Computer Networks* 44, 4 (2004), 463–479.

[15] Fabien Geyer. 2017. Routing Optimization for SDN Networks Based on Pivoting Rules for the Simplex Algorithm. In *Proc. of International Conference on the Design of Reliable Communication Networks (DRCN)*.

[16] Fabien Geyer and Steffen Bondorf. 2019. DeepTMA: Predicting Effective Contention Models for Network Calculus using Graph Neural Networks. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*.

[17] Fabien Geyer and Steffen Bondorf. 2022. Network Synthesis under Delay Constraints: The Power of Network Calculus Differentiability. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*.

[18] 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)*.

[19] Fabien Geyer, Alexander Scheffler, and Steffen Bondorf. 2022. Network Calculus with Flow Prolongation – A Feedforward FIFO Analysis enabled by ML. *arXiv:2202.03004 [cs.NI, cs.LG].* https://arxiv.org/abs/2202.03004

[20] Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. 2003. Delay bounds for FIFO aggregates: A case study. In *Proc. of the International Workshop on Quality of Future Internet Services (QofIS)*. 31–40.

[21] Christopher J. Price, Blair L. Robertson, and Marco Reale. 2009. A hybrid Hooke and Jeeves – direct method for non-smooth optimization. *Advanced Modeling and Optimization (AMO)* 11, 1 (2009), 43–61.

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

[23] Alexander Scheffler, Markus Fögen, and Steffen Bondorf. 2018. The Deterministic Network Calculus Analysis: Reliability Insights and Performance Improvements. In *Proc. of the IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*.

[24] Jens B. Schmitt, Frank A. Zdarsky, and Ivan Martinovic. 2008. Improving Performance Bounds in Feed-Forward Networks by Paying Multiplexing Only Once. In *Proc. of GI/ITG International Conference on Measurement, Modelling and Evaluation of Computing Systems (MMB)*.

[25] David Starobinski, Mark Karpovsky, and Lev A. Zakrevski. 2003. Application of network calculus to general topologies using turn-prohibition. *IEEE/ACM Transactions on Networking* 11, 3 (2003), 411–421.

[26] Luxi Zhao, Paul Pop, Qiao Li, Junyan Chen, and Huagang Xiong. 2017. Timing analysis of rate-constrained traffic in TTEthernet using network calculus. *Springer Real-Time Systems* 53, 2 (2017), 254–287.

[27] Luxi Zhao, Paul Pop, Zhong Zheng, and Qiao Li. 2018. Timing analysis of AVB traffic in TSN networks using network calculus. In *Proc. of the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*.

ERRATA

Erratum 1: 2022-12-24

Fixed unit in Figure 8, left side, to report *seconds* instead of *minutes* so the times sum up to the right side's total runtimes per network size.