On the Potential to Improve Accuracy of Network Calculus Analyses

Steffen Bondorf and Jens B. Schmitt

Distributed Computer Systems (DISCO) Lab,
University of Kaiserslautern, Germany,
Technical Report 392/15

Abstract The continuous evolution of network calculus led to a set of different analyses. Among them, there is a single analysis that can derive tights bounds in arbitrary feed-forward server graphs. However, the approach it takes results in the analysis being NP-hard and no efficient analysis algorithm is known. Therefore, the authors propose to confine to a less complex analysis based on their approach instead. Like previous network calculus analyses, it derives tight bounds for some networks and valid bounds with varying accuracy for any other network. In this paper, we examine the work on accurate network calculus analyses regarding their relative accuracy and the potential to improve these analyses.

1 Introduction to Network Calculus

We start with an in-depth introduction to network calculus that allows to derive the delay bounds used for accuracy evaluation.

1.1 The System Description

Network calculus was built around a simple system description [17] consisting of two parts:

Data Arrivals and Forwarding Service Flows are characterized by functions cumulatively counting their data. They belong to the set $\mathcal{F}_0$ of non-negative, wide-sense increasing functions:

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

We are particularly interested in the functions $A(t)$ and $A'(t)$ cumulatively counting a flow's data put into a server s and put out from s, both up until time t. These functions allow for a straight-forward derivation of flow delays.

Definition 1. *(Flow Delay)* Assume a flow with input A crosses a server s and results in the output A'. The (virtual) delay for a data unit arriving at time t is

$$D(t) = \inf\left\{\tau \ge 0 \mid A(t) \le A'(t+\tau)\right\}.$$

Note that the order of data within the flow needs to be retained for the (virtual) delay calculation [27].

Network calculus operates in the interval time domain, i.e., its functions of $\mathcal{F}_0$ bound the maximum data arrivals of a flow during any duration of length d.

Definition 2. *(Arrival Curve)* Given a flow with input A, a function $\alpha \in \mathcal{F}_0$ is an arrival curve for A iff

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

AFDX networks reserve resources for a maximum packet size b periodically sent with a minimum inter-arrival time t_δ such that flows have a maximum data arrival rate of $r = \frac{b}{t_\delta}$ in the fluid model of $\mathcal{F}_0$. This shape of arrival curve is commonly referred to as token bucket and belongs to the class $\mathcal{F}_{\mathrm{TB}} \subseteq \mathcal{F}_0$:

$$\mathcal{F}_{\mathrm{TB}} = \{\gamma_{r,b} \,|\, \gamma_{r,b}\,(0) = 0,\ \forall d > 0 \,:\, \gamma_{r,b}(d) = b + r \cdot d\}.$$

Scheduling and buffering leading to the output function $A'(t)$ depend on a server's forwarding service. It is lower bounded in interval time as well.

Definition 3. *(Service Curve)* If the service provided by a server s for a given input A results in an output A', then s offers a service curve $\beta \in \mathcal{F}_0$ iff

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

For instance, service offered by ethernet connections can be described by rate-latency curves $\mathcal{F}_{\mathrm{RL}} \subseteq \mathcal{F}_0$:

$$\mathcal{F}_{\mathrm{RL}} = \{\beta_{R,T} \,|\, \beta_{R,T}\,(d) = \max\{0, R \cdot (d - T)\}\,.$$

A number of servers fulfill a stricter definition of service curves that guarantees a higher output during periods of queued data, the so-called backlogged periods of a server.

Definition 4. *(Strict Service Curve)* Let $\beta \in \mathcal{F}_0$. Server s offers a strict service curve β to a flow iff, during any backlogged period of duration d, the output of the flow is at least equal to $\beta(d)$.

The Network In general, networks are modeled as graphs where a node represents a network device like a router or a switch. Devices can have multiple inputs and multiple outputs to connect to other devices (Figure 1a). This network model does not fit well with network calculus' server model for queueing analysis. DNC therefore analyzes so-called *server graphs*. Assuming that a network device's input buffer is served at line speed, queueing effects manifest at the output buffers. These are modeled by the graph's servers (see Figure 1b). AFDX equipment employs output buffering [16]. In wireless sensor networks, nodes usually possess a single transmitter. Thus, one sensor node corresponds to one server and the transmission range defines the server graph's links [3,5].

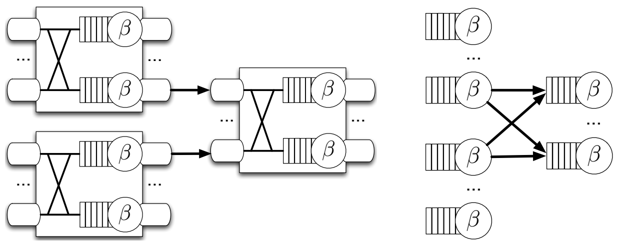
(a) Network Graph. (b) Server Graph.

Figure 1: A graph of network devices with output buffering (a) and its server graph connecting the devices' queues (b).

1.2 Algebraic Network Calculus

Network calculus was cast in a (min, +)-algebraic framework in [25,15]. We will first depict the basic operations and then present their combination for flow analysis.

(min,+)-Operations The following operations allow to manipulate arrival and service curve while retaining their worst-case semantic.

Definition 5. *((min,+)-Operations)* The (min,+)-aggrega-tion, -convolution and -deconvolution of two functions $f, g \in \mathcal{F}_0$ are defined as

$$\begin{aligned}
\textit{aggregation}: (f+g)(t) &= f(t)+g(t), \\
\textit{convolution}: (f \otimes g)(t) &= \inf_{0 \le s \le t} \{f(t-s)+g(s)\}, \\
\textit{deconvolution}: (f \oslash g)(t) &= \sup_{u \ge 0} \{f(t+u)-g(u)\}.
\end{aligned}$$

The system description's service curve definition then translates to $A' \geq A \otimes \beta$, the arrival curve definition to $A \otimes \alpha \geq A$, and performance characteristics can be bounded with the deconvolution $\alpha \oslash \beta$:

Theorem 1. (Performance Bounds) *Consider a server s that offers a service curve β. Assume a flow f with arrival curve α traverses the server. Then we obtain the following performance bounds for f:*

$$\begin{aligned}
\textit{delay}: \forall t \in \mathbb{R}^+ : \; & D(t) \le \inf\{d \ge 0 \,|\, (\alpha \oslash \beta)(-d) \le 0\}, \\
\textit{output}: \forall d \in \mathbb{R}^+ : \; & \alpha'(d) = (\alpha \oslash \beta)(d),
\end{aligned}$$

where the delay bound holds independent of t and α' is an arrival curve for A'.

Analyzing an entire flow with cross-traffic on its path is enabled by the following theorems. Table 1 depicts the notation required to analyze a sequence (tandem) of servers.

Theorem 2. (Concatenation of Servers) *Consider a single flow f crossing a tandem of servers $s_1, \ldots, s_n$ where each s_i offers a service curve β_{s_i}. The overall service curve for f is their concatenation by convolution*

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

Theorem 3. (Left-Over Service Curve) *Consider a server s that offers a strict service curve β_s. Let s be crossed by two flow aggregates $\mathbb{F}_0$ and $\mathbb{F}_1$ with aggregate arrival curves $\alpha^{\mathbb{F}_0}$ and $\alpha^{\mathbb{F}_1}$, respectively. Then $\mathbb{F}_1$'s worst-case residual resource share under arbitrary multiplexing at s, i.e., its left-over service curve at s, is*

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

with $(\beta \ominus \alpha)(d) := \sup \{0 \leq u \leq d \,|\, (\beta - \alpha)(u)\}$ denoting the non-decreasing upper closure of $(\beta - \alpha)(d)$.

Quantifier	Definition
foi	Flow of interest
$\mathbb{F}$	Aggregate of flows
$\{f_n, ..., f_m\}$	Flow aggregate containing flows $f_n, ..., f_m$
$F(s)$	Set of flows at server s
$F_{\mathrm{src}}(s)$	Set of flows originating at server s
$x(f)$, $x(\mathbb{F})$	Cross-traffic of flow f, aggregate $\mathbb{F}$
$\langle s_x, \ldots, s_y \rangle$	Tandem of consecutive servers s_x to s_y
α^f, $\alpha^{\mathbb{F}}$	Arrival curve of flow f, set of flows $\mathbb{F}$
α_s^f, $\alpha_s^{\mathbb{F}}$	Arrival bound at server s
β_s	Service curve of server s
$\beta^{\mathrm{l.o.}f}$, $\beta^{\mathrm{l.o.}\mathbb{F}}$	Left-over service curve

Table 1: Network Calculus Notation.

Algebraic Network Calculus Analysis A network calculus analysis computes the end-to-end delay bound for a specific flow (flow of interest, foi). From a conceptual point of view, algebraic network calculus analyses proceed in two steps [4,5]:

1. First, the analysis abstracts from the feed-forward network to the analyzed flow's path (tandem of servers). This step is enabled by recursive decomposing the server graph into tandems [28,19] and bounding the output arrivals of cross-traffic with Theorem 1, the output bound. Then, the worst-case shape

of cross-flows is known at the location of interference with the foi. Thanks to this step, the next one need not consider the part of the network traversed by cross-flows nor the interference pattern they are subject to.
2. The foi's end-to-end delay bound in the feed-forward network can now be calculated with a less complex *tandem analysis*. The flow's end-to-end service curve is derived and the delay bound computed.

The second step of the algebraic feed-forward analysis procedure has seen much treatment in the literature. Effort constantly focused on improving the ability to capture flow scheduling and cross-traffic multiplexing effects and thus provide more accurate delay bounds. The main results are:

The Separate Flow Analysis and the PBOO-effect [25]: Algebraic DNC tandem analyses result in a left-over service curve used for delay bound derivation with Theorem 1. Currently, there are two alternatives to compute the left-over service curve of a tandem: The Separate Flow Analysis (SFA) and the Pay Multiplexing Only Once (PMOO) analysis. The SFA is a straight-forward, hop-by-hop application of Theorems 3 and 2: First subtract cross-traffic arrivals and then concatenate the left-over service curves. Deriving the delay bound with a single, end-to-end left-over service curve will consider the flow of interest's burst term only once. This effect is therefore called Pay Bursts Only Once (PBOO). However, for cross-flows present at multiple consecutive hops, their bursts appear multiple times in the PBOO left-over service curve derivation.

The PMOO analysis /-effect [30]: The PMOO analysis provides an alternative containing each burst term only once. Its left-over service curve derivation reverses the operations, i.e., it convolves the tandem of servers before subtracting cross-traffic. Due to this end-to-end approach for all flows on the analyzed tandem, the PMOO analysis was considered superior to SFA. Yet, [29] shows that the SFA can arbitrarily outperform a PMOO tandem analysis. Both algebraic analyses thus complement each other. [29] also provides a left-over service curve derivation that outperforms both algebraic ones. It transforms the network calculus system description to an optimization-based analysis (OBA); departing from the algebraic methodology.

Optimization Based Network Calculus Analysis Based on the insight of [29], an optimization-based, tight feed-forward analysis was proposed in [9]. Its distinguishing factors allowing for tight bounds are:

– The server graph is not decomposed into tandems anymore. Instead, the new analysis transforms the entire feed-forward server graph into a set of linear programs (LPs) to optimize.
– Whereas the OBA of [29] optimizes the burstiness increase of flows, the LP analysis of [9] optimizes the start of backlogged periods propagating through the network.

– Moreover, the LP approach departs from the need to decompose arrival curves into multiple token buckets and service curves into multiple rate latencies in order to analyze all combinations individually [30]. The inferiority of curve decomposition is conceptually proven by examples in [22] and [8,9].

The LP analysis was, however, shown to be NP-hard. Therefore, the authors propose to confine to a less complex, accurate analysis based on their approach instead. This analysis is known as the unique LP (ULP). We provide a high-level wrap-up of the LP in order to answer the question how the ULP is derived from it. Clarifying where the ULP trades accuracy against computational effort allows us to judge the ULP's potential for improvement later. Intuitively, LP delay analysis proceeds as follows:

1. Starting from the flow of interest's sink server, flows as well as their respective cross-flows are recursively traced towards their sources. For every link traversed backwards, the start of backlogged periods at the link's source and its sink are related. This step terminates as soon as all flows are traced to their sources. The result is a partial order where, for example, there is no known order between the starts of backlogged periods for servers in different branches of a tree.
2. The second step is to extend the partial order to the set of all compatible total orders. This procedure enumerates all potential entanglements of events that define the start of backlogged periods at servers. For the above example, the entanglement of flow interference on distinct branches of a tree are enumerated such that all outcomes at these branches' common root server can be considered later. Special care must be taken of relations caused by rejoining flows. Each total orders resulting from this step constitutes the basis for one LP to solve with the help of a solver software like LpSolve or CPLEX.
3. This step transforms each total orders to one linear program. The order between the start of backlogged periods as well as the network calculus model (strict service curves, arrival curves, non-decreasing functions, non-negativity, flow constraint) are used to derive the constraints of each LP.
4. In the final step, all LPs need to be solved. The LPs model all potential entanglements of flows via the backlogged periods they influence, therefore only the maximum among the solutions is a valid worst-case bound for the flow of interest's delay. I.e., the LP analysis constitutes an all-or-nothing approach.

The second step suffers from combinatorial explosion [26] and known algorithms for linear extension like the Varol-Rotem algorithm [31] do not allow for the analysis of large networks. For this reason, the authors of the LP approach propose to skip this step with the ULP. The unique linear program consists of only the common constraints of the LPs. This effectively means executing the transformation of step 3 on the partial order that resulted from step 1; deriving the LPs to identify the common constraints is not required. Obviously, the ULP neither relates the flow entanglement between different branches in a tree nor does it handle the rejoining flows setting in any explicit way. For both situations the worst-case assumptions of network calculus ensure valid upper bounds.

2 Improving the Accuracy of Analyses

Given that the combinatorial explosion forces the LP approach to dismiss the tight bound derivation in general feed-forward networks such that accurate bounds are derived instead, we want to address the question of potential accuracy improvement among alternative network calculus analyses.

Improving the ULP The ULP constitutes the return to accurate, yet, untight bounds in general feed-forward networks. However, for some special networks it derives tight bounds nonetheless. The special case holds in tandem server graphs (see Figure 3) where all servers are crossed by the flow of interest. Then, the partial order happens to be a total order and the second step of the LP analysis does not have any impact. For more involved server graphs, we only know that the ULP will have less constraints than any of the LPs. Improving the ULP's result can only be achieved by adding more constraints to it. Judging the addition of constraints regarding the ability to improve bounds while simultaneously guaranteeing to retain their validity is, however, an open problem – it is probably even undecidable considering the computationally infeasible all-or-nothing approach it takes to identify the complete set of constraints for tight bounding. The ULP therefore does not seem to have much potential for accuracy improvements.

Algebraic Analyses Improving the algebraic network calculus analyses seems to have more potential than the ULP. We base this conclusion on following observations: The work pioneering optimization approaches in network calculus [29] also proves that the PMOO bound is in fact tight for tandem server graphs with a single sink and monotonically decreasing residual service rates. I.e, the PMOO analysis constitutes an algebraic solution to the optimization problem for this specific server graph. Recently, the cross-traffic arrival bounding step required for algebraic feed-forward analysis has seen a substantial improvement [6] over the state-of-the art approach at the time the ULP analysis was constructed [7]. This shows that algebraic network calculus analyses possess untapped potential for improvement. In the remainder of this paper we will address the questions if exploiting this potential is meaningful with respect to the currently known accuracy gap to the ULP (Section 3) and computational effort of analyses (Section 4).

3 Accuracy Evaluation Network Calculus Analyses

In this section, we want to address the question of accuracy of the existing network calculus analyses. For the ULP, we strive for insight on the impact of omitting step 2 from the LP approach, and regarding the algebraic analyses, we aim to illustrate the advantage of the three distinguishing factors of the LP approach.

In order to do so, we need to rely on the evaluation and tooling provided by the authors of [9] and their additional material (incl. a tool) available online [2]. In contrast to algebraic network calculus, there is no comprehensive tool support available for the LP analysis of feed-forward networks as we will se in the following. Unfortunately, [2] neither provides the algebraic calculations resulting in the bounds shown in [8,9,20,7,10] nor could we verify them. Therefore, we provide an entire re-evaluation of the (U)LP as well as the SFA. We omit the Total Flow Analysis (TFA) for being known to result in inferior end-to-end delay bounds but contribute PMOO analysis results.

We apply the measure of accuracy used in [9]: $\delta_{\mathrm{LP}}\,(\text{analysis}) = \frac{D(\text{analysis})}{D(\text{LP})}$. Knowing that the LP will result in the tight bound, this measure starts at a value of 1.0 in network settings where analysis$\in$ {ULP,PMOO,SFA} derives the tight bound as well.

3.1 The Non-Nested Tandem

Analyzing tandem networks with the LP is made possible with a tool available online [2]. It transforms a text file describing the tandem (servers, service curves, flows, paths, arrival curves) into the (U)LP – given as an LpSolve lp file. In [9], the so-called tandem with non-nested interference is analyzed. It consists of a sequence of servers crossed by the flow of interest and overlapping interference of cross-flows such that there are three flows at every server (see Figure 2). This tandem constitutes a standard example in network calculus; it is already used for introducing the PMOO analysis [30].

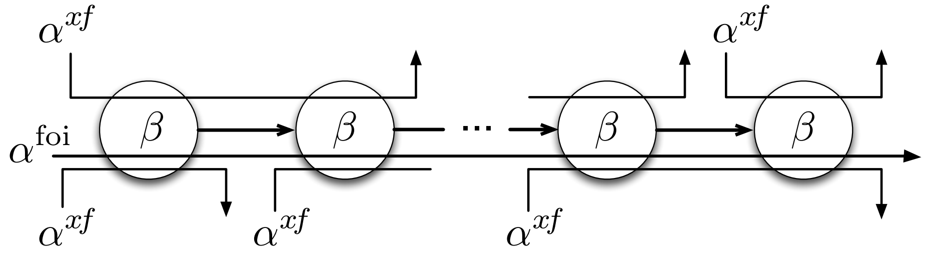

Figure 2: Tandem server graph with homogeneous service and homogeneous non-nested cross-traffic arrivals.

Two evaluations are carried out: One that evaluates the impact of tandem length, and another one varying the utilization for the tandem of length 20. For algebraic DNC, we provide a sample computation for the tandem of length 2, the minimum tandem length where the PMOO effect can be exploited. These computation illustrate the algebraic procedure as implemented in the DiscoDNC [4], our tool for algebraic DNC server graph analysis that we used for the remaining computations. For further reading about the computations behind the following $(\min, +)$-operations, please refer to [25,30,11] as well as the documentation of the DiscoNC tool found online [1].

The 2-tandem server graph is constructed as follows; servers are enumerated to be s_1 and s_2 and the flow configuration is:

– The flow of interest (foi) crosses s_1 and s_2,
– Cross-flow $x\!f_1$ crosses server s_1,
– Cross-flow $x\!f_2$ crosses servers s_1 and s_2, and
– Cross-flow $x\!f_3$ crosses server s_2.

Arrival curves and service curves are provided in [9]: Both evaluations assign rate-latency service $\beta_{R,L} = \beta_{10,\frac{1}{10}}$, the evaluation under varying utilization $u \in \{0.1, \ldots, 0.9\}$ operates with token bucket arrival curves of $\alpha = \gamma_{r,b} = \gamma_{\frac{10u}{3},1}$ where $\frac{10u}{3}$ is rounded to two decimal digits. The evaluation of tandem length is carried out at a utilization of 0.2.

Separate Flow Analysis (SFA) First, we derive the flow of interest's end-to-end left-over service curve:

$$\begin{aligned}
\beta^{\text{l.o.foi}}_{\langle s_1,s_2\rangle} &= \beta^{\text{l.o.foi}}_{s_1} \otimes \beta^{\text{l.o.foi}}_{s_2} \\
&= \left(\beta_{s_1} \ominus \left(\alpha^{x\!f_1}_{s_1} + \alpha^{x\!f_2}_{s_1}\right)\right) \otimes \left(\beta_{s_2} \ominus \left(\alpha^{x\!f_2}_{s_2} + \alpha^{x\!f_3}_{s_2}\right)\right) \\
&= \left(\beta_{s_1} \ominus \left(\alpha^{x\!f_1} + \alpha^{x\!f_2}\right)\right) \otimes \left(\beta_{s_2} \ominus \left(\left(\alpha^{x\!f_2}_{s_1} \oslash \beta^{\text{l.o.}x\!f_2}_{s_1}\right) + \alpha^{x\!f_3}\right)\right) \\
&= \left(\beta_{s_1} \ominus \left(\alpha^{x\!f_1} + \alpha^{x\!f_2}\right)\right) \otimes \left(\beta_{s_2} \ominus \left(\left(\alpha^{x\!f_2} \oslash \left(\beta_{s_1} \ominus \alpha^{x\!f_1}\right)\right) + \alpha^{x\!f_3}\right)\right) \\
&= (\beta_{10,0.1} \ominus (\gamma_{0.67,1} + \gamma_{0.67,1})) \otimes (\beta_{10,0.1} \ominus ((\gamma_{0.67,1} \oslash (\beta_{10,0.1} \ominus \gamma_{0.67,1})) + \gamma_{0.67,1})) \\
&= (\beta_{10,0.1} \ominus \gamma_{1.34,2}) \otimes \left(\beta_{10,0.1} \ominus \left(\left(\gamma_{0.67,1} \oslash \beta_{9.33,\frac{2}{9.33}}\right) + \gamma_{0.67,1}\right)\right) \\
&= \beta_{8.66,\frac{3}{8.66}} \otimes \left(\beta_{10,0.1} \ominus \left(\gamma_{0.67,1+\frac{1.34}{9.33}} + \gamma_{0.67,1}\right)\right) \\
&= \beta_{8.66,\frac{3}{8.66}} \otimes \left(\beta_{10,0.1} \ominus \gamma_{1.34,2+\frac{1.34}{9.33}}\right) \\
&= \beta_{8.66,\frac{3}{8.66}} \otimes \beta_{8.66,\frac{3\frac{1.34}{9.33}}{8.66}} \\
&= \beta_{8.66,\frac{6\frac{1.34}{9.33}}{8.66}}
\end{aligned}$$

And next, we compute the delay bound:

$$\begin{aligned}
D^{\mathrm{SFA}} &= h\left(\alpha^{\mathrm{foi}}, \beta^{\mathrm{l.o.}f_1}_{\langle s_3,s_4\rangle}\right) = h\left(\gamma_{\frac{2}{3},1}, \beta_{8.66,\frac{6\frac{1.34}{9.33}}{8.66}}\right) \\
&= \frac{1 + 8.66 \cdot \frac{6\frac{1.34}{9.33}}{8.66}}{8.66} \\
&= \frac{7\frac{1.34}{9.33}}{8.66} \\
&= 0.8248986977
\end{aligned}$$

In this bound derivation, we can see that the PMOO property is not fulfilled. We need to pay for xf_2's arrivals at every server it shares with the flow of interest. I.e., not only at s_1 it starts to interfere, but also at s_2 where xf_2's (increased) worst-case burstiness impacts the derivation (see the occurrence of α^{xf_2} on both sides of the convolution $\otimes$). For this reason, the SFA result will be worst than the PMOO results.

Pay Multiplexing Only Once (PMOO) analysis We start with the left-over service curve derivation again:

$$\beta^{\mathrm{l.o.foi}}_{\langle s_1,s_2\rangle} = \beta_{R^{\mathrm{l.o.foi}}_{\langle s_1,s_2\rangle}, T^{\mathrm{l.o.foi}}_{\langle s_1,s_2\rangle}}$$

$$\begin{aligned}
R^{\mathrm{l.o.foi}}_{\langle s_1,s_2\rangle} &= \left(R_{s_1} - r^{xf_1} - r^{xf_2}\right) \wedge \left(R_{s_2} - r^{xf_2} - r^{xf_3}\right) \\
&= (10 - 0.67 - 0.67) \wedge (10 - 0.67 - 0.67) \\
&= 8.66
\end{aligned}$$

$$\begin{aligned}
T^{\mathrm{l.o.foi}}_{\langle s_1,s_2\rangle} &= T_{s_1} + T_{s_2} + \frac{b^{xf_1}_{s_1} + b^{xf_2}_{s_1} + b^{xf_3}_{s_2} + \left(r^{xf_1}_{s_1} + r^{xf_2}_{s_1}\right) \cdot T_{s_1} + \left(r^{xf_2}_{s_2} + r^{xf_3}_{s_2}\right) \cdot T_{s_2}}{R^{\mathrm{l.o.foi}}_{\langle s_3,s_4\rangle}} \\
&= 0.1 + 0.1 + \frac{1 + 1 + 1 + (0.67 + 0.67) \cdot 0.1 + (0.67 + 0.67) \cdot 0.1}{8.66} \\
&= 0.2 + \frac{3.268}{8.66} \\
&= \frac{5}{8.66} \\
&= \\
&=
\end{aligned}$$

$$\beta^{\mathrm{l.o.foi}}_{\langle s_1,s_2\rangle} = \beta_{8.66,\frac{5}{8.66}}$$

And compute the delay bound:

$$\begin{aligned}
D^{\mathrm{PMOO}} &= h\left(\alpha^{\mathrm{foi}}, \beta^{\mathrm{l.o.foi}}_{\langle s_1, s_2\rangle}\right) \\
&= h\left(\gamma_{\frac{2}{3},1}, \beta_{8.66,\frac{5}{8.66}}\right) \\
&= \frac{1 + 8.66\frac{5}{8.66}}{8.66} \\
&= \frac{6}{8.66} \\
&= 0.6928406467
\end{aligned}$$

In contrast to the SFA, the xf_2's burstiness only appears once in the $\beta^{\mathrm{l.o.foi}}_{\langle s_1, s_2\rangle}$-derivation. From the flow of interest's point of view, multiplexing with cross-traffic in payed for only once.

Linear Programming (LP) analysis and comparison The results for the LP analysis are depicted alongside the PMOO and SFA delay bounds in Tables 2a and 2b.

The results for the non-nested tandem do not come at a surprise: PMOO was designed to counteract the burstiness increase being considered in the flow of interest analysis multiple times. Therefore, it performs better on the non-nested tandem than the SFA. Moreover, the weakness of the PMOO analysis only surfaces in tandems with unbalanced (left-over) service rates [29]. This is not the case in the non-nested tandem setting chosen in [9]. Therefore, it was to be expected that the PMOO result equals any optimization based analysis delay bound, i.e., $\delta_{\mathrm{LP}}\,(\mathrm{PMOO}) = 1.00$. Similarly, the increased utilization is uniform among the servers such that PMOO and LP delay bounds increase in lockstep, starting from the same value.

The SFA performs worse due to the PMOO effect not being exploited by it. Increasing the tandem length, this leads to an initial gap, yet, the growth of $\delta_{\mathrm{LP}}\,(\mathrm{SFA})$ is slowing down fast and does not to change significantly anymore when adding another server to ten ore more existing ones. When increasing the utilization, the SFA performs considerably worse than the other analyses due to the lack of PMOO effect.

Servers	LP	PMOO	δ_{LP} (PMOO)	SFA	δ_{LP} (SFA)
1	0.46189376	0.46189376	1.00	0.46189376	1.00
2	0.69284065	0.69284065	1.00	0.82489870	1.19
3	0.92378753	0.92378753	1.00	1.18909460	1.29
4	1.15473441	1.15473441	1.00	1.55337602	1.35
5	1.38568129	1.38568129	1.00	1.91766358	1.38
6	1.61662818	1.61662818	1.00	2.28195158	1.41
7	1.84757506	1.84757506	1.00	2.64623962	1.43
8	2.07852194	2.07852194	1.00	3.01052766	1.45
9	2.30946882	2.30946882	1.00	3.37481570	1.46
10	2.54041570	2.54041570	1.00	3.73910373	1.47
11	2.77136259	2.77136259	1.00	4.10339177	1.48
12	3.00230947	3.00230947	1.00	4.46767981	1.49
13	3.23325635	3.23325635	1.00	4.83196785	1.49
14	3.46420323	3.46420323	1.00	5.19625590	1.50
15	3.69515012	3.69515012	1.00	5.56054392	1.50
16	3.92609700	3.92609700	1.00	5.92483196	1.51
17	4.15704388	4.15704388	1.00	6.28912000	1.51
18	4.38799076	4.38799076	1.00	6.65340804	1.52
19	4.61893764	4.61893764	1.00	7.01769607	1.52
20	4.84988453	4.84988453	1.00	7.38198412	1.52

(a) Delay bounds depending on the tandem length.

Utilization	LP	PMOO	δ_{LP} (PMOO)	SFA	δ_{LP} (SFA)
10%	4.49678801	4.49678801	1.00	6.67453059	1.48
20%	4.84988453	4.84988453	1.00	7.38198412	1.52
30%	5.25000000	5.25000000	1.00	8.21484375	1.56
40%	7.86516854	5.72207084	1.00	9.23976737	1.61
50%	6.30630631	6.30630631	1.00	10.57098749	1.68
60%	7.00000000	7.00000000	1.00	12.24074074	1.75
70%	7.86516854	7.86516854	1.00	14.45688339	1.84
80%	9.01287554	9.01287554	1.00	17.62145123	1.96
90%	10.50000000	10.50000000	1.00	22.09375003	2.10

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

Table 2: Delay bounds for the server tandem with non-nested interference of cross-flows.

3.2 The Square

Next, we re-evaluate the square server graph with two flows per server (see Figure 3). This server graph is evaluated for varying utilizations.

For this evaluation, service curves remain $\beta_{s_i} = \beta_{R,L} = \beta_{10,\frac{1}{10}}$, $i \in \{1,2,3,4\}$ and arrival curves are again adapted to the utilization $u \in \{0.1,\ldots,0.9\}$. As there are only two flows per server, this scheme translates to $\alpha^{f_i} = \gamma_{r,b} = \gamma_{\frac{10u}{2},1}$.

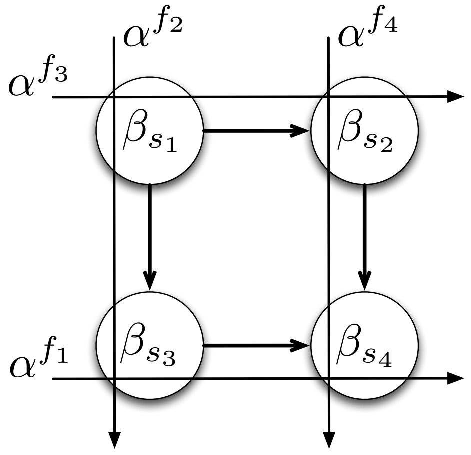

Figure 3: Square server graph.

Separate Flow Analysis (SFA) We start with the SFA left-over service curve derivation steps shared by every utilization's analysis:

$$\begin{aligned}
\beta^{\text{l.o.}f_1}_{\langle s_3,s_4\rangle} &= \beta^{\text{l.o.}f_1}_{s_3} \otimes \beta^{\text{l.o.}f_1}_{s_4} \\
&= \left(\beta_{s_3} \ominus \alpha^{f_2}_{s_3}\right) \otimes \left(\beta_{s_4} \ominus \alpha^{f_4}_{s_4}\right) \\
&= \left(\beta_{s_3} \ominus \left(\alpha^{f_2} \oslash \beta^{\text{l.o.}f_2}_{s_1}\right)\right) \otimes \left(\beta_{s_4} \ominus \left(\alpha^{f_4} \oslash \beta^{\text{l.o.}f_4}_{s_2}\right)\right) \\
&= \left(\beta_{s_3} \ominus \left(\alpha^{f_2} \oslash \left(\beta_{s_1} \ominus \alpha^{f_3}\right)\right)\right) \otimes \left(\beta_{s_4} \ominus \left(\alpha^{f_4} \oslash \left(\beta_{s_2} \ominus \alpha^{f_3}_{s_2}\right)\right)\right) \\
&= \left(\beta_{s_3} \ominus \left(\alpha^{f_2} \oslash \left(\beta_{s_1} \ominus \alpha^{f_3}\right)\right)\right) \otimes \left(\beta_{s_4} \ominus \left(\alpha^{f_4} \oslash \left(\beta_{s_2} \ominus \left(\alpha^{f_3} \oslash \beta^{\text{l.o.}f_3}_{s_1}\right)\right)\right)\right) \\
&= \left(\beta_{s_3} \ominus \left(\alpha^{f_2} \oslash \left(\beta_{s_1} \ominus \alpha^{f_3}\right)\right)\right) \otimes \left(\beta_{s_4} \ominus \left(\alpha^{f_4} \oslash \left(\beta_{s_2} \ominus \left(\alpha^{f_3} \oslash \left(\beta_{s_1} \ominus \alpha^{f_2}\right)\right)\right)\right)\right) \\
&= \left(\beta_{R_{s_3},T_{s_3}} \ominus \left(\gamma_{r^{f_2},b^{f_2}} \oslash \left(\beta_{R_{s_1},T_{s_1}} \ominus \gamma_{r^{f_3},b^{f_3}}\right)\right)\right) \\
&\quad \otimes \left(\beta_{R_{s_4},T_{s_4}} \ominus \left(\gamma_{r^{f_4},b^{f_4}} \oslash \left(\beta_{R_{s_2},T_{s_2}} \ominus \left(\gamma_{r^{f_3},b^{f_3}} \oslash \left(\beta_{R_{s_1},T_{s_1}} \ominus \gamma_{r^{f_2},b^{f_2}}\right)\right)\right)\right)\right) \\
&= \left(\beta_{10,\frac{1}{10}} \ominus \left(\gamma_{r,1} \oslash \left(\beta_{10,\frac{1}{10}} \ominus \gamma_{r,1}\right)\right)\right) \\
&\quad \otimes \left(\beta_{10,\frac{1}{10}} \ominus \left(\gamma_{r,1} \oslash \left(\beta_{10,\frac{1}{10}} \ominus \left(\gamma_{r,1} \oslash \left(\beta_{10,\frac{1}{10}} \ominus \gamma_{r,1}\right)\right)\right)\right)\right)
\end{aligned}$$

This derivation illustrates the different approaches applied by algebraic DNC to model the flow of interest's worst-case scenario in a feed-forward network. On the foi's path (i.e., the foi tandem analysis part of step 2), the left-over service curve derivation assumes lowest priority for the flow of interest (cf. Theorem 3). In the arrival bounding, each flow's worst-case interference is modeled individually. Therefore, at server s_1, flows f_2 and f_3 are considered as mutual interference: $\beta_{s_1}^{\text{l.o.}f_2} = \left(\beta_{s_1} \ominus \alpha^{f_3}\right)$ and $\beta_{s_1}^{\text{l.o.}f_3} = \left(\beta_{s_1} \ominus \alpha^{f_2}\right)$. For rejoining interference with the flow of interest (see Diamond network in [9]) this means the flow of interest is assumed to interfere with its own cross-flows during their arrival bounding. The lowest priority modeling applied in the flow of interest analysis is not carried over to cross-traffic arrival bounding, i.e., algebraic DNC analysis does not assign lowest priority to the foi within the entire feed-forward network but depending on the analysis context.

We restrict our manual computation to the case of 90% utilization, i.e., all flow arrival rates are set to $\frac{9}{2}$:

$$\begin{aligned}
\beta_{\langle s_3,s_4\rangle}^{\text{l.o.}f_1} &= \left(\beta_{10,\frac{1}{10}} \ominus \left(\gamma_{4\frac{1}{2},1} \oslash \left(\beta_{10,\frac{1}{10}} \ominus \gamma_{4\frac{1}{2},1}\right)\right)\right) \\
&\quad \otimes \left(\beta_{10,\frac{1}{10}} \ominus \left(\gamma_{4\frac{1}{2},1} \oslash \left(\beta_{10,\frac{1}{10}} \ominus \left(\gamma_{4\frac{1}{2},1} \oslash \left(\beta_{10,\frac{1}{10}} \ominus \gamma_{4\frac{1}{2},1}\right)\right)\right)\right)\right) \\
&= \left(\beta_{10,\frac{1}{10}} \ominus \left(\gamma_{4\frac{1}{2},1} \oslash \beta_{5\frac{1}{2},\frac{4}{11}}\right)\right) \\
&\quad \otimes \left(\beta_{10,\frac{1}{10}} \ominus \left(\gamma_{4\frac{1}{2},1} \oslash \left(\beta_{10,\frac{1}{10}} \ominus \left(\gamma_{4\frac{1}{2},1} \oslash \beta_{5\frac{1}{2},\frac{4}{11}}\right)\right)\right)\right) \\
&= \left(\beta_{10,\frac{1}{10}} \ominus \gamma_{4\frac{1}{2},2\frac{7}{11}}\right) \otimes \left(\beta_{10,\frac{1}{10}} \ominus \left(\gamma_{4\frac{1}{2},1} \oslash \left(\beta_{10,\frac{1}{10}} \ominus \gamma_{4\frac{1}{2},2\frac{7}{11}}\right)\right)\right) \\
&= \beta_{5\frac{1}{2},\frac{80}{121}} \otimes \left(\beta_{10,\frac{1}{10}} \ominus \left(\gamma_{4\frac{1}{2},1} \oslash \beta_{5\frac{1}{2},\frac{80}{121}}\right)\right) \\
&= \beta_{5\frac{1}{2},\frac{80}{121}} \otimes \left(\beta_{10,\frac{1}{10}} \ominus \gamma_{4\frac{1}{2},3\frac{118}{121}}\right) \\
&= \beta_{5\frac{1}{2},\frac{80}{121}} \otimes \beta_{5\frac{1}{2},\frac{1204}{1331}} \\
&= \beta_{5\frac{1}{2},1\frac{753}{1331}}
\end{aligned}$$

$$\begin{aligned}
D^{\text{SFA}} &= h\left(\alpha^{f_1}, \beta_{\langle s_3,s_4\rangle}^{\text{l.o.}f_1}\right) \\
&= h\left(\gamma_{4\frac{1}{2},1}, \beta_{5\frac{1}{2},1\frac{753}{1331}}\right) \\
&= 1\frac{995}{1331} \\
&= 1.7475582269
\end{aligned}$$

Pay Multiplexing Only Once (PMOO) analysis Utilization 90%:

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

$$\begin{aligned} R^{\mathrm{l.o.}f_1}_{\langle s_3,s_4\rangle} &= \left(R_{s_3} - r^{f_2}\right) \wedge \left(R_{s_4} - r^{f_4}\right) \\ &= \left(10 - 4\frac{1}{2}\right) \wedge \left(10 - 4\frac{1}{2}\right) \\ &= 5\frac{1}{2} \end{aligned}$$

$$\begin{aligned} T^{\mathrm{l.o.}f_1}_{\langle s_3,s_4\rangle} &= T_{s_3} + T_{s_4} + \frac{b^{f_2}_{s_3} + b^{f_4}_{s_4} + r^{f_2}_{s_3} \cdot T_{s_3} + r^{f_4}_{s_4} \cdot T_{s_4}}{R^{\mathrm{l.o.}f_1}_{\langle s_3,s_4\rangle}} \\ &= \frac{1}{10} + \frac{1}{10} + \frac{2\frac{7}{11} + 3\frac{118}{121} + 4\frac{1}{2} \cdot \frac{1}{10} + 4\frac{1}{2} \cdot \frac{1}{10}}{5\frac{1}{2}} \\ &= \frac{2}{10} + \frac{6\frac{74}{121} + \frac{9}{10}}{5\frac{1}{2}} \\ &= \frac{2}{10} + \frac{7\frac{619}{1210}}{5\frac{1}{2}} \\ &= \frac{2}{10} + 1\frac{2434}{6655} \\ &= 1\frac{753}{1331} \end{aligned}$$

$$\beta^{\mathrm{l.o.}f_1}_{\langle s_3,s_4\rangle} = \beta_{5\frac{1}{2},\frac{753}{1331}}$$

$$\begin{aligned} D^{\mathrm{PMOO}} &= h\left(\alpha^{f_1}, \beta^{\mathrm{l.o.}f_1}_{\langle s_3,s_4\rangle}\right) \\ &= h\left(\gamma_{4\frac{1}{2},1}, \beta_{5\frac{1}{2},1\frac{753}{1331}}\right) \\ &= 1\frac{995}{1331} \\ &= 1.7475582269 \end{aligned}$$

The mutual interference modeling persists. It can be seen in the cross-flow burst terms $b^{f_2}_{s_3}$ and $b^{f_4}_{s_4}$ that equal the one of the according arrival bounds $\alpha^{f_2}_{s_3}$ and $\alpha^{f_4}_{s_4}$ used in the SFA.

Linear Programming (LP, ULP) analysis and comparison Online [2], there are LpSolve lp files for the square server graph's LP analysis and the ULP analysis; no tool for automatic generation of the LPs for general feed-forward networks is available. Given the similarity of the diamond network used in [9] and the square network, it can be assumed that these files have been set up manually for rate-latency service and token bucket arrivals. From the user perspective, all that needs to be done is to fill in the curves' parameters.

Recall that the LP approach enumerates all potential entanglements of flows by extending the partial order (defined by consecutive hops of flows) to the set of all compatible total orders. The benefit is already pointed out above and part of the diamond network to explain the procedure of deriving the set of LPs. The ULP, in contrast, is solely based on the partial order, the extension step is omitted due to its combinatorial explosion. This results in a smaller set of ULP constraints, especially the potential entanglements of f_2 and f_3 at s_1 are not considered anymore. Instead they are assumed to constitute the respective flow's worst case. Based on this insight, is it no surprising that the ULP actually does not beat the PMOO analysis in the square network (see Table 3 and the solution to all LPs in Appendix I). In fact, even the SFA yields the same results. The reason for SFA's accuracy is the lack of multi-hop interference. The PMOO effect cannot manifest in this server graph.

Our evaluation shows that the ULP models the worst-case interference between f_2 and f_3 in the exact same way as all the algebraic analyses (see $\delta_{\text{ULP}}\,(\text{analysis}) = \frac{D(\text{analysis})}{D(\text{ULP})} = 1.00$ for all analysis$\in$ {PMOO,SFA} and all utilizations).

Utilization	LP	ULP	PMOO	SFA	δ_{LP}	δ_{ULP}
10%	0.54351946	0.54905963	0.54905963	0.54905963	1.00	1.00
20%	0.59533608	0.60768176	0.60768176	0.60768176	1.02	1.00
30%	0.65784653	0.67860778	0.67860778	0.67860778	1.03	1.00
40%	0.73437500	0.76562500	0.76562500	0.76562500	1.04	1.00
50%	0.82962963	0.87407407	0.87407407	0.87407407	1.05	1.00
60%	0.95043732	1.01166181	1.01166181	1.01166181	1.06	1.00
70%	1.10696404	1.18980428	1.18980428	1.18980428	1.07	1.00
80%	1.31481481	1.42592593	1.42592593	1.42592593	1.08	1.00
90%	1.65777147	1.74755823	1.74755823	1.74755823	1.05	1.00

Table 3: Square server graph delay bounds.
PMOO and SFA results are rounded to match LpSolve's single precision results.

Summing up, neither of the evaluations given in [9] illustrated the superiority of the ULP over the PMOO analysis. Moreover, the ULP evaluation consists of a single server graph, such that no conclusions about its accuracy in general feed-forward networks cannot be drawn.

4 Computational Effort Considerations

Last, let us extend our reflection of the currently available evaluation of network calculus analyses by computational effort considerations – they caused the step from tight LP analysis back to accurate analysis with the ULP in the first place.

Algebraic network calculus can be considered proven to be applicable. There is work for sensor networks of increasing size [24,21,3,5] and the continuous effort to verify AFDX networks [12,18,13] which tend to grow in size as well. Tool support for algebraic network calculus is also available, open-source [4] as well as commercial [14].

Optimization based network calculus analysis does not offer comprehensive tool support as of now. Therefore evaluation of effort is scarce in the literature. The effort of the OBA was evaluated in [22,23]. It is shown that the computational hardness mainly arises from the large amount of combinations to consider when decomposing arrival curves into token buckets and service curves into rate latencies for individual analysis of every pair – a situation that does not constitute a problem for the algebraic PMOO analysis.

We saw in Section 1 that the LP approach does not decompose curves for working with the combinations similarly. Its NP-hardness results from the extension of the partial order modeling the dependencies of starts of backlogged periods to the set of compatible total orders – a different source for combinatorial explosion. The ULP circumvents this step, yet, its computational effort is not evaluated, i.e., it is currently not known how the optimization of the delay bound problem scales with the network size. [9] only states that the (U)LP can be solved quickly in the tandem case. There is, however, not information given for more general feed-forward networks. This is probably due to the current lack of comprehensive tools support for the LP approach.

5 Conclusion and Future Work

In this report, we briefly survey results concerning the accuracy, potential for improvement, and computational effort of current network calculus analyses. Correcting the evaluation found in [8,9,20,7,10] reveals there are knowledge gaps concerning the superiority of the ULP over SFA and especially the PMOO analysis as well as the OBA it was inspired by. The ULP's distinguishing strengths that should manifest in bounds that are superior to the algebraic network calculus analyses are theoretically known (see Section 1), yet, they are not evaluated as of now – neither individually nor in the combination found in the LP analysis and the ULP analysis. Given this problem and the slim prospect to improve accuracy of the ULP with reasonable effort, we conclude that investing in improving the accuracy of algebraic network calculus analyses is currently the more promising approach.

Evaluating the distinguishing factors of the (U)LP analysis thus remains an item for future work. The material currently available [2] (tool support / pre-computed linear programs) allows for an evaluation of unbalanced left-over rates and curves requiring decomposition for PMOO and OBA in tandems as well as the square server graph shown above. However, in order to draw broader conclusions, more comprehensive tool support is required. Especially to gain the most interesting insights on how the OBA and the LP approach compare.

References

1. The Disco Deterministic Network Calculator. http://disco.cs.uni-kl.de/index.php/projects/disco-dnc.
2. Tight bounds in feed-forward Networks: Linear programming and Network Calculus. http://perso.bretagne.ens-cachan.fr/~bouillar/NCbounds/.
3. Steffen Bondorf and Jens B. Schmitt. Statistical Response Time Bounds in Randomly Deployed Wireless Sensor Networks. In *Proc. IEEE LCN*, 2010.
4. Steffen Bondorf and Jens B. Schmitt. The DiscoDNC v2 – A Comprehensive Tool for Deterministic Network Calculus. In *Proc. ValueTools*, 2014.
5. Steffen Bondorf and Jens B. Schmitt. Boosting Sensor Network Calculus by Thoroughly Bounding Cross-Traffic. In *Proc. IEEE INFOCOM*, 2015.
6. Steffen Bondorf and Jens B. Schmitt. Calculating Accurate End-to-End Delay Bounds – You Better Know Your Cross-Traffic. In *Proc. ValueTools*, December 2015.
7. Anne Bouillard. *Algorithms and Efficiency of Network Calculus.* Habilitation thesis, École Normale Supérieure, April 2014.
8. Anne Bouillard, Laurent Jouhet, and Eric Thierry. Tight Performance Bounds in the Worst-Case Analysis of Feed-Forward Networks. Technical Report RR-7012, Institut National de Recherche en Informatique et en Automatique, November 2009.
9. Anne Bouillard, Laurent Jouhet, and Eric Thierry. Tight Performance Bounds in the Worst-Case Analysis of Feed-Forward Networks. In *Proc. INFOCOM*, March 2010.
10. Anne Bouillard and Giovanni Stea. *Worst-Case Analysis of Tandem Queueing Systems Using Network Calculus* in *Quantitative Assessments of Distributed Systems*, chapter 6, pages 129–173. Wiley, April 2015.
11. Anne Bouillard and Eric Thierry. An algorithmic toolbox for network calculus. *Journal Discrete Event Dynamic Systems*, 2008.
12. M. Boyer and C. Fraboul. Tightening end to end delay upper bound for afdx network calculus with rate latency fifo servers using network calculus. In *Factory Communication Systems, 2008. WFCS 2008. IEEE International Workshop on*, pages 11–20, 2008.
13. Marc Boyer, Nicolas Navet, and Marc Fumey. Experimental assessment of timing verification techniques for AFDX. In *Proc. ERTS*, 2012.
14. Marc Boyer, Nicolas Navet, Xavier Olive, and Eric Thierry. The pegase project: precise and scalable temporal analysis for aerospace communication systems with network calculus. In *Proceedings of the 4th international conference on Leveraging applications of formal methods, verification, and validation - Volume Part I*, ISoLA'10, pages 122–136, Berlin, Heidelberg, 2010. Springer-Verlag.
15. Cheng-Shang Chang. *Performance Guarantees in Communication Networks.* Springer, 2000.
16. Rodrigo Ferreira Coelho, Gerhard Fohler, and Jean-Luc Scharbarg. Worst-Case Backlog for AFDX Network with n-Priorities. In *Proc. RTN Workshop*, 2014.
17. Rene L. Cruz. A Calculus for Network Delay, Part I: Network Elements in Isolation. *IEEE Transactions on Information Theory*, 1991.
18. Fabrice Frances, Christian Fraboul, and Jérôme Grieu. Using Network Calculus to Optimize AFDX Network. In *ERTS*, 2006.
19. Nicos Gollan, Frank A. Zdarsky, Ivan Martinovic, and Jens B. Schmitt. The DISCO Network Calculator. In *14th GI/ITG Conference on Measurement, Modeling, and*

Evaluation of Computer and Communication Systems (MMB 2008), Dortmund, Germany, March 2008. GI/ITG.

20. Laurent Jouhet. *Algorithmique du Network Calculus*. PhD thesis, École Normale Supérieure de Lyon, November 2012.
21. Petr Jurcik, Anis Koubâa, Ricardo Severino, Mário Alves, and Eduardo Tovar. Dimensioning and Worst-Case Analysis of Cluster-Tree Sensor Networks. *ACM Transactions on Sensor Networks*, pages 14:1–14:47, September 2010.
22. Andreas Kiefer. *Computation of Tight Bounds in Networks of Arbitrary Schedulers*. Diploma (Master's) thesis, University of Kaiserslautern, available online: http://disco.cs.uni-kl.de/discofiles/completed_theses/kiefer09.pdf, March 2009.
23. Andreas Kiefer, Nicos Gollan, and Jens Schmitt. Searching for Tight Performance Bounds in Feed-Forward Networks. In *GI/ITG MMB and DFT*, 2010.
24. Anis Koubâa, Mario Alves, and Eduardo Tovar. Modeling and Worst-Case Dimensioning of Cluster-Tree Wireless Sensor Networks. In *Proc. IEEE RTSS*, pages 412–421, December 2006.
25. Jean-Yves Le Boudec and Patrick Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer, 2001.
26. Frank Ruskey. *Combinatorial generation*. 2003.
27. Jens Schmitt, Nicos Gollan, Steffen Bondorf, and Ivan Martinovic. Pay Bursts Only Once Holds for (Some) non-FIFO Systems. In *Proc. IEEE INFOCOM*, April 2011.
28. Jens B. Schmitt and Frank A. Zdarsky. The DISCO Network Calculator – A Toolbox for Worst Case Analysis. In *Proc. ValueTools*, October 2006.
29. Jens B. Schmitt, Frank A. Zdarsky, and Markus Fidler. Delay Bounds under Arbitrary Multiplexing: When Network Calculus Leaves You in the Lurch ... In *Proc. IEEE INFOCOM*, April 2008.
30. Jens B. Schmitt, Frank A. Zdarsky, and Ivan Martinovic. Improving Performance Bounds in Feed-Forward Networks by Paying Multiplexing Only Once. In *Proc. GI/ITG MMB*, 2008.
31. Yaakov L Varol and Doron Rotem. An algorithm to generate all topological sorting arrangements. *The Computer Journal*, 24(1), 1981.

Appendix I: LP Results for the Square Server Graph

LP	10%	20%	30%	40%	50%
1-1	0.54351946	0.59533608	0.65784653	0.73437500	0.82962963
1-2	0.54351946	0.59533608	0.65784653	0.73437500	0.82962963
1-3	0.53796472	0.58285322	0.63653572	0.70156250	0.78148148
1-4	0.53796472	0.58285322	0.63653572	0.70156250	0.78148148
2-1	0.53796472	0.58285322	0.63653572	0.70156250	0.78148148
2-2	0.33825631	0.38422497	0.44019947	0.50937500	0.59629630
3-1	0.54293629	0.59259259	0.65051903	0.71875000	0.80000000
3-2	0.33850416	0.38518519	0.44221453	0.51250000	0.60000000
4-1	0.54293629	0.59259259	0.65051903	0.71875000	0.80000000
4-2	0.54293629	0.59259259	0.65051903	0.71875000	0.80000000
4-3	0.33857611	0.38535860	0.44253073	0.51301757	0.60080321

(a) LP delays for utilizations from 10% to 50%.

LP	60%	70%	80%	90%
1-1	0.95043732	1.10696404	1.31481481	1.62533600
1-2	0.95043732	1.10696404	1.31481481	1.65777147
1-3	0.88134111	1.00851161	1.22839506	1.64103848
1-4	0.88134111	1.00851161	1.22839506	1.64103848
2-1	0.88134111	1.00851161	1.23148148	1.64103848
2-2	0.70758017	0.85949021	1.08518519	1.52082812
3-1	0.89795918	1.01775148	1.20987654	1.58909911
3-2	0.71020408	0.85949021	1.08518519	1.48473465
4-1	0.89795918	1.01775148	1.16666667	1.35537190
4-2	0.89795918	1.01775148	1.20987654	1.58909911
4-3	0.71141627	0.86257579	1.09013014	1.49688479

(b) LP delays for utilizations from 60% to 90%.

Table 4: Square server graph: All LP delays.