Short Paper: Analyzing FIFO-Multiplexing Tandems with Network Calculus and a Tailored Grid Search

Alexander Scheffler, Steffen Bondorf
Distributed and Networked Systems
Ruhr University Bochum, Germany

Jens Schmitt
Distributed Computer Systems (DISCO) Lab
TU Kaiserslautern, Germany

***Abstract*—Safety-critical applications are increasingly deployed on shared networks. Among the features of current standards, there is one prominent, common characteristic: At queuing locations, different applications' traffic flows multiplex in a First-In First-Out (FIFO) fashion. The Network Calculus framework provides several FIFO analyses for computing a bound on the end-to-end delay of a data flow. However, tracing FIFO relations increases the computational cost and an accurate analysis is typically a slow one. Therefore, we propose a two-step heuristic in this paper. We devise a new, fast analysis to rank alternative tandem designs before a more costly analysis is applied to the top-ranked ones. Our new analysis employs a tailored grid search to resolve the FIFO effects between flows. Numerical evaluations show that we create a ranking that is very close to the one by the accurate yet slower FIFO analysis we base our work on.**

I. Introduction

In this paper, we use Network Calculus (NC) for the derivation of bounds on a system design's end-to-end delay. In general, the NC approach is able to compute bounds if the network is modeled by curves. More precisely, each flow's data has to be upper bounded by an arrival curve α and the forwarding service of a server has to be lower bounded by a service curve β, both non-decreasing and zero in the origin (see [1] for details). As service is provided to all the data queued at a server, a service curve is "shared" by the aggregate of flows crossing the server. To compute an end-to-end delay bound of a particular flow, the flow of interest (foi), its service share needs to be computed, subject to the multiplexing and queueing discipline. The NC First-In First-Out (FIFO) analysis is enabled by this left-over service curve theorem [2]:

Theorem 1. *Let server $\mathcal{S}$ offer service curve β. Assume flows* foi *and f_2 with arrival curves α_{foi} and α_2 cross $\mathcal{S}$. Assuming FIFO multiplexing, the left-over service curve for* foi *is*

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

with $[g(x)]^{\uparrow} = \sup_{0\leq z\leq x} g(z)$, the indicator function $1_{\{cond\}}$, and the free FIFO parameter $\theta \in \mathbb{R}^+$.

In some of our examples we abbreviate the above equation by $\beta_{\text{foi}}^{\text{l.o.}} = \beta \ominus_{\theta} \alpha_2$ where we might add an index to the different θs that arise from "subtracting" cross-flows.

Per application of this theorem, an open parameter $\theta \geq 0$ is introduced and needs to be set, which will result in a θ-specific left-over service curve. Deciding on how to set this value remains a challenging task since this local decision has an influence on the global, i.e., end-to-end, delay bound of a flow. Most notably, a method called Least Upper Delay Bound (LUDB) [4] derives all θs and transforms the challenge to set them into solving several Linear Programs (LPs). We have implemented the LUDB analysis for feedforward networks (LUDB-FF) in the NetworkCalculus.org Deterministic Network Calculator (NCorg DNC) [5]. Latest work [3] showed that 95% of the feedforward analysis effort is in solving the LPs, previous work using a tandem-only tool had observed a share of 85% [6]. More importantly, LUDB is dissected in [3] such that finding the θs can be achieved with a different method – namely search methods. While [3] provides a sophisticated solution that aims at rivaling LUDB delay bounds while being faster, we take a different approach here: We aim at being fast and to (mostly) preserve the relation ($\{<, =, >\}$) between delay bounds for any two analyzed tandems (the basic unit of operation of LUDB). To this end, we instantiate the framework established in [3] with a different search method, a grid search (GS) instead of a directed search.

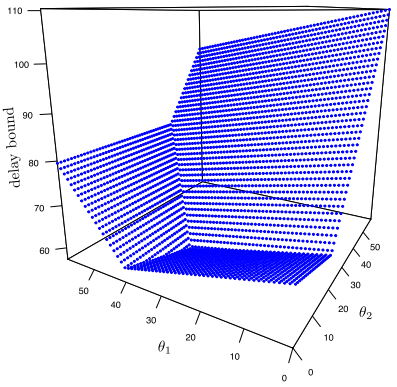


Figure 1. The grid of delay bounds for settings of $\Theta = (\theta_1, \theta_2)$ in a 3-server nested tandem with servers $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3\}$, the foi crossing $\mathcal{S}_1$ to $\mathcal{S}_3$ and two crossflows: f_1 from $\mathcal{S}_2$ to $\mathcal{S}_3$ and f_2 on $\mathcal{S}_3$ (Source: [3], CC BY-SA).

Sampling all combinations of sensible values for all θs basically puts a grid over the problem, see Figure 1. In this paper, we construct a GS for setting the θs with limited number of grid points. While this leads to inaccurate delay bounds, our GS analysis is fast. More importantly, when used to rank design alternatives, our GS will create a ranking that is very close to the one LUDB-FF creates. We can thus devise a new two-step approach consisting of a fast ranking that discards inferior design alternatives early and is then followed by a more accurate but slower analysis of the best-ranked designs.

II. Finding the NC FIFO Parameters: A Grid Search Analysis

This section presents our novel grid search analysis, short GS. We present the method in the context of so-called nested tandems, i.e., servers that can be represented in a line for which every pair of flows are either completely included in each other or have disjoint paths. For example, Figure 2 depicts a nested tandem. For good performance bounds it is crucial to apply the Pay Multiplexing Only Once (PMOO) principle [7] which – for nested tandems – can be briefly summarized as "concatenation before subtraction", i.e., apply min-plus convolution $(\beta_1 \otimes \beta_2)(t) = \inf_{0 \le s \le t} \{\beta_1(t-s) + \beta_2(s)\}$ to service curves β_1, β_2 before Theorem 1. However, the principle can only be achieved for nested tandems. Note that any non-nested tandem needs to be transformed into nested tandems. For simplicity, we focus on the latter in the following.

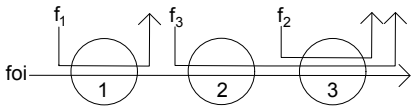


Figure 2. A tandem with 3 servers $(1, 2, 3)$ and 3 crossflows f_1, f_2, f_3 that interfere with the foi in a nested pattern, in short: a *nested tandem* (Source: [3], CC BY-SA).

With regard to the exemplary nested tandem in Figure 2, "concatenation before subtraction" can be achieved by the following computations:

- Left-over service for flow f_2: $\beta^{\text{l.o.}}_{f_2} = \beta_3$
- Left-over service for flow f_3: $\beta^{\text{l.o.}}_{f_3} = \beta_2 \otimes (\beta_3 \ominus_{\theta_2} \alpha_2)$
- Left-over service for flow f_1: $\beta^{\text{l.o.}}_{f_1} = \beta_1$

and finally by combining these partial curves to

$$\beta^{\text{l.o.}}_{\text{foi}} = (\beta_1 \ominus_{\theta_1} \alpha_1) \otimes [(\beta_2 \otimes (\beta_3 \ominus_{\theta_2} \alpha_2)) \ominus_{\theta_3} \alpha_3]$$

which is the left-over service curve for the foi.

This order of operations can be conveniently represented with a nesting tree [4] (see Figure 3). There, the leaf-nodes represent the servers (with respective service curves) and each other node represents a certain crossflow (with its respective left-over curve) except the root which holds the foi. A flow-node f_i is a child of a flow-node f_p iff $\text{Path}(f_i) \subseteq \text{Path}(f_p)$ and $\nexists\, f_k \neq f_p, f_i$ with $\text{Path}(f_i) \subseteq \text{Path}(f_k) \subseteq \text{Path}(f_p)$ where $\text{Path}(f_x)$ in the context of nested tandems can be defined as set of servers that flow f_x crosses.

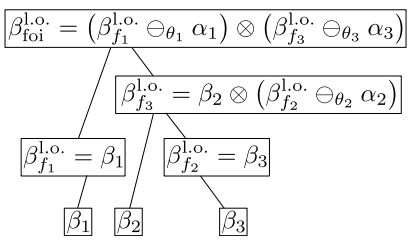


Figure 3. Nesting tree of the tandem in Figure 2 (Source: [3], CC BY-SA).

From this example it is already evident that for nested tandems and set F_x of crossflows with unique paths, we have exactly $|F_x|$ FIFO parameters to set simultaneously in the foi's left-over service $\beta^{\text{l.o.}}_{\text{foi}}$. Having derived $\beta^{\text{l.o.}}_{\text{foi}}$, our GS considers different settings of these parameters. It starts with the initial parameter combination $\Theta = (\theta_1, ..., \theta_{|F_x|}) = (0, ..., 0)$. This results in a left-over service curve $\beta^{\text{l.o.}}$ and corresponding delay bound we call d^{start}, which subsequent parameter combinations try to lower by improving how the GS analysis makes use of the positive impact of FIFO multiplexing has on the actually analyzed system's performance (d^{current} holds the currently best delay bound found by the GS analysis).

The natural question that then arises is, which combinations are worth considering since each parameter θ_i is in $\mathbb{R}^+_0$, i.e., is unbounded, in principle. GS will exclude combinations for which $\theta_i > d^{\text{current}}$ since applying Theorem 1 with such a large θ_i results in a left-over service curve that has a larger latency component than d^{current}. If so, then one can conclude, independent of its service rate, that any resulting delay bound from this service curve will be larger than d^{current}. Hence, combinations for which at least one $\theta_i > d^{\text{current}}$ can be excluded[1]. Having defined a reasonable range for each θ_i to be set, we now look at how to set them. We interpret the combination of choices as an $|F_x|$-dimensional space that we want to search. We partition the space, creating an $|F_x|$-dimensional grid where each point of the grid sets the values for all θ_i, resulting in a new delay bound d. In practice, we approach this as follows: For each θ_i, we consider values between 0 and d^{start} that are equidistant from each other. Allowing for some flexibility when instantiating GS, the distance will be $sp := \frac{d^{\text{start}}}{g-1}$ where parameter g can be freely chosen as long as $g \in \mathbb{N}_{>1}$. We abbreviate these instantiations as GS-g. GS will consider all possible combinations of these values, i.e., all grid points, as long as the condition $\theta_i \le d^{\text{current}}$ holds. The worst-case complexity can be given by the maximal number of combinations that will be considered which is $\mathcal{O}(g^{|F_x|})$, so exponential in the number of crossflows on the nested tandem.

Our GS consists of two algorithms: one to compute the foi's left-over service curve $\beta^{\text{l.o.}}_{\text{foi}}$ on a symbolical level, based on the nesting tree, and one algorithm to execute the grid search steps that instantiates $\beta^{\text{l.o.}}_{\text{foi}}$ by setting the θ_i. Last, note that after every step of setting the θ_i, we can simply compute d.

Algorithm 1 computes the left-over service curve for the foi, given a parameter combination Θ. It executes a post order traversal of the nesting tree by using the concatenation and FIFO left-over service curve Theorem 1. The algorithm starts with the innermost flow from the left (f_1 in Figure 2) according to the nesting tree for which the full service on its path is available (β_1). Similarly, the algorithm considers the next innermost flow (f_2) which is nested into another crossflow f_3. f_3 gets the left-over service (after "subtracting" f_2) at server 3 and full service at server 2 and so on. The algorithm

[1] Also note that the concatenation of two service curves results in a latency that is the sum of both curves' latencies, i.e., the aforementioned argument holds for any flow's θ_i.

Algorithm 1 Left-over service computation on a nested tandem given a parameter combination

Input i, Θ Flow, parameter combination
Output $\beta^{\text{l.o.}}$ Left-over service curve for flow i

procedure COMPUTELEFTOVERSERVICE(i, Θ)
  $\beta^{\text{l.o.}} \leftarrow \delta_0(t) = \begin{cases} 0, & t \leq 0 \\ +\infty, & t > 0 \end{cases}$
  $\forall c \in \text{Children}(i) \backslash F_x$:
    $\beta^{\text{l.o.}} \leftarrow \beta^{\text{l.o.}} \otimes \beta_c$
  $\forall c \in \text{Children}(i) \cap F_x$:
    $\beta_c^{\text{l.o.}} \leftarrow$ COMPUTELEFTOVERSERVICE(c, Θ)
    $\theta_c \leftarrow \Theta(c)$
    $\beta^{\text{l.o.}}(t) \leftarrow \beta^{\text{l.o.}}(t) \otimes ([\beta_c^{\text{l.o.}}(t) - \alpha_c(t-\theta_c)]^{\uparrow} \cdot 1_{\{t>\theta_c\}})$
  return $\beta^{\text{l.o.}}$

will result in an end-to-end left-over service curve for the foi.

Algorithm 2 is the actual grid search method for nested tandems which sets up the different Θ combinations that will be tried out to reduce the delay bound d^{current}. Although this method is shown for nested tandems only, it can be used for the analysis of entire feedforward networks. This can be achieved by replacing the nested LUDB analysis given in [5] with our Algorithm 2. We restrain from doing so in this paper, as the delay bounds derived with GS are not competitive in absolute terms, yet, as we show later, they are very useful for a ranking of design alternatives.

Algorithm 2 Grid search on a nested tandem

Input i, Θ Flow, parameter combination
Output $\beta^{\text{l.o.}}$ Left-over service curve for flow i

procedure GS(i, Θ)
  for $(\theta_i \leftarrow 0; \theta_i \leq d^{\text{current}}; \theta_i \leftarrow \theta_i + sp)$ **do**
    $\Theta(i) \leftarrow \theta_i$
    if $i == |F_x|$ **then**
      $\beta_{\text{foi}}^{\text{l.o.}} \leftarrow$ COMPUTELEFTOVERSERVICE(foi, Θ)
      $d \leftarrow \text{hdev}(\alpha_{\text{foi}}, \beta_{\text{foi}}^{\text{l.o.}})$ ▷ horizontal deviation
      if $d < d^{\text{current}}$ **then**
        $d^{\text{current}} \leftarrow d$
        $\beta^{\text{l.o.}} \leftarrow \beta_{\text{foi}}^{\text{l.o.}}$
    else
      GS($i+1, \Theta$)
  return $\beta^{\text{l.o.}}$

Note that although a higher value for g of the GS algorithm tends to deliver better delay bounds in most cases (see Section III), we now show how two different settings for g, i.e., g_1 and g_2 relate to guarantee delay(GS-g_1) $\geq$ delay(GS-g_2).

Lemma 1. *Let $g_1, g_2 \in \mathbb{N}_{>1}$ with $k \cdot (g_1 - 1) = g_2 - 1$ for $k \in \mathbb{N}$. Then, delay(GS-g_1) $\geq$ delay(GS-g_2) holds.*

Proof. Let $g_1, g_2 \in \mathbb{N}_{>1}$ with $k \cdot (g_1 - 1) = g_2 - 1$ for $k \in \mathbb{N}$. For GS-g_1, note that each $\theta_i \in [0, \frac{1}{g_1-1}d^{\text{start}}, \frac{2}{g_1-1}d^{\text{start}}, ..., \frac{g_1-1}{g_1-1}d^{\text{start}}]$. Similarly, for GS-$g_2$, we have $\theta_i \in [0, \frac{1}{g_2-1}d^{\text{start}}, \frac{2}{g_2-1}d^{\text{start}}, ..., \frac{g_2-1}{g_2-1}d^{\text{start}}]$
$\overset{k\cdot(g_1-1)=g_2-1}{=} [0, \frac{1}{k}\frac{d^{\text{start}}}{g_1-1}, \frac{2}{k}\frac{d^{\text{start}}}{g_1-1}, ..., \frac{k\cdot(g_1-1)}{k}\frac{d^{\text{start}}}{g_1-1}]$.
Hence, every θ_i(GS-g_1)$= \frac{a}{g_1-1}d^{\text{start}}$ with $a \in \mathbb{N} \cap [0, g_1 - 1]$ can be mapped to a θ_i(GS-g_2)$= \frac{b}{g_2-1}d^{\text{start}}$ with $b = a \cdot k \in \mathbb{N} \cap [0, g_2 - 1]$. Thus, $\{\Theta|\text{GS-}g_1\} \subseteq \{\Theta|\text{GS-}g_2\}$ and delay(GS-g_1) $\geq$ delay(GS-g_2). □

III. EVALUATION

Experimental Setup

Our GS implementation is released as part of the NCorg DNC v2.8.1. All results were measured on a Lenovo ThinkStation P620 with AMD Threadripper PRO 3955WX CPU (multithreading, frequency scaling) running Ubuntu 20.04.2 and OpenJDK 16. LPs were solved with IBM CPLEX v20.1.

For our evaluation we created 2086 unique nested tandems with random crossflow interference pattern. The idea is first to create a random tree which will be interpreted as a nesting tree. Then we derive a tandem with nested crossflow interference from the tree. I.e., we reverse the actual analysis steps for tandem generation. Our dataset is available online[2].

Per such random nested tandem we choose a random number n of tree nodes $\in [1, 40]$ and a random maximum degree $\in [1, n-1]$. After creating the respective tree, it might happen that a non-leaf node only has one child that is a non-leaf node, too – these nodes would lead to two flows having the same path on the nested tandem. As such flows are aggregated during the analysis, we merge them into a single non-leaf node as depicted in Figure 4.

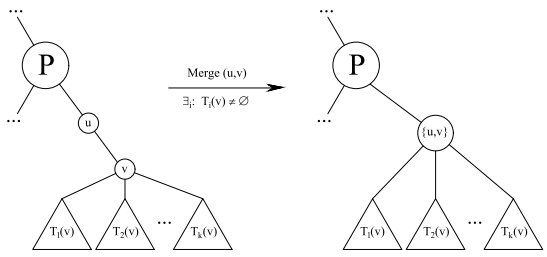
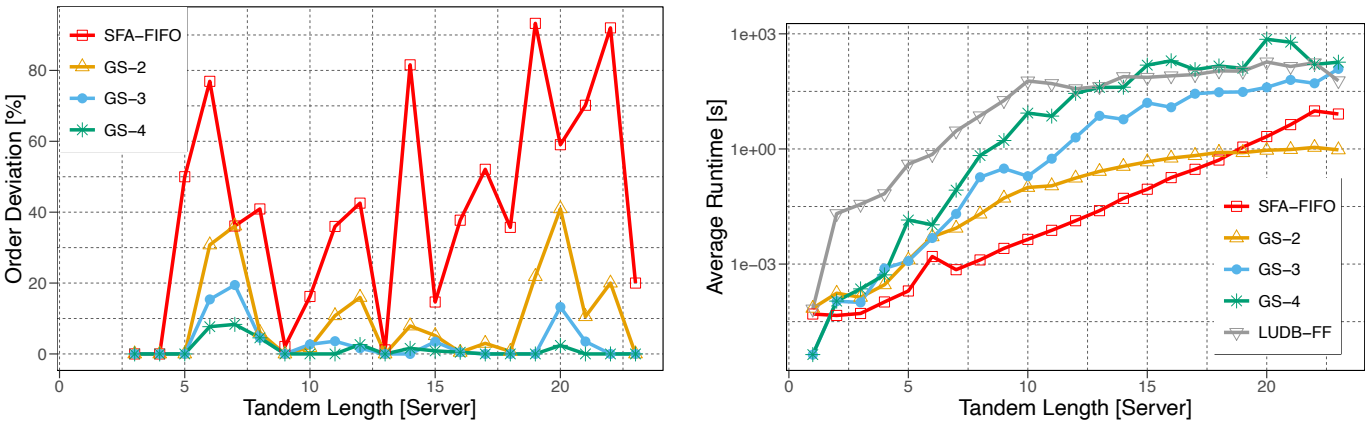


Figure 4. Merging of nodes to get a valid nesting tree: u is a non-leaf node that has only one child, v, which is a non-leaf node, too since it has a non-empty subtree $T_i(v) \neq \emptyset$. Hence, both nodes can be merged. P is the parent node of u, but it has 2 (or more) children so it can't be merged with u since its path is different.

Next, we will interpret the nesting tree as nested tandem where the number of servers as well as the path of the crossflows is already set (by the nesting tree).

As arrival curve α_i for flow i we use the token-bucket shape $\gamma_{\sigma_i,\rho_i}(t) = (\sigma_i + \rho_i \cdot t) \cdot 1_{\{t>0\}}$ with burst $\sigma_i = 1$ and rate $\rho_i = 1$. For node j's the service curve we use the rate-latency shape $\beta_{T_j,R_j}(t) = R_j \cdot [t - T_j]^+$ with $[x]^+ = \max(0, x)$. We use a zero latency ($T_j = 0$) and the rate R_j is set to achieve a utilization of 95%, i.e., $R_j = \frac{\sum_{j \in \text{Path}(f_i)} \rho_i}{0.95}$.

[2] https://github.com/alexscheffler/dataset-itc2022



(a) Ranking deviation compared to LUDB-FF.

(b) Average analysis runtime to compute a delay bound. (y-axis in log scale)

Figure 5. Ranking deviation w.r.t. LUDB-FF and computation runtimes of all analyses. GS-2 and GS-3 are generally faster than LUDB-FF yet not necessarily less precise in ranking design alternatives.

Numerical Results

For each length, we consider all the tandems in our dataset having that length and rank them according to their delay bound. I.e., we assume that the system designer has a fixed amount of servers but some flexibility regarding the flow paths. The most precise analysis we consider is LUDB-FF and we investigate how good the other, less precise but also less costly, analyses are in ranking these tandems.

Figure 5(a) depicts how far off the rankings of GS and SFA-FIFO are compared to LUDB-FF. The order deviation is defined as the difference in positioning of the top-ranked tandem by LUDB-FF compared to the other analyses, i.e., $\frac{\Delta\text{position (top-ranked)}}{\#\text{ tandems}-1}$. We can see that the ranking by SFA-FIFO is rather coarse with a maximal order deviation of up to 93% while GS is more suitable for this task. The reason for this is due to the fact that SFA-FIFO basically computes output arrival curves for the cross-traffic at each server while GS (and LUDB-FF) make use of the concatenation theorem as much as possible. In other words, GS (and LUDB-FF) do not have to compute any output arrival curves for cross-traffic in nested tandems. The preciseness in ranking by GS also depends on the parameter g which tends to increase with larger g as Figure 5(a) suggests. A larger g causes a denser grid and thus more Θ combinations will be evaluated. However, this is only a trend since a higher g does not directly imply that the Θ combinations will be a superset. We have presented a formal statement for which pairs of (g_1, g_2) a proper superset is achieved (see Lemma 1). In fact, for our set of tandems in 23.59% cases GS-3 delivers a better delay bound than GS-4. Hence, we can already observe some outliers where GS-4 leads to a worse top-ranking than GS-3 (see Figure 5(a) with tandem length 12 for example). Also note that the requirements for Lemma 1 are not met for $g_1 = 3$ and $g_2 = 4$ since $2 \nmid 3$. However, we can apply it in case of $g_1 = 2$ and $g_2 \in \{3, 4\}$. That's why GS-2 is not able to provide better bounds than GS-3 or GS-4.

Having seen that we can accurately rank alternative designs w.r.t. LUDB-FF, we now shift our focus to the runtime of these analyses. Figure 5(b) shows the average runtime of each analysis per tandem of a certain length. LUDB-FF is the most costly analysis since it optimizes the Θ setting – for the best delay bound for which several LPs have to be solved. The next computation-intense analysis is GS which needs more computation time the higher the parameter g since then more combinations will be considered (which is upper bounded by $g^{|F_x|}$ with $|F_x|$ being the number of crossflows with distinct paths in the nested tandem). If the GS parameter g is set high, especially in large tandems with many crossflows, then we can observe that the runtime is even worse than LUDB-FF. In our dataset we experience this for GS-4 and tandem-lengths of length 15 or larger. On the other hand, GS-3 is considerably faster than LUDB-FF while still providing a good ranking with an order deviation of only up to 19.45%. The fastest analysis is SFA-FIFO since it fixes any occurring θ_i to a locally derived value (i.e., no concatenation of subsequent servers), so neither a search nor an optimization for these values takes place within the SFA-FIFO analysis. The case where SFA-FIFO gets outperformed by GS-2 in large tandems (length 19 and beyond) can be explained by the recursive backtracking scheme in SFA-FIFO that is required to compute arrival curves for the crossflows on the tandem (again, due to not making use of concatenation).

All in all, we recommend using LUDB-FF as analysis for ranking when the tandems are short with a length of up to 6 nodes – for our dataset, it took, on average, less than 1s to compute the delay bound of a tandem for a fixed length in this range. For larger tandems up to length 8 (11) one can use GS-4 (GS-3) with a ranking quite close to LUDB-FF. For more than 11 nodes we recommend GS-2 to obtain a fast yet still reasonable ranking.

Compared to the approach to run LUDB-FF for all design alternatives, our ranking with a fast GS, followed by a single LUDB-FF analysis, results in a favorable overall runtime. What remains is the question as to why we do not propose

GS as the sole analysis that derives the delay bound. For GS-4, Figure 5(b) already gives a clear answer as its runtime is not outperforming LUDB-FF significantly. For GS-2, GS-3, and SFA-FIFO, we shift our attention to the delay bounds for an answer. Figure 6 shows the delay bound differences of the top-ranked tandem by a certain analysis compared to the LUDB-FF-top-ranked tandem's delay bound[3].

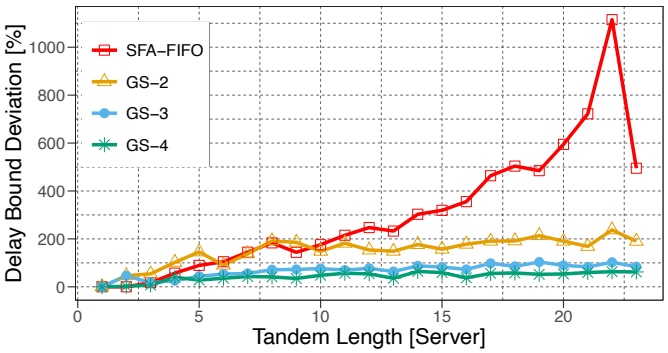


Figure 6. Delay bound deviation of the top-ranked tandems compared to the LUDB-FF-top-ranked tandem's delay.

We can observe that the delay bound deviation of SFA-FIFO [1], [3] (that does not convolve before subtraction) compared to LUDB-FF increases with increasing tandem length. This is due to the fact that SFA-FIFO effectively cuts the crossflows at each server while LUDB-FF makes use of the concatenation theorem. It then needs to compute output arrival curves for crossflows on the nested tandems which results in less accurate delay bounds. Moreover, LUDB-FF optimizes the Θ setting for the whole nested tandem simultaneously while SFA-FIFO sets each θ_i only with local knowledge (see [5] for details) which further tightens the LUDB-FF bounds. Both contribute to the increasing gap between both analyses in longer tandems. SFA-FIFO should not be used due to its largely overestimated delay bounds or only for very small tandems up to length 4 where the delay bound of the top-ranked compared to the LUDB-FF-top-ranked can already be up to 57%.

Concerning GS, remember that it shares the "concatenation before subtraction"-approach with LUDB-FF such that their significant difference lies in the values for θ_i eventually chosen for the delay bound derivation. For the smallest possible $g = 2$, GS brings down the maximal SFA-FIFO[4] gap from more than 1000% to 192% which tends to get reduced further for higher values of g. However, the GS results are still above the LUDB-FF delay bounds by more than one order of magnitude. Hence, we propose a two-stage approach in this paper for fastest and most accurate bounds by combining the GS-based ranking with an LUDB-FF analysis of the best ranked alternative.

[3]The delay bound deviation is defined as relative delay bound between the LUDB-FF delay bound of the top-ranked tandem ranked by LUDB-FF and the respective analysis' delay bound of the top-ranked tandem (ranked by the respective analysis).

[4]Note that our sample size for tandem length 23 is naturally only a fraction of all possible tandems of that length, $5.1 \cdot 10^{-19}\%$ to be precise which is considerably smaller than $1.3 \cdot 10^{-17}\%$ (length 22) for example. For a larger sample, we expect to not see the drop as in Figure 6 but an even larger deviation.

IV. Conclusion

This paper discusses the exploration of network design space w.r.t. delay bounds and with First-In First-Out (FIFO) multiplexing at the queuing locations in the network. We benchmark several Network Calculus (NC) analyses for that task, showing that none fulfills our wish for a fast and accurate ranking. The fast analysis SFA-FIFO resulted in a ranking deviation of up to 93% to the most precise analysis of interest in our evaluation, LUDB-FF. I.e., there is almost no correlation from best to worst tandem when compared to the LUDB-FF-ranking. Therefore, we contribute a new analysis for ranking of tandems called GS.

GS's ranking of alternatives is close to the one by LUDB-FF – we achieve a worst-case order deviation of no more than 41% – while GS is considerably faster. Moreover, the precision of the ranking with GS can be tuned with a parameter g which comes with exactly the tradeoff we are aiming at: Increasing g tends to yield a better ranking but at the cost of a larger analysis runtime. Our numerical evaluation reveals that values for g in the range of 2-4 are most suitable for striking a reasonable balance between quality of the ranking and runtime.

Future Extensions

The presented GS-ranking can be easily interfaced with other analyses. For example, if the delay bound itself becomes important yet the computational demand needs to be kept at bay. Then, LUDB-FF can analyze the $n \geq 1$ alternatives that were ranked top by GS. I.e., the tradeoff between delay bound tightness and effort can be steered via the n parameter. Similarly, top n alternatives can be used to create good quality training sets of restricted size for, e.g., reinforcement learning applied to finding flow prolongations [8].

References

[1] J.-Y. Le Boudec and P. Thiran, *Network calculus: a theory of deterministic queuing systems for the internet*. Springer LNCS, 2001, vol. 2050.

[2] R. L. Cruz, "SCED+: Efficient management of quality of service guarantees," in *Proc. of IEEE INFOCOM*, 1998.

[3] A. Scheffler, J. B. Schmitt, and S. Bondorf, "Searching for upper delay bounds in FIFO multiplexing feedforward networks," in *Proc. of RTNS*, 2022.

[4] L. Lenzini, E. Mingozzi, and G. Stea, "A methodology for computing end-to-end delay bounds in FIFO-multiplexing tandems," *Performance Evaluation*, vol. 65, no. 11-12, pp. 922–943, 2008.

[5] A. Scheffler and S. Bondorf, "Network calculus for bounding delays in feedforward networks of FIFO queueing systems," in *Proc. of QEST*, 2021.

[6] L. Bisti, L. Lenzini, E. Mingozzi, and G. Stea, "Estimating the worst-case delay in FIFO tandems using network calculus," in *Proc. of ICST Valuetools*, 2008.

[7] J. B. Schmitt, F. A. Zdarsky, and I. Martinovic, "Improving performance bounds in feed-forward networks by paying multiplexing only once," in *Proc. of GI/ITG MMB*, 2008.

[8] F. Geyer, A. Scheffler, and S. Bondorf, "Tightening network calculus delay bounds by predicting flow prolongations in the FIFO analysis," in *Proc. of IEEE RTAS*, 2021.