Virtual Cross-Flow Detouring in the Deterministic Network Calculus Analysis

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

Fabien Geyer
Technical University of Munich | Airbus CRT
Munich, Germany

***Abstract*—Deterministic Network Calculus (DNC) is commonly used to compute bounds on the worst-case communication delay in data networks. It provides various analyses to derive these bounds from a model, giving different tradeoffs between accuracy and analysis efficiency. Improving the tradeoff led to increasingly complex algorithms. We set out to design a novel DNC algorithm that is of low complexity while still providing competitive delay bounds. To achieve this goal, we make use of the insight that added pessimism in the model can alleviate more severe limitations of the DNC analysis. To that end, we introduce the concept of *virtual cross-flow detouring* where data flows are assumed to cross additional servers. Ultimately, we provide a heuristic that is simple, fast and high-quality. We show in numerical evaluations that our detouring not only provides a competitive alternative, it also outperforms current algebraic algorithms' delay bounds for >50% of analyzed flows.**

I. Introduction

The correctness of many safety-critical applications is based upon formally verifying upper bounds on the end-to-end delay of data communication. They ensure proper functionality of the system. Deterministic Network Calculus (DNC) is a mathematical framework that has been applied for this verification in a wide variety of applications such as virtual machine placement in data-centers [1], in aerospace for the certification of fly-by-wire avionics [2], or admission control in self-modeling sensor networks [3]. To achieve valid bounds in the DNC framework, the analysis computing them adds some pessimism along the way. This may lead to over-provisioning of network resources and should thus be minimized. There have been efforts to extend the DNC with new results accordingly [4, 5, 6, 7] as well as efforts to recombine existing results to mitigate addition of pessimism [8, 6, 7]. Both of these streams of improvements have resulted in ever more complex algorithms; the tradeoff between complexity and quality became an active research topic in DNC. E.g., this tradeoff was improved by technical advances [9, 10], using machine learning in heuristics [11, 12], or recombination with other known results [13, 14]. In this paper, we will present a novel result called *virtual cross-flow detouring*, in short *detouring*. We integrate detouring into DNC to create a low-complexity analysis algorithm. The result of this integration is not only a considerably less complex and faster to execute heuristic. Its delay bounds also achieve a high level of quality, even exceeding those of the currently best fast analysis algorithm in the majority of our samples.

ISBN 978-3-903176-28-7 ©2020 IFIP

The DNC framework consists of two parts: modeling and analysis. A minimal DNC model provides the network topology and functions that either bound resource availability or demand at queueing locations (service curves and arrival curves). A DNC analysis has the objective to derive a bound on a specific flow's end-to-end delay when crossing the modeled network. The complexity in computing accurate delay bounds arises from the following characteristics of a minimal model:
- Arrival curves are provided per-flow at network entrance.
- Service curves bound the aggregate forwarding capability.

Yet, the DNC analysis aims to compute per-flow delay bounds.

In this paper, we propose virtual cross-flow detouring as an addition to existing analyses. Detouring defines a new degree of freedom in the search for the best tradeoff between length of analyzed server sequences (tandems) and flow aggregation. The main idea is that, if a cross-flow is detoured over (parts of) another flow's path, both can be treated by the analysis algorithm as an aggregate on a longer tandem. Despite the additional load at servers to be detoured over, this approach attains improved, valid delay bounds under certain conditions.

The addition of pessimism is not an entirely novel idea, a proof of concept that is considerably more restrictive than our detouring was presented in [15, 16]. This work proposed to prolong flows over the end of their respective paths, not to add servers at any location. The proposed exhaustive prolongation algorithm has two main characteristics: it is nearly infeasible to execute and the improvements to delay bounds are negligible. Secondly, focusing on longer tandems was also attempted in the recent literature [7]. While this can indeed lead to improved delay bounds, it becomes forbiddingly complex and infeasible to execute, too. We provide the first algorithm that makes ideas from both these concepts feasible to execute, even in combination, and without the use of techniques that do not allow for traceability of the solution process like optimization [5, 17] or machine learning [18, 11]. Despite the necessary measures to reduce computational complexity of the proposed algorithm, we observed that more than $50\,\%$ of delay bounds improved in every single network we analyzed (taken from [6]). Execution times increased by $42.6\,\%$ when adding detouring to a known DNC analysis, yet, creating a heuristic that is vastly faster than other similarly accurate analyses.

The paper is structured as follows: Section II presents the DNC background. In Section III, we provide the virtual cross-flow detouring idea and a heuristic. Section IV benchmarks against existing analyses before Section V draws conclusions.

II. Deterministic Network Calculus Background

An extensive treatment of DNC can be found in [19, 20]. For brevity, we only presents the required background to understand virtual cross-flow detouring. DNC builds non-negative, wide-sense increasing functions that are used to lower bound resource availability guarantees (service curve β) or upper bound demand (arrival curve α), both in interval time. We abbreviate affine arrival curves (so-called token buckets) as $\alpha = \gamma_{r,b}(t) = \{rt + b\} \cdot \mathbb{1}_{t>0}$ and affine service curves (so-called rate latencies) as $\beta = \beta_{R,T}(t) = R \cdot \max\{0, T-t\}$.

We assume three further properties that are not explicitly modeled by these curves:

- order of data within a flow will not change (FIFO per flow),
- no knowledge about flow multiplexing in a server's queue is present (blind multiplexing of flows),
- multiplexing with cross-flows impacts the analyzed flow once per shared path (Pay Multiplexing Only Once, PMOO).

Hence, it is beneficial to analyze a flow over a long sequence of servers to capture the effect of the PMOO property. The work of [4] proposed an analysis implementing the PMOO property under the first two assumptions, known as PMOO Analysis (PMOOA). In this work, we extensively apply the following computation. It lower bounds the minimum residual service on a sequence of servers.

Theorem 1: The affine PMOOA left-over service curve $\beta^{\text{l.o.}} = \beta_{R^{\text{l.o.}},T^{\text{l.o.}}}$ for an analyzed flow of interest (foi) on a tandem of servers $\mathcal{T}$ is computed as

$$R^{\text{l.o.}} = \bigwedge_{s \in \mathcal{T}} \left(R_s - \sum_{(f \in s) \setminus \text{foi}} r^f \right)$$

$$T^{\text{l.o.}} = \frac{\sum_{(f \in \mathcal{T}) \setminus \text{foi}} b^f + \sum_{s \in \mathcal{T}} \left(T_s \cdot \sum_{(f \in s) \setminus \text{foi}} r^f \right)}{R^{\text{l.o.}}} + \sum_{s \in \mathcal{T}} T_s$$

where $\wedge$ is the minimum, $s \in \mathcal{T}$ is a server on tandem $\mathcal{T}$, $f \in \mathcal{T}$ is a flow on $\mathcal{T}$, and $f \in s$ is a flow at server s.

For the computation of a single server's $\beta^{\text{l.o.}}$, we usually abbreviate the computation with the binary operator $\ominus$ to $\beta \ominus \alpha$. The two major known issues of Theorem 1 are:

1) Cross-flow bursts are served with the foi path's minimum left-over service rate $R^{\text{l.o.}}$, i.e., with the minimum across the entire tandem. Thus, $\beta^{\text{l.o.}}$ cannot benefit from increased service curves for individual servers on the tandem [5].

2) Arrival curves are required per set of cross-flows sharing one subpath of the foi path. This is known as segregation [7].

A recent, very accurate analysis called Tandem Matching (TMA, [6]) proposed an exhaustive search among all possible tradeoffs between these two aspects. Thus, TMA applies PMOOA left-over service computations. Large analysis complexity stems from finding the best tandems (tandem matching), not Theorem 1. Recently, it was proposed to replace the exhaustive search with machine learning predictions [11].

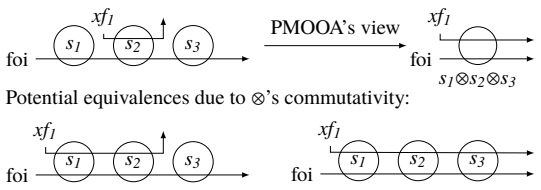


Figure 1: Potentially equivalent networks for a PMOOA due to commutativity of the DNC convolution $\otimes$.

The intuition behind PMOOA is simple: convolve the tandem of servers into a single system before subtracting the interfering arrivals. While this implements the PMOO principle, the underlying convolution causes issue 1. Convolution is commutative and therefore multiplexing cannot be localized to exactly the server where it occurs. In this paper, we look at PMOOA's issue 1 from a different angle: In Figure 1 we might be able to detour cross-flow xf_1 over server s_1 or s_3 without incurring a penalty for this added pessimism of distributing server resources among more flows. The conditions for such a PMOOA-equivalence (four equal $\beta^{\text{l.o.}}$ in Figure 1 by Theorem 1) of the tandems are: all curves are affine with $T_{s1} = T_{s3} = 0$, $R_{s_2} \geq R_{s_1}$, $R_{s_2} \geq R_{s_3}$.

III. Virtual Cross-Flow Detouring

Detouring is a simple extension of the DNC analysis that adds servers to cross-flows' paths (see Figure 2). Similar to improved speed of existing servers (issue 1), adding servers cannot improve the result of Theorem 1. Yet, it can have a positive impact by allowing the analysis to derive better network-internal arrival curves – output bounds derived as $\alpha \oslash \beta$ where $\oslash$ is the min-plus deconvolution [19]. We use output bounds to get the arrivals of cross-flows at a location where these flows interfere with a different, analyzed flow.

A. Detouring at the Front

In Figure 2, we are interested in a bound on the output after the 2-server tandems. We aggregate flows as much as possible and with $\alpha \oslash \beta_x \oslash \beta_y = \alpha \oslash (\beta_x \otimes \beta_y)$ ([19] Thm 3.1.12), the computation can progress server by server without losing the PMOO-benefits when convolving their service curves first. In Figure 2a, the output of server s_1 is bounded by

$$\alpha_{s_1}{}' = \gamma_{r^{f_1},b^{f_1}} \oslash \beta_{R_{s_1},T_{s_1}} = \gamma_{r^{f_1},b^{f_1}+r^{f_1}T_{s_1}}$$

and the output bound of server s_2, the final result, is

$$(\alpha_{s_1}{}' + \alpha_{s_2}) \oslash \beta_{s_2} = \gamma_{r^{f_1}+r^{f_2},b^{f_1}+b^{f_2}+r^{f_1}(T_{s_1}+T_{s_2})+r^{f_2}T_{s_2}}.$$

For this flow detouring at the front ($\text{Det}_{\text{front}}$) version in Figure 2b, we assume both flows already multiplexed in

(a) Original network (b) Flow detouring at the front

Figure 2: Adding servers at the front of a flow's path.

s_1's queue and therefore we compute the output bound for the aggregate of both flows. In general, aggregation benefits the DNC analysis for this reason, yet, as we will see, flow detouring at the front introduces pessimism that outweighs the aggregation benefits in the output bound computation, too:

$$\begin{aligned}\alpha_{s_1}^{\text{Det}_{\text{front}}\prime} &= (\gamma_{r^{f_1},b^{f_1}} + \gamma_{r^{f_2},b^{f_2}}) \oslash \beta_{R_{s_1},T_{s_1}} \\ &= (\gamma_{r^{f_1},b^{f_1}} \oslash \beta_{R_{s_1},T_{s_1}}) + (\gamma_{r^{f_2},b^{f_2}} \oslash \beta_{R_{s_1},T_{s_1}}) \\ &= \gamma_{r^{f_1},b^{f_1}+r^{f_1}T_{s_1}} + \gamma_{r^{f_2},b^{f_2}+r^{f_2}T_{s_1}}\end{aligned}$$

Note, that we applied distributivity of $\oslash$ over $+$ for rate-latency service and token-bucket arrivals [3]. This is not necessary at this point but it results in two separate output bound computations, nicely illustrating the increase of f_2's maximum burstiness arriving at s_2 by $r^{f_2}T_1$. The disadvantage is carried over to the output of server s_2 of Figure 2b:

$$\begin{aligned}\alpha_{s_2}^{\text{Det}_{\text{front}}\prime} &= \alpha_{s_1}^{\text{Det}_{\text{front}}\prime} \oslash \beta_{s_2} \\ &= (\gamma_{r^{f_1},b^{f_1}+r^{f_1}T_{s_1}} + \gamma_{r^{f_2},b^{f_2}+r^{f_2}T_{s_1}}) \oslash \beta_{R_{s_2},T_{s_2}} \\ &= \gamma_{r^{f_1},b^{f_1}+r^{f_1}(T_{s_1}+T_{s_2})} + \gamma_{r^{f_2},b^{f_2}+r^{f_2}(T_{s_1}+T_{s_2})}\end{aligned}$$

We have seen that flow detouring at the front will result in worse bounds – even if flows can be aggregated for output bounding. Next we show how flow detouring at the front can lead to better bounds nonetheless.

B. Detouring of Cross-traffic: Improving Bounds Nonetheless

In this section, we demonstrate that detouring over a server added to the middle of a flow's path can improve delay bounds derived by DNC despite the problems illustrated in the underlying detouring at the front.

We investigate the network shown in Figure 3a [7]. The output of cross-flows xf_1 and xf_2 after server s_0 needs to be bounded. Current DNC alternatives at s_0 are

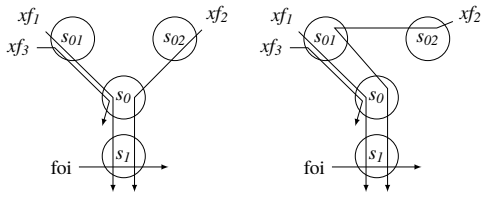
• *Maximize aggregation.* This enforces a hop-by-hop analysis due to the fork above s_0 and xf_1 cannot benefit from PMOOA when subtracting the impact of xf_3 on the tandem s_{01}, s_0.

• *Maximize tandem length.* This strategy allows for implementation of the PMOO principle, yet, it requires to segregate xf_1 and xf_2. I.e., these flows are analyzed individually and assume mutually exclusive worst-case system behavior at s_0.

With the PMOOA (Theorem 1) and virtual flow detouring in the analysis, we can benefit from the PMOO principle when subtracting cross-flow xf_3 and from aggregating xf_1 and xf_2 (see Figure 3b). Detouring is paid for by a different penalty (cf. $\text{Det}_{\text{front}}$ in Section III-A), creating a new tradeoff that can beat the two existing strategies. The longer tandem will be able to hold more data in transit (added pessimism), and the issues of PMOOA prevent introduction of dangerous optimism by making it lack the ability to distribute load in a better way than on the original tandem. In summary, it is key to virtually detour a flow over its cross-flows such that the PMOOA has an impact. Then, the analysis of a more pessimistic setting can indeed result in better output bounds as we illustrate on Figure 3b next. For readability, we skip the s and f labels.

1) *Detouring $\gamma^{Detouring} = \gamma_{r^{Detouring},b^{Detouring}}$ arrivals at s_1*

$$\gamma^{\text{Detouring}} = ((\alpha_2 \oslash \beta_{02}) + \alpha_1) \oslash ((\beta_{01} \otimes \beta_0) \ominus \alpha_3)$$



(a) Original network (b) Network after detouring

Figure 3: Cross-flow detouring in the network [7] known to benefit from a longer tandem analysis of xf_1.

$$\begin{aligned}&= ((\gamma_{r_2,b_2} \oslash \beta_{R_{02},T_{02}}) + \gamma_{r_1,b_1}) \\ &\quad \oslash ((\beta_{R_{01},T_{01}} \otimes \beta_{R_0,T_0}) \ominus \gamma_{r_3,b_3}) \\ &= (\gamma_{r_2,b_2+r_2T_{02}} + \gamma_{r_1,b_1}) \oslash (\beta_{R_{01}\wedge R_0,T_{01}+T_0} \ominus \gamma_{r_3,b_3}) \\ &= \gamma_{r_1+r_2,b_1+b_2+r_2T_{02}} \oslash \beta_{(R_{01}\wedge R_0)-r_3,T_{01}+T_0+\frac{b_3+r_3(T_{01}+T_0)}{(R_{01}\wedge R_0)-r_3}}\end{aligned}$$

with $r^{\text{Detouring}} = r_1 + r_2$ and

$$\begin{aligned}b^{\text{Detouring}} &= b_1 + b_2 + r_2T_{02} \\ &\quad + (r_1 + r_2)\left(T_{01} + T_0 + \frac{b_3 + r_3(T_{01} + T_0)}{(R_{01} \wedge R_0) - r_3}\right) \\ &= b_1 + b_2 + r_2T_{02} \\ &\quad + r_2T_{01} + (r_1 + r_2)T_0 + r_1T_{01} + (r_1 + r_2)\frac{b_3 + r_3(T_{01} + T_0)}{(R_{01} \wedge R_0) - r_3}\end{aligned}$$

From the literature, we get two further valid bounds for the arrival of $xf_1 + xf_2$ at s_1, derived as follows:

2) *Aggregate Arrival Bounding's γ^{AggrAB} [8] in Figure 3a*

$$\begin{aligned}\gamma^{\text{AggrAB}} &= \gamma_{r_1+r_2,\, b_1+b_2+r_1T_{01}+r_2T_{02}} \\ &\quad + r_1\tfrac{b_3+r_3T_{01}}{R_{01}-r_3} + (r_1+r_2)\left(T_0 + \tfrac{b_3+r_3(T_{01}+T_0)}{R_0-r_3}\right)\end{aligned}$$

3) *Segregated Arrival Bounding's γ^{SegrAB} [7] in Figure 3a*

$$\begin{aligned}\gamma^{\text{SegrAB}} &= \gamma_{r_1+r_2,\, b_1+b_2+r_1T_{01}+r_2T_{02}+(r_1+r_2)T_0} \\ &\quad + r_1\tfrac{b_2+b_3+r_3T_{01}+(r_2+r_3)T_0}{(R_{01}-r_3)\wedge(R_0-r_2-r_3)} + r_2\tfrac{b_1+b_3+(r_1+r_3)T_0}{R_{02}\wedge(R_0-r_1-r_3)}\end{aligned}$$

We see that all three alternatives compute the same arrival rate at s_1, $r_1 + r_2$, and that there are common terms in the arrival burstiness, $b_1 + b_2 + r_1T_{01} + r_2T_{02} + (r_1 + r_2)T_0$. We assume a network with homogeneous rates $R_0 = R_{01} = R_{02} =: R$, $r_1 = r_2 = r_3 =: r$ and compare the remaining burst terms.

• $\gamma^{\text{Detouring}} < \gamma^{\text{AggrAB}}$:

$$\begin{aligned}&rT_{01} + 2r\frac{b_3 + r(T_{01} + T_0)}{R - r} < \\ &\quad r\frac{b_3 + rT_{01}}{R - r} + 2r\frac{b_3 + r(T_{01} + T_0)}{R - r} \\ \Leftrightarrow\quad &T_{01}(R - 2r) < b_3\end{aligned}$$

• $\gamma^{\text{Detouring}} < \gamma^{\text{SegrAB}}$:

$$\begin{aligned}&rT_{01} + 2r\frac{b_3 + r(T_{01} + T_0)}{R - r} < \\ &\quad r\frac{b_2 + b_3 + rT_{01} + 2rT_0}{R - 2r} + r\frac{b_1 + b_3 + 2rT_0}{R - 2r} \\ \Leftrightarrow\quad &(R - r)T_{01} - 2rT_0 < b_1 + b_2\end{aligned}$$

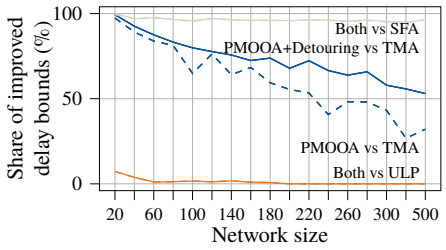


Figure 4: Delay bound improvements.

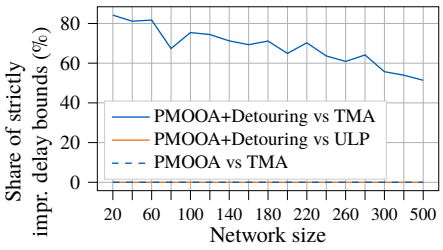


Figure 5: Strict delay bound reductions.

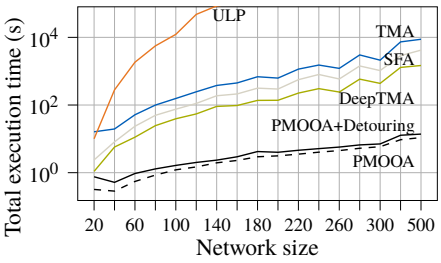


Figure 6: Execution times.

In [7], parameters were fixed to show $\gamma^{\text{SegrAB}} < \gamma^{\text{AggrAB}}$: latencies are 0, $b_1 = b_2 = 0$. With the former, virtual detouring will attain better results. Adding the latter, its results will equal the SegrAB. The remaining parameters are constant. In a homogeneous network modeled with affine curves, detouring can theoretically outperform the AggrAB and SegrAB output bounding strategies by an arbitrarily large margin.

C. Application to Large Feedforward Networks

In Figure 3, there is only a single sensible detouring alternative. In larger networks with flows taking different paths, there may be several virtual detouring alternatives in order to allow for the PMOO principle's application to different sets of their cross-flows – the amount of alternatives certainly increases with the network size. This creates potential for a combinatorial explosion. Therefore, we will provide a simple yet effective heuristic to select one single detouring path.

From TMA [6] we learn the reason for effort in the DNC analysis: The analysis is executed with recursive backtracking [8], starting from the foi and backtracking over cross-flows. Bounds on arrivals of these cross-flows are not known before the backtracking terminated and results for the piled up recursion levels are computed. During backtracking, these still missing results are yet required to find the best bound computation for a recursion level. Thus, the exhaustive search keeps all alternatives alive until all information is known.

From DeepTMA [11] we learn that we need a non-exhaustive heuristic. Our detouring heuristic only operates on the invariant data available at any location in the network: service curves and the sole presence of flows (without arrival curves / output bounds). The former is not of much use without arrival curves and thus, we use the latter. We interpret presence of flows as potential for cross-flow aggregation and additional servers to detoured over. The decision on detouring can be easily done server-by-server, embedded into the backtracking: *At any server, simply check all in-links and detour over the one that has the most flows on it*. If there are multiple alternatives, randomly choose one of them. Note, that the random choice we opted for in our heuristic negatively impacts reproducibility of our evaluation results. But any other tie-breaker would increase the complexity of the detouring heuristic. We terminate the detouring-path search when one of these conditions is met:
i) there is no cross-flow or analyzed flow left to prolong over
ii) bounds become infinite (long-term service $<$ arrivals).

IV. Numerical Evaluation

A. Evaluation Setup

For benchmarks, we compute delay bounds in the networks from [6][1] (20 to 500 devices, 152 to 7504 flows). We mainly compare PMOOA with detouring (PMOOA+Detouring), the Tandem Matching Analysis (TMA) [6] paired with the Burst Cap (BC) [13] mechanism and the optimization-based ULP analysis [17]. The basis for our PMOOA+Detouring implementation is the open-source NetCal DNC v2.6 [21][2].

B. Improvement over Other Analyses

We compare in this section the resulting end-to-end delays produced by PMOOA+Detouring and compare them against TMA and SFA [19, 20]. We aim to derive the best delay bound for every flow. There are no semantics assigned; the flow with the best improvement could be the most important one.

We first evaluate how many flows have their end-to-end delay bound matched or reduced with the use of detouring, compared to the other analyses:

$$delay_{PMOOA+Detouring} \leq delay_X$$

Results are presented in Figure 4. PMOOA+Detouring is able to match or improve delay bounds of TMA for at least $53.0\,\%$ of the analyzed flows – this lowest value is obtained in the largest network. In contrast to PMOOA without Detouring, which matches at most $26.7\,\%$ of the analyzed flows compared in the largest network, the addition of detouring is beneficial. As expected, the use of detouring has thus almost no impact on the existing relation to SFA and ULP delay bounds [6].

As a second metric, we evaluate how many flows have their end-to-end delay bound strictly reduced, namely we use a strict inequality compared to before:

$$delay_{PMOOA+Detouring} < delay_X$$

Results are presented in Figure 5. As expected, PMOOA does not produce tighter bounds than TMA. Using detouring results in strictly tighter bounds for more than $51.4\,\%$ of the analyzed flows, meaning that our approach is able to outperform the tightness of TMA for more than half of the evaluated flows. Compared to ULP, no strict improvement is achieved, again an expected result for the same reasons as above.

[1] Available at: `https://github.com/NetCal/DNC_experiments`
[2] Available at: `https://github.com/NetCal/DNC`

C. Relative Change

We illustrated that our heuristic can match or even outperform PMOOA and TMA+BC. As PMOOA+Detouring competes with segregate arrival bounding of [7] (SegrPMOO), we benchmark against the results available for TMA+SegrPMOO. We compare delay bounds with the *relative change* metric:

$$RelativeChange_x = \left(delay_{Detouring} - delay_x\right) / delay_x$$

Results are presented in Figure 7 for the only two networks from the dataset that can be analyzed with TMA+SegrPMOO.

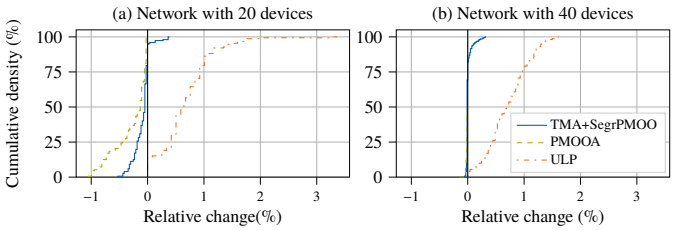


Figure 7: Relative change of PMOOA+Detouring compared to TMA+SegrPMOO, PMOOA and ULP.

As illustrated in Figure 7a, detouring can result in the reduction of end-to-end delay bounds of up to $1\,\%$ compared to PMOOA, or up to $0.5\,\%$ compared to TMA+SegrPMOO. This effect is less visible in Figure 7b, but detouring still results in $0.2\,\%$ in the best case.

D. Execution Time

In order to illustrate the impact of PMOOA+Detouring's small complexity, we compare the execution time of our heuristic with other analyses. We benchmark the different analyses by measuring the execution time per network, i.e. the overall elapsed wall clock time of an analysis for all the flows in a network size. Execution times presented here do not include the time to model a network or derive the server graph or adding the flows. These are not in focus of this work and they are the shared prerequisite for all the evaluated analyses.

Results are presented in Figure 6. All execution times were measured on the same machine with an Intel Xeon CPU E5420 at $2.50\,\mathrm{GHz}$. For DeepTMA, no GPU acceleration was used. For ULP, CPLEX was used for solving the linear program.

PMOOA+Detouring takes advantage of the efficiency of PMOOA and outperforms all other analyses by at least two orders of magnitude, except pure PMOOA of course. On average, the addition of detouring (with a PMOOA on the detour) resulted in an increase of only $42.6\,\%$ in average, indicating that detours are rather short. Those results illustrate that the computational cost of using detouring is negligible compared to the gains in tightness illustrated earlier.

V. Conclusion

In this paper, we contribute *virtual cross-flow detouring* (detouring) to deterministic network calculus. We show that it is a powerful extension to the PMOOA analysis, resulting in delay bounds matching or even outperforming the state-of-the-art analyses that are considerably more involved. Our contribution is based on the counter-intuitive idea of adding pessimism to the model. Due to a previously frowned upon characteristic of PMOOA, we can compute better delay bounds nonetheless.

Our evaluation shows that detouring is able to outperform the previously best analyses, in particular TMA. Delay bounds of more than $50\,\%$ of the analyzed flows improved compared to TMA. This is achieved by only a small addition to PMOOA's execution time of $42.6\,\%$ on average. That makes our proposed heuristic not only comparable with TMA w.r.t. the computed delay bounds but also faster by two orders of magnitude.

References

[1] K. Jang, J. Sherry, H. Ballani, and T. Moncaster, "Silo: Predictable message latency in the cloud," in *Proc. of ACM SIGCOMM*, 2015.

[2] F. Geyer and G. Carle, "Network engineering for real-time networks: comparison of automotive and aeronautic industries approaches," *IEEE Communications Magazine*, vol. 54, no. 2, pp. 106–112, 2016.

[3] S. Bondorf and J. B. Schmitt, "Boosting sensor network calculus by thoroughly bounding cross-traffic," in *Proc. of IEEE INFOCOM*, 2015.

[4] 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.

[5] J. B. Schmitt, F. A. Zdarsky, and M. Fidler, "Delay bounds under arbitrary multiplexing: When network calculus leaves you in the lurch…," in *Proc. of IEEE INFOCOM*, 2008.

[6] S. Bondorf, P. Nikolaus, and J. B. Schmitt, "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)*, vol. 1, no. 1, pp. 16:1–16:34, 2017.

[7] ——, "Catching corner cases in network calculus – flow segregation can improve accuracy," in *Proc. of GI/ITG MMB*, 2018.

[8] S. Bondorf and J. B. Schmitt, "Calculating accurate end-to-end delay bounds – you better know your cross-traffic," in *Proc. of EAI ValueTools*, 2015.

[9] N. Luangsomboon, R. Hesse, and J. Liebeherr, "Fast min-plus convolution and deconvolution on GPUs," in *Proc. of EAI ValueTools*, 2017.

[10] A. Scheffler, M. Fögen, and S. Bondorf, "The deterministic network calculus analysis: Reliability insights and performance improvements," in *Proc. of IEEE Intl. Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks (CAMAD)*, 2018.

[11] F. Geyer and S. Bondorf, "DeepTMA: Predicting effective contention models for network calculus using graph neural networks," in *Proc. of INFOCOM*, 2019.

[12] ——, "On the robustness of deep learning-predicted contention models for network calculus," 2019, arxiv:1911.10522.

[13] S. Bondorf and J. B. Schmitt, "Improving cross-traffic bounds in feed-forward networks – there is a job for everyone," in *Proc. of GI/ITG MMB & DFT*, 2016.

[14] A. Mifdaoui and T. Leydier, "Beyond the accuracy-complexity tradeoffs of compositional analyses using network calculus for complex networks," in *Proc. of CRTS Workshop*, 2017.

[15] 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.

[16] S. Bondorf, "Better bounds by worse assumptions – improving network calculus accuracy by adding pessimism to the network model," in *Proc. of IEEE ICC*, 2017.

[17] A. Bouillard, L. Jouhet, and É. Thierry, "Tight performance bounds in the worst-case analysis of feed-forward networks," in *Proc. of IEEE INFOCOM*, 2010.

[18] F. Geyer, "Performance evaluation of network topologies using graph-based deep learning," in *Proc. of EAI ValueTools*, 2017.

[19] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer-Verlag, 2001.

[20] A. Bouillard, M. Boyer, and E. Le Corronc, *Deterministic Network Calculus: From Theory to Practical Implementation*. Wiley, 2018.

[21] S. Bondorf and J. B. Schmitt, "The DiscoDNC v2 – a comprehensive tool for deterministic network calculus," in *Proc. of EAI ValueTools*, 2014.