An Empirical Study of Tightest Network Calculus Analyses for Networks with Multicast Flows

Bruno Cattelan*§, Steffen Bondorf†‖, Alberto E. Schaeffer-Filho‡

*TU Kaiserslautern, Dept. of Computer Science, DISCO | Distributed Computer Systems Lab, Kaiserslautern, Germany
†Ruhr University Bochum, Faculty of Mathematics, Center of Computer Science, Bochum, Germany
‡Federal University of Rio Grande do Sul (UFRGS), Institute of Informatics, Porto Alegre, Brazil

***Abstract*—The Network Calculus (NC) analysis is concerned with deriving an upper bound on the end-to-end delay of data flows. For tighter bounds, the analysis needs to incorporate features of the network model in the best way possible. This has proven to be a non-trivial task, resulting in what seemed to be a succession of improved NC capabilities. However, it was discovered that neither of the two most prominent analyses in the literature is always best. Deriving the tightest delay bound thus became an expensive search for the most beneficial tradeoff between drawbacks of alternative NC analyses. In networks with multicast flows this problem is amplified, yet, while causes for this situation are known, there is no empirical investigation on the ratio between different analyses obtaining the tightest bound. In this paper, we provide an empirical study of the distribution of delay bounds derived with NC analyses that can be used to bound delays in networks with multicast flows. We do so by evaluating differently sized avionics-like networks.**

I. Introduction

Modern network standards like Avionics Full-duplex Ethernet (AFDX) [1] often allow for concepts like virtual links, i.e., data flows that have a single source and multiple destinations. They can be modeled with the established concept of multicast communications in IP networks [2], in short *multicasting*. The avionics domain imposes strict certification requirements on their networks. Foremost, the upper bounds on the end-to-end delay of data communication must be formally verified. Network Calculus (NC) is a mathematical framework capable of deriving such bounds. NC as a methodology is under active research and multiple improvements have been made regarding its modeling power [3], the tightness of derived bounds [4], computational efficiency [5], or multiple of these aspects [6]. Network Calculus was recently extended to analyze networks with multicast flows [7, 8]. This extension inherited an existing drawback of NC: there is a multitude of possibilities to instantiate the NC analysis framework, however, the one leading to tightest bounds is unknown a priori. This problem stems from the insight that none of the two most prominent analyses is strictly best: neither the server-by-server analysis SFA [9] nor the end-to-end analysis PMOO [10]. This even led to an entirely novel, optimization-based NC analysis branch [4, 11] that, in turn, is not yet capable of analyzing multicast flows. The available NC framework extension for analyzing multicasting only exists for the classical algebraic NC [9]. Not only does it inherit the above SFA-vs-PMOO dilemma, it also amplifies it: besides the need to decide a priori, there are three additional, mutually incompatible analysis proceedings to instantiate (UT, EIB, MFF). The initial work on this topic [7] could already show two aspects of interest: *1)* the most trivial multicasting extension of NC is inferior to the other alternatives and *2)* NC for multicasting achieves bounds at least as tight as non-NC analyses methodologies that were newly developed to overcome this previous lack of a multicast analysis [12, 13, 14]. The follow-up work in NC [8] includes an evaluation on an avionics-like network that illustrates the situation between the remaining two ones. The shown results confirm that neither of the two major new NC multicasting analyses can be assumed invariantly better, but the paper does not address this observation. Overall, the task to find the tightest delay bound becomes considerably more involved.

In this paper, we present a notably more comprehensive numerical evaluation of AFDX-like networks. Our aim is to provide insight into the deviation of delay bounds as derived with one of the four most advanced alternative instantiations of the NC framework for multicasting networks. Our empirical results confirm that neither of the two major new NC multicasting analyses can be assumed invariantly better, but we shed insights into the delay bound distributions. To the best of our knowledge, the closest work to ours is an experimental assessment of timing verification techniques for AFDX that predates the NC multicasting extensions [15]. With our new investigation, we shed light onto the potential loss of tightness when choosing a suboptimal NC analysis. This seems particularly important as NC multicasting is currently adopted in the AFDX and factory automation research community that previously developed competing approaches [16]. Moreover, there is continued effort to improve the already existing search for the best unicast analysis: after investigating the exhaustive search [6], machine learning has been proposed to replace this expensive approach with recommendations [17, 18]. Any machine-learning approach requires a training set, i.e., network models and delay bounds, to learn from. The tools and results

§This work was partially supported by the Carl-Zeiss Foundation while in the DISCO Lab at TU Kaiserslautern and it was partially carried out at the Federal University of Rio Grande do Sul (UFRGS).

‖This work was partially carried out during the tenure of a Carl-Zeiss Foundation fellowship in the DISCO Lab at TU Kaiserslautern, partially during the tenure of an ERCIM 'Alain Bensoussan' Fellowship Programme at NTNU Trondheim, Norway, and partially in the Faculty of Mathematics' Center of Computer Science at Ruhr University Bochum.

we present in this paper can provide the foundation for potential future work on a deep learning-assisted NC multicasting analysis. Having these tools at hand, we are also the first to give trends regarding the sensitivity of the analyses to changes in size of the underlying avionics-like topology.

The paper is organized as follows: First, Section II points to NC background theory. Our empirical evaluation follows suit, with Section III detailing the experimental design and Section IV giving the results. Section V concludes the paper.

II. Network Calculus Background

For brevity, we restrict our presentation of the background to providing pointers to relevant literature. A comprehensive treatment of the algebraic network calculus (NC) basics is presented in [9], the later work in [19, 6, 20] develops the state-of-the art analysis techniques for feed-forward networks of work-preserving schedulers under no assumptions on the multiplexing of unicast flows. Note that the feed-forward property must hold for the server graph, i.e., the network of queueing locations of network devices (so-called *servers*). Details on network modeling are presented in [21, 22]. Unicast flows cross a sequence of servers, a so-called *tandem*. Currently, the analysis of tandems can make use of either the Separate Flow Analysis *SFA* (i.e., Pay Bursts Only Once *PBOO* property and order of crossed servers) or the Pay Multiplexing Only Once *PMOO* property [10] but not both simultaneously [4]. Unfortunately, neither is strictly better than the other. Finally, the advanced multicast flow analyses we evaluate in this paper were added to NC in [7, 8]. They are, in fact, analysis frameworks that require instantiation with a tandem analysis – either SFA or PMOO. The frameworks are:

- Unicast Transformation (UT): each multicast flow is transformed to a set of unicast flows.
- Explicit Intermediate Bounds (EIB): multicast flows are cut at their forking locations, an explicitly derived bound on the flow's data characterizes the subflows (cut location-to-sink pairs) taking the subsequent paths.
- Multicast Feed-Forward (MFF): the existing NC feed-forward analysis provides information on the parts of a multicast flow under analysis. It is used to abstract to the unicast subflow (subtandem of entire path) of interest for the current analysis step.

III. Numerical Evaluation Setup

In this section, we provide the experimental design we developed for our numerical evaluation. We aim at full coverage of the details to allow for repeatability and reproducibility.

A. Implementing NC Multicasting Analyses

Our tool of choice for NC analyses is the open-source NetworkCalculus.org DNC Tool[1]. It already provides implementations for arbitrary multiplexing SFA and PMOO analyses in unicast feed-forward networks (matching our assumption that we do not have knowledge about multiplexing behaviour). For UT and EIB, we implemented the network transformation steps that precede the actual analysis with the existing DNC implementations [7]. MFF is fundamentally different in that it does not have a preceding, static network transformation step. It is tightly integrated into the feed-forward analysis (the backtracking) depicted in [23] in pseudo-code. This is the basis for our implementation that makes analysis-internal information accessible for the MFF analysis.

[1] See networkcalculus.org/dnc, formerly known as Disco Network Calculator [23] and as DiscoDNC [24].

B. Avionics-like Topology Generation

For our experiments we used an AFDX generator, which creates topologies similar to the one presented in [15]. The first AFDX experiment used a small topology, with 10 End-Systems (ESs) and 6 switches. Each switch can connect from 1 to 4 switches and to between 1 and 3 ESs, creating up to seven servers per switch (at the output port locations). Since AFDX is based on Ethernet technology, we modeled all service curves as rate latency curves, with rate $100e^6$ (i.e., 100 Mbps) and 0 latency. Each Virtual Link (VL, i.e. the multicast flows in AFDX terminology) has 4 sinks, with the Bandwidth Allocation Gap (BAG, a minimum inter-arrival time for a flow's data) being randomly selected and varying from 1 to 128 milliseconds (in powers of 2). We varied the maximum frame size from 500 to 1200 bytes.

Due to the limited size of the AFDX-like networks used in this experiment, as well as the limits used for the other variables, we opted to create different sizes (or load scenarios) by increasing the number of VLs per topology. In order to verify how the algorithms behaved to differently occupied networks, we chose 10, 40 and 80 VLs each. The VL paths were chosen randomly, as well as the source-sink pairs. This can cause some networks to be invalid when the long-term rate of incoming data at a server exceeds its capacity. Then, the worst-case delay cannot be bounded.

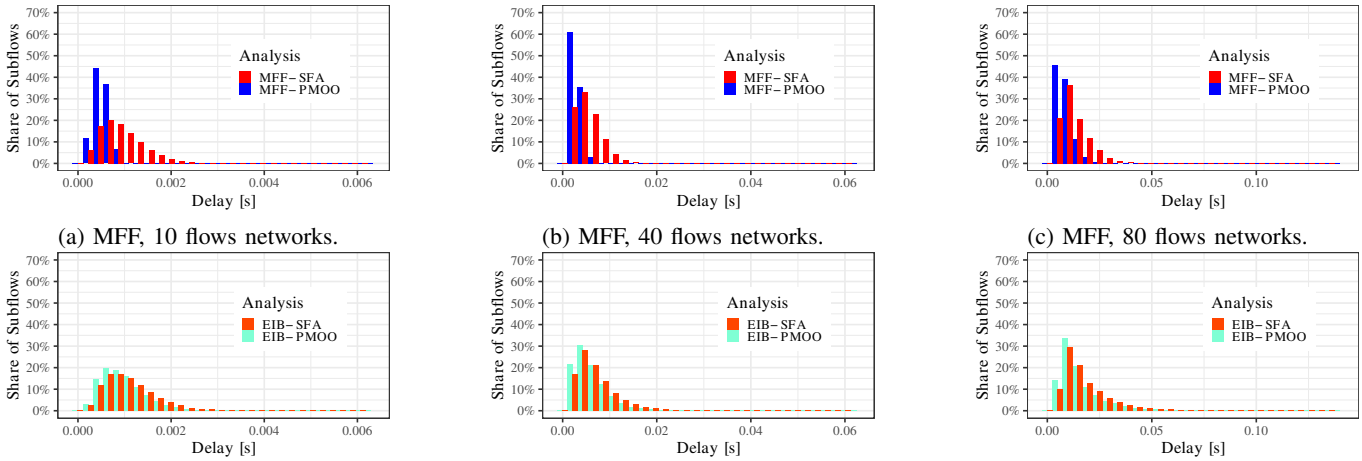
The second experiment for AFDX networks is supposed to resemble an Airbus A350-like topology. We created networks similar to the one described in [25], which is one of the few sources available for dimensioning more realistic networks than the one in the scaling experiments. However, due to the complexity of the analyses and resulting computation times we had to use a reduced number of VLs – this is the first experience we can share regarding the currently available NC tools and multicasting. For this experiment, we used 650 VLs per network, with BAG values ranging from 2 to 128 milliseconds (in powers of 2), and max frames values from 300 to 500 bytes. Each VL also had from 1 to 12 sinks.

C. Evaluation Equipment

The experiments described in this section were executed on a Supermicro X7DVL rack server, with dual-Intel Xeon E5420 CPUs and 12GB RAM. Note, however, that the DNC tool is single-threaded at the time of writing.

IV. Numerical Evaluation Results

In our numerical experiments, we aim at comparing the three different NC framework extensions. UT, the trivial



(a) MFF, 10 flows networks. (b) MFF, 40 flows networks. (c) MFF, 80 flows networks.

(d) EIB, 10 flows networks. (e) EIB, 40 flows networks. (f) EIB, 80 flows networks.

Figure 1: Delay bound distribution of MFF (a to c) and EIB (d to f) analyses in the small AFDX sample networks.

extension, suffers from duplication of flows and thus it is not a competitive contender. For this reason, we reduce the set of thoroughly investigated analyses to EIB and MFF. However, we also provide a benchmark of their respective improvement over UT. In the first set of AFDX numerical results, we intend to identify fundamental trends, not outliers. Therefore, we created 1000 valid networks for each size and will show aggregated results.

A. Avionics-like networks

EIB and MFF: PMOO vs. SFA

The experimental results clearly show that the MFF analysis is in general superior to the EIB analysis, and that the UT is inferior to both of them. For MFF and EIB, this is shown in Figure 1a to 1f. We focus on the spread of delay bounds, visualized by the interval in which each histogram has its bins. For MFF, the maximum bin value is of 0.004 seconds delay, whereas for EIB it is 0.006 seconds. As the number of flows increase, we see that the maximum bin value for MFF increases more slowly than for EIB. This means that the MFF analysis scales better than EIB on the number of flows and thus server utilization. At this point, we make no distinction between PMOO or SFA, because these trends apply to both versions of the analyses.

In Figure 1a to Figure 1c, we show the results for the MFF analysis for the small AFDX topology. Each histogram considers all subflows from the 1000 generated networks, and each subflow is analyzed under PMOO and SFA. The aim is to give a general idea of the average distribution of delays bounded by each analysis. Figure 1a shows the distribution for the networks with only 10 flows, Figure 1b shows the distribution for 40 flows, and Figure 1c shows the distribution for 80 flows. The first important point to observe in these histograms is the difference between the PMOO version (in blue) and the SFA version (in red). For all three network sizes, we have the PMOO concentrated in the left portion of the histograms. This shows that the PMOO instantiations of the analysis is, in general, yielding smaller (i.e., tighter) delay bounds than SFA, which spreads bins ranging from the lesser values of the histograms to the larger ones. This holds for all three sets of experiments shown in Figure 1a to 1c.

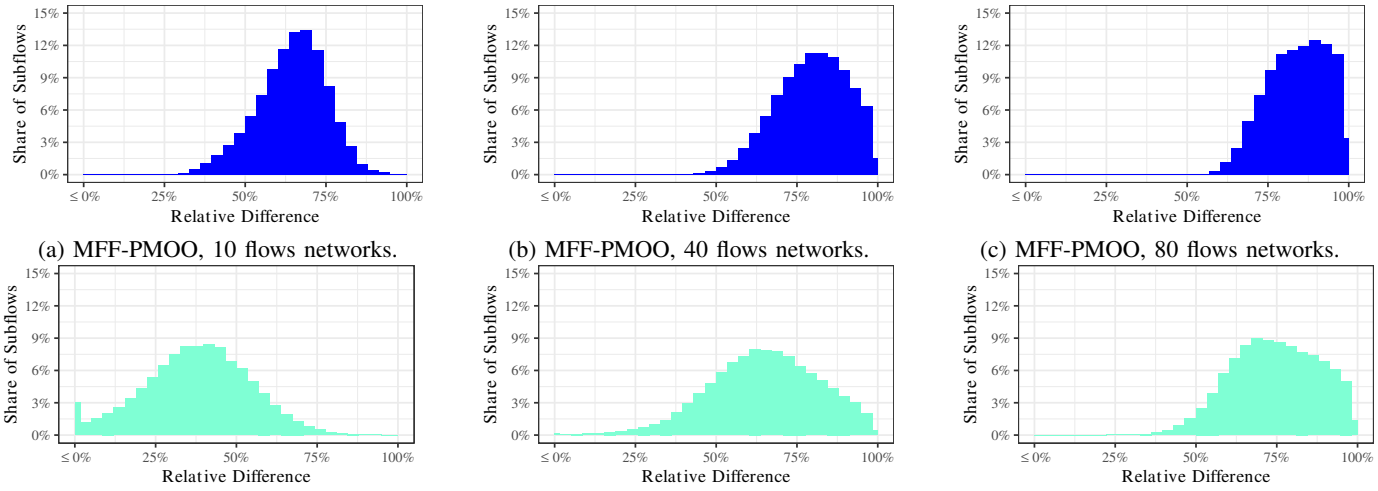
Figures 1d to 1f show the results for the EIB analysis for the same small AFDX topologies. As with the previous results, Figure 1d gives the delay distribution bounded by the analysis for all networks with 10 flows, Figure 1e the delays bounded for all networks with 40 flows, and Figure 1f the delays for the networks with 80 flows. Similar to the MFF results, we point the reader to the left portion of the histograms. The PMOO portion (in light blue) has much larger bins in the left portion of the histograms than SFA (in light red), meaning that the delays bounded by it are tighter.

However, by looking at the maximal and minimal values of the histogram bins, we can see a large difference between MFF and EIB. The maximum value present in the MFF histogram of Figure 1a is 0.004 seconds, whereas in EIB Figure 1d is 0.006 seconds. This means that the EIB analysis for the same networks as MFF has computed larger delays, caused by the analysis adding more pessimism (by the cuts, see Section II) when deriving its results.

When looking only at the PMOO values, a similar behaviour can be seen. For the MFF Figure 1a, the larger bins of PMOO are around 0.001 seconds. On the other hand, for the EIB Figure 1d, the larger bins of PMOO are over 0.003 seconds. This is also present in the histograms of the networks with 40 flows and 80 flows.

EIB-PMOO and MFF-PMOO: Improvement over UT

With the overly pessimistic behaviour of UT in mind, we use the UT results of each subflow and computed the relative difference from the results of MFF and EIB. Since for MFF and EIB the PMOO versions yield the tightest bounds, we compute this difference using the values bounded by their version of the analysis. This relative difference illustrates how much the new analyses improved with respect to the more



(a) MFF-PMOO, 10 flows networks.

(b) MFF-PMOO, 40 flows networks.

(c) MFF-PMOO, 80 flows networks.

(d) EIB-PMOO, 10 flows networks.

(e) EIB-PMOO, 40 flows networks.

(f) EIB-PMOO, 80 flows networks.

Figure 2: Relative Difference of MFF (a to c) and EIB (d to f) analyses to UT int the small AFDX sample networks.

traditional UT analysis. The relative difference of MFF to UT is shown in Figures 2a to 2c, whereas the relative difference of EIB to UT is shown in Figures 2d to 2f. Results where MFF and EIB outperform UT are represented by values in the rightmost portion of these relative difference histograms. However, it is possible for UT to equally perform or even outperform the EIB analysis. This happens when the benefits of the PMOO analysis outweigh the flow duplication costs of UT, since EIB suffers from its explicit intermediate bounds that limit the property to sub-paths of the analyzed flow. Values in the leftmost bin of the histograms illustrate this scenario.

In Figures 2a to 2c we show the relative difference of MFF with respect to UT. It is worth noting that for all network sizes the leftmost bin, which represents results that were equal or in which UT outperformed MFF, is empty – our implementation passed its sanity check, that UT cannot outperform MFF. Moreover, in Figure 2a the relative difference ranges around 65%. As the number of flows increases, the distribution of bins tends more and more to the right, up to the point in Figure 2c where they all range from over 50% to 100%. This value of 100% can be attributed to the extreme pessimism shown by some bounds given by UT. In fact, when looking at the raw data, the largest delay given by UT for those networks is 30 seconds. This is an extremely pessimistic bound. The MFF analysis, for the same flow, yield a delay bound of only 0.01 seconds. Because of such a large difference between the results, we see a large 100% bin for the networks of size 80 flows. This is, again, explained by the flow duplication of UT, since each flow adds more utilization to the network when compared to the other analysis, i.e., UT does not scale with the number of flows.

The relative difference histograms from EIB to UT are shown in Figure 2d to Figure 2f. When looking at these histograms, we see a similar behaviour to the MFF histograms. As the number of flows in the network increases, the distribution tends to be concentrated in the rightmost bins, showing that EIB also scales better than UT. However, in the histogram of Figure 2d, we see that the first bin is not empty, with 3% of all flows being in there. This means that, for the given network size, UT outperformed EIB in some cases. Since this is the network with the least flows, UT's pessimistic resource demand duplication does not affect it as severely. Moreover, UT can fully benefit from the PMOO property, whereas EIB is forced to work with less servers due to its inherent break of multicast flows into smaller unicast ones. However, for the network with 40 flows in Figure 2e this first bin is much less pronounced, and in Figure 2f it is barely visible, showing that indeed EIB scales better than UT.

Table I: SFA outperforming PMOO, small AFDX, subflow %.

Algorithm	*10 Flows*	*40 Flows*	*80 Flows*
UT	8.7%	28.1%	46.8%
EIB	15.1%	23.4%	25.9%
MFF	5.2%	2.3%	3.9%

In Table I we present the percentage of subflows (source/sink pair) for each topology and each analysis where the delay bound given by SFA is lower than by PMOO. These are cases in which the PMOO-degrading settings addressed in the previous discussion are most likely to be present. Looking at the table, we see that as the number of flows is increased, the number of subflows affected by this behaviour increases for UT and EIB. On the other hand, for MFF the percentage ranges from 5% to 2%, showing no scaling to the utilization of the network. Although UT and MFF have the same PMOO worst-case, UT also suffers from its pessimistic server utilization, which is very relevant in this small topology, as shown by all data collected and presented. As a result, we see that after a given number of flows, analyses such as UT can have a large number of subflows in which SFA outperforms PMOO. In such networks, it might be more useful to use a SFA version of these analyses. This also has implications to unicast analyses in general. If we ignore the semantic of the multicast flows

in a UT transformed network, it is in fact a unicast network. Therefore, as we increase the number of flows in this type of highly utilized network, it seems that SFA might in fact be more suitable than PMOO – the motivation for the exhaustive search in the TMA analysis [6].

B. A350-like networks

As UT is clearly inferior to the other analyses, we only executed EIB and MFF in this experiment. This allows for the study of networks with much higher server utilization, since they were often invalidated by the over-pessimism of UT and its unbounded delays.

The histograms of Figure 3 show that for this topology MFF again outperforms EIB. The difference, however, is much more pronounced, with EIB deriving considerably more pessimistic bounds. In the histogram of Figure 3b, the maximum value of the largest bins are over 1000 seconds, whereas the values in the MFF histogram of Figure 3a are much less pessimistic. This is a similar behaviour to the one seen in the UT analysis. We believe that this happens because these networks have a much higher server utilization than the previous experiments, making the cutting in EIB's network transformation affect the analysis.

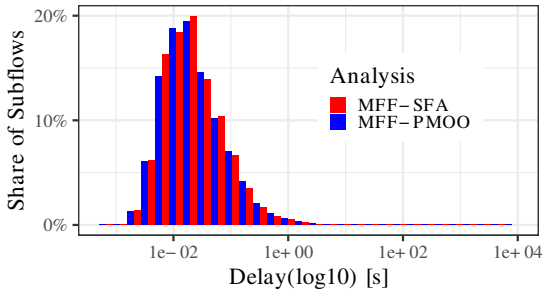
Table II: SFA outperforming PMOO, large AFDX network.

Algorithm	*Subflows [%]*	*Algorithm*	*Subflows [%]*
EIB	63.3%	MFF	78.6%

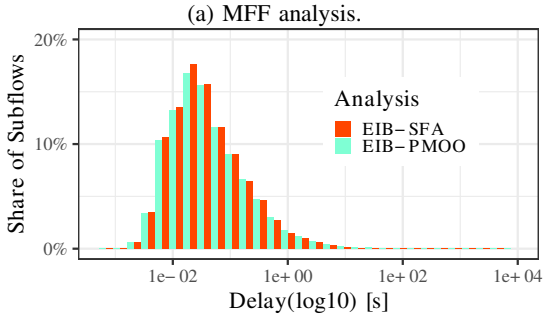
A difference from the previous experiment is that, in this case, the SFA versions of both analyses outperformed the PMOO versions in the majority of subflows, as shown in Table II, i.e., the end-to-end semantic of PMOO is less advantageous compared to having knowledge about servers in isolation (like their order). As mentioned before, this is related to the possible difference in server utilization in the network. The PMOO must assume the same worst-case residual service for all flows, whereas the SFA can account for such differences. This shows that for these specific cases, the SFA version might in fact be better suitable. Moreover, in this case the percentage of subflows suffering from this PMOO worst-case was higher for MFF. This can also be explained by the MFF analysis sharing the same PMOO-impacting scenarios.

C. EIB vs. MFF

Out of all the subflows tested in this experiment, only 4 of them had the delay bounds computed by EIB being smaller than MFF. On the other hand, we can report the tendency that EIB is the faster analysis to execute. We attribute this to the fundamentally different design of EIB as compared to MFF: it is more efficient to convert the network model before the analysis is run than adaptively select and ignore subflows during the analysis (see Section II). A thorough investigation of the analysis execution times is out of scope of this paper. Instead, and given this general trend, we want to focus on the question of how much tightness is lost in case a non-optimal analysis is chosen a priori. E.g., if a factor like analysis



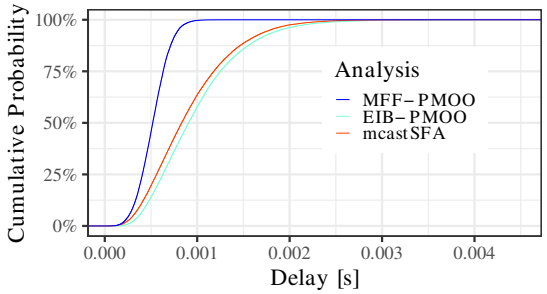
(a) MFF analysis.



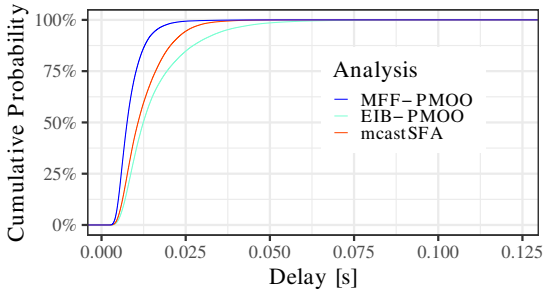
(b) EIB analysis.

Figure 3: Delay bound distribution in the large AFDX network.

run time is the driving force behind the decision and an EIB analysis is executed instead of an MFF analysis. As we have seen, the PMOO-instantiations are generally preferable. In order to reduce the complexity of the evaluation, we focus on EIB-PMOO and MFF-PMOO. The respective -SFA variants are combined into mcastSFA that we define as the minimum of EIB-SFA and MFF-SFA.



(a) 10 flows networks.



(b) 80 flows networks.

Figure 4: Delay bounds (ECDF) in the small AFDX networks.

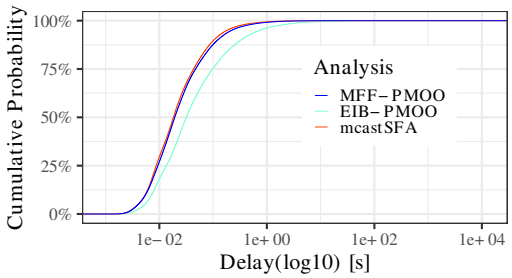


Figure 5: Delay bounds (ECDF) in the large AFDX network.

Figure 4 shows the delay bound ECDF for our set of small avionics-like networks. MFF-PMOO is clearly outperforming the other alternatives. Its curve rises to 100% faster, i.e., its delay bounds are in a smaller range. This is true for both sizes, but more pronounced in the smallest network. Moreover, note that the SFA-variants outperform EIB-PMOO, and that the gap increases with the network size. This trend continues as evident by increasing the size to the A350-like topology shown in Figure 5. It even leads to the fact that the mcastSFA curve is slightly left of the MFF-PMOO one – another confirmation of our previous assessment. This shows that the MFF-PMOO analysis is not strictly better than the others, yet, in particular in smaller networks it yields much tighter bounds in the vast majority of cases.

V. Conclusion

In this paper, we present a comprehensive numerical study of Network Calculus (NC) analyses for multicasting – a benchmarking that is necessary as neither of the major alternatives can be assumed invariantly superior to the others. Despite this being known, what remained without a thorough numerical investigation until now is how to best instantiate the NC framework to obtain an analysis that, in turn, derives the tightest possible bound on the worst-case end-to-end delay of a flow. This decision has to be taken a priori, and with the results in this paper we provide ample guidance on this issue.

We aim at providing a reproducible study in multiple dimensions: software implementation, experimental setup, numerical results. Besides a comparative depiction of an unprecedentedly large set of experiments (multiple avionics-like network design with 1000 individual network instantiations each), we are also able to give concise interpretation of the reasons behind our observations. Thus, we are able to provide a clear view on the actual importance of analyses' properties for the eventually derived delay bound. Additionally, we show fundamental trends when the network size and utilization increases.

In short, based on our comprehensive study, we can conclude that the multicasting analysis MFF (Multicast Feed-Forward) yields the tightest bounds. Moreover, we can recommend to instantiate it with NC's PMOO (Pay Multiplexing Only Once) analysis. However, if the network under investigation has high server utilizations and if computing resources are not restricted to running a single analysis only, then we recommend to execute a MFF-SFA (i.e. MFF with the Separated Flow Analysis) in addition.

References

[1] Aeronautical Radio, Incorporated (ARINC), "Avionics Full-Duplex Switched Ethernet (AFDX)," Specification 664 Part 7, Tech. Rep., 2009.

[2] S. Deering, "Host extensions for IP multicasting," Internet RFC 1112, Tech. Rep., Aug. 1989.

[3] H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Worst-case end-to-end delay analysis of an avionics AFDX network," in *Proc. of DATE*, 2010.

[4] 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.

[5] A. Scheffler, M. Fögen, and S. Bondorf, "The deterministic network calculus analysis: Reliability insights and performance improvements," in *Proc. of IEEE CAMAD*, 2018.

[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," *Proc. of the ACM on Measurement and Analysis of Computing Systems (POMACS)*, vol. 1, no. 1, pp. 16:1–16:34, 2017.

[7] S. Bondorf and F. Geyer, "Generalizing network calculus analysis to derive performance guarantees for multicast flows," in *Proc. of EAI ValueTools*, 2016.

[8] ——, *Deterministic Network Calculus Analysis of Multicast Flows*, ser. EAI/Springer Innovations in Communication and Computing. Springer International Publishing, 2019.

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

[10] 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.

[11] 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.

[12] H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Applying and optimizing trajectory approach for performance evaluation of AFDX avionics network," in *Proc. of IEEE ETFA*, 2009.

[13] G. Kemayo, F. Ridouard, H. Bauer, and P. Richard, "A forward end-to-end delays analysis for packet switched networks," in *Proc. of RTNS*, 2014.

[14] G. Kemayo, N. Benammar, F. Ridouard, H. Bauer, and P. Richard, "Improving AFDX end-to-end delays analysis," in *Proc. of IEEE ETFA*, 2015.

[15] M. Boyer, N. Navet, and M. Fumey, "Experimental assessment of timing verification techniques for AFDX," in *Proc. of ERTSS*, 2012.

[16] D. Petit, J.-P. Georges, T. Divoux, B. Regnier, and P. Miramont, "An admission control method to provide higher sampling rates over space launchers networks," in *Proc. of IFAC TA*, 2019.

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

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

[19] 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.

[20] S. Bondorf, P. Nikolaus, and J. B. Schmitt, "Catching corner cases in network calculus – flow segregation can improve accuracy," in *Proc. of GI/ITG MMB*, 2018.

[21] B. Cattelan and S. Bondorf, "Iterative design space exploration for networks requiring performance guarantees," in *Proc. of IEEE/AIAA DASC*, 2017.

[22] H. Yang, L. Cheng, and X. Ma, "Analyzing worst-case delay performance of IEC 61850-9-2 process bus networks using measurements and network calculus," in *Proc. of ACM e-Energy*, 2017.

[23] J. B. Schmitt and F. A. Zdarsky, "The DISCO network calculator – a toolbox for worst case analysis," in *Proc. of ICST ValueTools*, 2006.

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

[25] O. Hotescu, K. Jaffrès-Runser, J.-L. Scharbarg, and C. Fraboul, "Towards quality of service provision with avionics full duplex switching," in *Proc. of ECRTS Work-in-Progress Session*, 2017.