Network Synthesis under Delay Constraints: The Power of Network Calculus Differentiability

Fabien Geyer[1,2] and Steffen Bondorf[3]

INFOCOM 2022

Wednesday 4th May, 2022

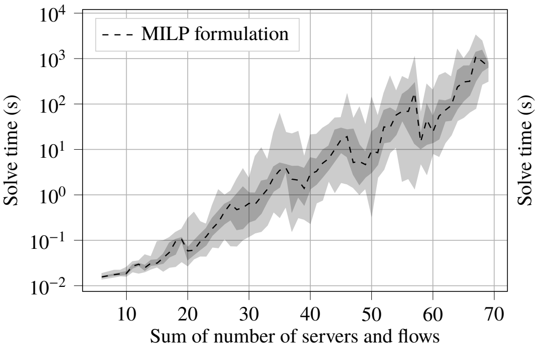
[2] Airbus Central R&T
Munich

[1] Chair of Network Architectures and Services
Technical University of Munich (TUM)

[3] Faculty of Computer Science
Ruhr University Bochum

Motivation

Network synthesis with strict delay constraints



Real-time network synthesis task

- Meet hard real-time end-to-end delay guarantees

$\rightarrow$ **Formal validation using Network Calculus**

- Optimize the network: paths, scheduler parameters, . . .

$\rightarrow$ **Combinatorial problem with exponential growth**

Application to Time-Sensitive Networking (TSN) and other real-time networks (eg. AFDX)

Main contributions

- Demonstrate how to differentiate network calculus delay bounds $\rightarrow$ **Differential Network Calculus**
- Illustrate how to optimize network paths using gradient-based optimization
- Enables scalability to large networks (1000+ flows)

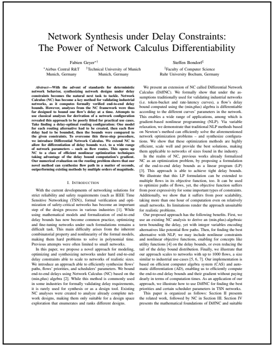
Outline of the talk

Introduction to network calculus

Differential Network Calculus

Numerical evaluation

Conclusions



Network Synthesis under Delay Constraints:
The Power of Network Calculus Differentiability

Fabien Geyer*†
*Airbus Central R&T, Munich, Germany
†Technical University of Munich, Munich, Germany

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

***Abstract*—With the advent of standards for deterministic network behavior, synthesizing network designs under delay constraints becomes the natural next task to tackle. Network Calculus (NC) has become a key method for validating industrial networks, as it computes formally verified end-to-end delay bounds. However, analyses from the NC framework were thus far designed to bound one flow's delay at a time. Attempts to use classical analyses for derivation of a network configuration revealed this approach to be poorly fitted for practical use cases. Take finding a delay-optimal routing configuration: One model for each routing alternative had to be created, then each flow delay had to be bounded, then the bounds were compared to the given constraints. To overcome this three-step procedure, we introduce Differential Network Calculus. We extend NC to allow for differentiation of delay bounds w.r.t. to a wide range of network parameters – such as flow routes. This opens up NC to a class of efficient nonlinear optimization techniques taking advantage of the delay bound computation's gradient. Our numerical evaluation on the routing problem shows that our novel method can synthesize flow path in a matter of seconds, outperforming existing methods by multiple orders of magnitude.**

I. Introduction

With the current developments of networking solutions for strict reliability and safety requirements (such as IEEE Time Sensitive Networking (TSN)), formal verification and optimization of safety-critical networks has become an important step of the design process in various industries [1]. While using mathematical models and formalization of end-to-end delay bounds has now become common practice, optimizing and fine-tuning networks under such formulations remains a difficult task. This main difficulty arises from the inherent combinatorial property and nonlinearity of the formal models, making them hard problems to solve in polynomial time. Previous attempts were often limited to small networks.

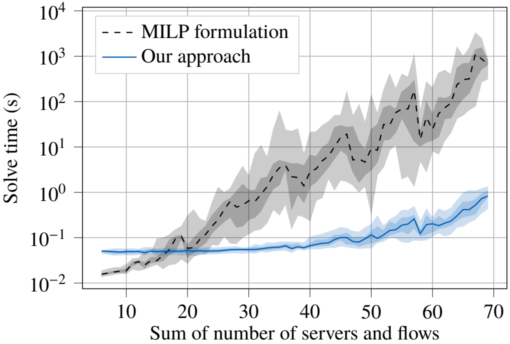
In this paper, we propose a novel approach for modeling, optimizing and synthesizing networks under hard end-to-end delay constraints able to scale to networks of realistic sizes. We introduce an approach able to efficiently synthesize flows' paths, flows' priorities, and schedulers' parameters. We bound end-to-end delays using Network Calculus (NC) based on the (min,plus) algebra [2]. While this method is commonly used in some industries for formally validating delay requirements, it is rarely used for synthesis or as a design tool. Existing NC analyses were created to analyze already complete network designs, making them only suitable for a design space exploration that enumerates and ranks different designs.

We present an extension of NC called Differential Network Calculus (DiffNC). We formally show that under the assumptions traditionally used for validating industrial networks (i.e. token-bucket and rate-latency curves), a flow's delay bound computed using the (min,plus) algebra is differentiable according to the different curves' parameters in the network. This enables a wide range of applications, among which is gradient-based nonlinear programming (NLP). Via variable relaxation, we demonstrate that traditional NLP methods based on Newton's method can efficiently solve the aforementioned network optimization problems – and synthesize configurations. We show that these optimization methods are highly efficient, scale well and provide the best solutions, making them applicable to networks of sizes found in the industry.

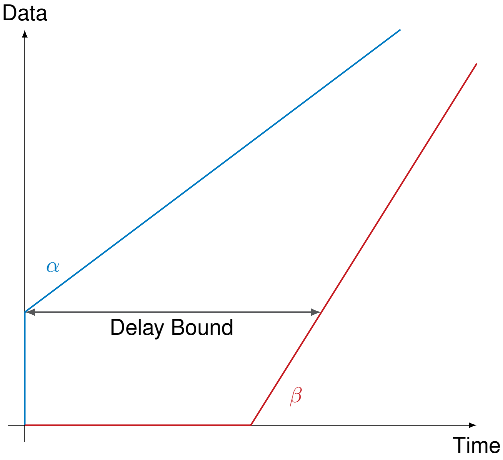
In the realm of NC, previous works already formalized NC as an optimization problem, by proposing a formulation of the end-to-end delay bounds as a linear program (LP) [3]. This approach is able to achieve tight delay bounds. We illustrate that this LP formulation can be extended to multiple flows in its objective function, too. It can be used to optimize paths of flows, yet, the objective function suffers from poor expressivity for some important types of constraints. Additionally, we show that it suffers from poor scalability, taking more than one hour of computation even on relatively small networks. Its limitations render the approach unsuitable for realistic problems.

Our proposed approach has the following benefits. First, we use an existing NC analysis to derive an (min,plus)-algebraic term bounding the delay, yet with integer variables encoding alternatives like potential flow paths. Then, for finding the best alternative with NLP, we may include nonlinear constraints and nonlinear objective functions, enabling for concepts like utility functions [4] on the delay bounds, or even reducing the tail of the delay bound distribution. Finally, we illustrate that our approach scales to networks with up to 1000 flows, a size similar to industrial use-cases [5, 6, 7]. Our implementation is based on efficient computer algebra system (CAS) and automatic differentiation (AD), enabling us to efficiently compute the end-to-end delay bounds and their gradient without paying dearly in terms of computation times. As an application of our approach, we illustrate how to use DiffNC for finding the best priorities and certain scheduler parameters in TSN networks.

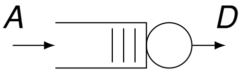
This paper is organized as follows: Section II presents the related work, followed by NC in Section III. Section IV presents the mathematical foundations of DiffNC and suitable



Network Calculus – Basics

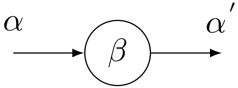


Basis: Cumulative arrivals and services [Cruz, 1991]



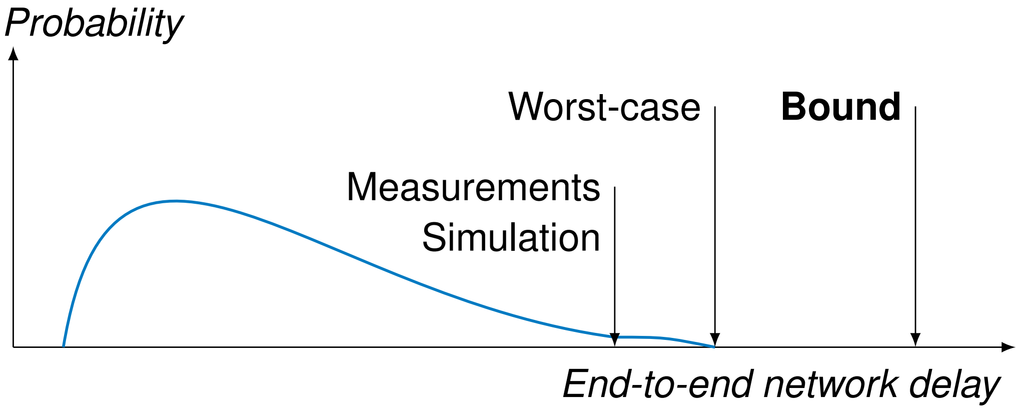
Arrival curve α: $A(t) - A(t-s) \leq \alpha(s), \forall t \leq s$

Service curve β: a server is said to offer a strict service curve β if, during any backlogged period of duration u, the output of the system is at least equal to $\beta(u)$



Introduction to network calculus

Bounds vs. worst-case



Introduction to network calculus

Algebraic Network Calculus Analysis

(min,plus) Algebra [Le Boudec and Thiran, 2001]

Based on the previous definitions, the (min,plus) algebra can be defined as:

$$\begin{aligned}
\text{aggregation:}\ (f+g)(d) &= f(d) + g(d) \\
\text{convolution:}\ (f \otimes g)(d) &= \inf_{0 \le u \le d} \{f(d-u) + g(u)\} \\
\text{deconvolution:}\ (f \oslash g)(d) &= \sup_{u \ge 0} \{f(d+u) - g(u)\} \\
\text{left-over:}\ (f \ominus g)(d) &= \sup_{0 \le u \le d} \{f(u) - g(u)\}
\end{aligned}$$

The combination of these operations is used for computing end-to-end delay bounds

$\rightarrow$ *Separate Flow Analysis* (SFA) and *Pay Multiplexing Only Once Analysis* (PMOO)

Differential Network Calculus

From algebraic NC analysis to differentiable delay bound

Closed-form expression of NC operations

With the assumption of using rate-latency service curves and token-bucket arrival curves, the NC (min,plus) operations have the following closed-form solutions:

$$
\begin{aligned}
\text{aggregation:} \quad & \gamma_{r_1,B_1} + \gamma_{r_2,B_2} = \gamma_{r_1+r_2,B_1+B_2} \\
\text{convolution:} \quad & \beta_{R_1,L_1} \otimes \beta_{R_2,L_2} = \beta_{\min(R_1,R_2),L_1+L_2} \\
\text{deconvolution:} \quad & \gamma_{r,B} \oslash \beta_{R,L} = \gamma_{r,B+r\cdot L} \\
\text{left-over:} \quad & \beta_{R,L} \ominus \gamma_{r,B} = \beta_{R-r,(B+R\cdot L)/(R-r)} \\
\text{delay bound:} \quad & h(\gamma_{r,B}, \beta_{R,L}) = B/R + L
\end{aligned}
$$

Theorem: Differentiability of delay expression

With the assumption of using rate-latency service curves and token-bucket arrival curves, **a NC end-to-end delay bound is differentiable w.r.t. the curves parameters.**

Back to main goal: optimize flows' path

Virtual flow concept

A set of paths $\mathcal{P}_{f_i}$ are considered for each flow $f_i \in \mathcal{F}$.
For each flow f_i and each potential path $j \in \mathcal{P}_{f_i}$, we define $p_{f_{i,j}}$ as a binary variable representing the choice of path j for flow f_i.

$$\sum_{j \in \mathcal{P}_{f_i}} p_{f_{i,j}} = 1, \forall f_i \in \mathcal{F}$$

For each virtual flow, the arrival curve is reformulated as:

$$\forall\, 0 \leq d \leq t \;:\; A_{f_{i,j}}(t) - A_{f_{i,j}}(t-d) \leq \alpha_{f_i}(d) \cdot p_{f_{i,j}}$$

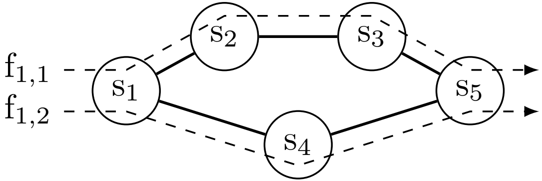


Illustration of virtual flow concept with one flow taking two potential paths in the server graph.

Lemma

Using the previous theorem, we can **differentiate according to the flow's paths**.

Differential Network Calculus

Constrained nonlinear programming

Using the previous results, the following Nonlinear Program is defined:

$$\min_{p_{f_{i,j}},\forall f_i \in \mathcal{F}, j \in \mathcal{P}_{f_i}} \frac{1}{|\mathcal{F}|} \sum_{i,j} \mathit{delay\ bound}(f_{i,j}) \cdot p_{f_{i,j}}$$

$$\text{s.t.} \quad 0 \leq p_{f_{i,j}} \leq 1, \forall f_i \in \mathcal{F}, j \in \mathcal{P}_{f_i}$$

$$\sum_{j \in \mathcal{P}_{f_i}} p_{f_{i,j}} = 1, \forall f_i \in \mathcal{F}$$

$$\sum_{i \in T(k)} r_i \cdot p_{f_{i,j}} \leq R_k, \forall k \in \mathcal{S}$$

$$\sum_{j \in \mathcal{P}_{f_i}} \mathit{delay\ bound}(f_{i,j}) \cdot p_{f_{i,j}} \leq \mathit{Req.}$$

Can be optimized using gradient-based optimization

Other objective functions

Using nonlinear utility functions U_i for the delay bounds

$$\min_{p_{f_{i,j}},\forall i,j} \sum_i U_i \left(\sum_j \mathit{delay\ bound}(f_{i,j}) \cdot p_{f_{i,j}} \right)$$

Tail of the delay bound distribution

$$\min_{p_{f_{i,j}},\forall i,j} \max_i \left(\sum_j \mathit{delay\ bound}(f_{i,j}) \cdot p_{f_{i,j}} \right)$$

Differential Network Calculus

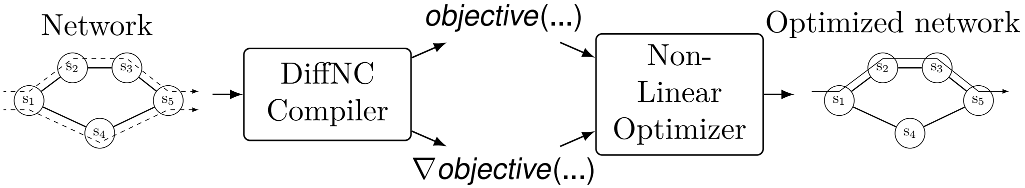
Putting it all together in practice

Practical and efficient way of computing delay bounds and their derivatives w.r.t. the paths

- Closed-form expressions of the gradient (eg. with SymPy [Meurer et al., 2017]) → Poor scalability
- Automatic Differentiation with computer algebra system using CasADi [Andersson et al., 2019] → **Good scalability**

Efficient gradient-based optimizer

- Sequential least squares quadratic programming (SLSQP) [Kraft, 1988, Johnson, 2020] showed great performance



Numerical evaluation

Evaluated networks and other methods

Table 1: Statistics about the small networks

Number of	Min	Mean	Max
Servers	3	8.68	18
Flows	3	9.70	21
Virtual flows	4	18.62	45
Path combinations	$10^{0.30}$	$10^{2.07}$	$10^{5.52}$

Used for comparison against:

- Bruteforce approach
- Mixed-Integer Linear Programming

Table 2: Statistics about the large networks

Number of	Min	Mean	Max
Servers	8	17.08	31
Flows	5	170.67	1001
Virtual flows	9	355.22	1884
Path combinations	$10^{1.08}$	$10^{46.04}$	$10^{229.08}$

Used for comparison against:

- Randomized search
- Shortest path (i.e. similar to Dijkstra's algorithm)
- Meta-heuristic algorithms (eg. evolution-based)
- Global optimization

Dataset with networks available at: `https://github.com/fabgeyer/dataset-infocom2022`

Numerical evaluation

Optimality against a bruteforce approach

How close are we to the optimal solution?

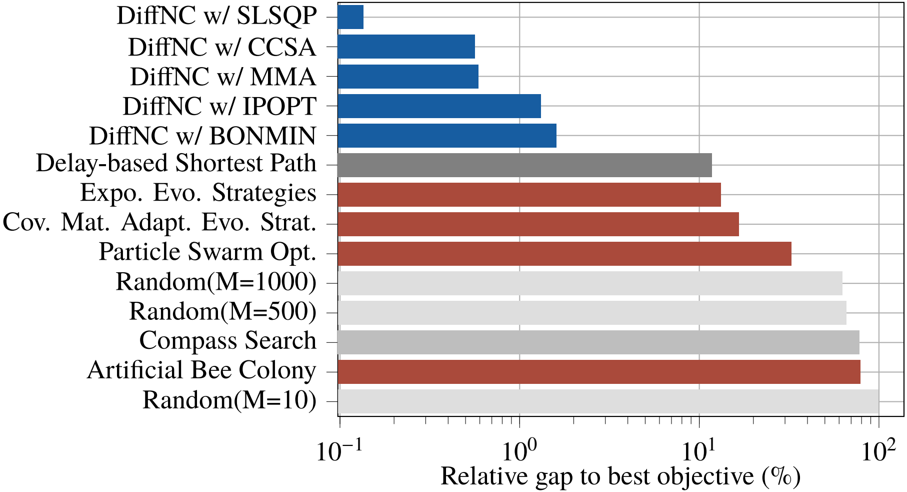
Metric used:

$$RelGap_{\text{method}} = \frac{objective_{\text{method}}}{objective_{\text{Bruteforce}}} - 1$$

Method	Optimum found	Rel. gap to bruteforce	Avg. exec. time
Bruteforce	-	-	123,05 s
DiffNC w/o restarts	85,30 %	0,17 %	0,05 s
DiffNC w/ restarts	99,53 %	$7{,}1 \times 10^{-4}$ %	0,17 s

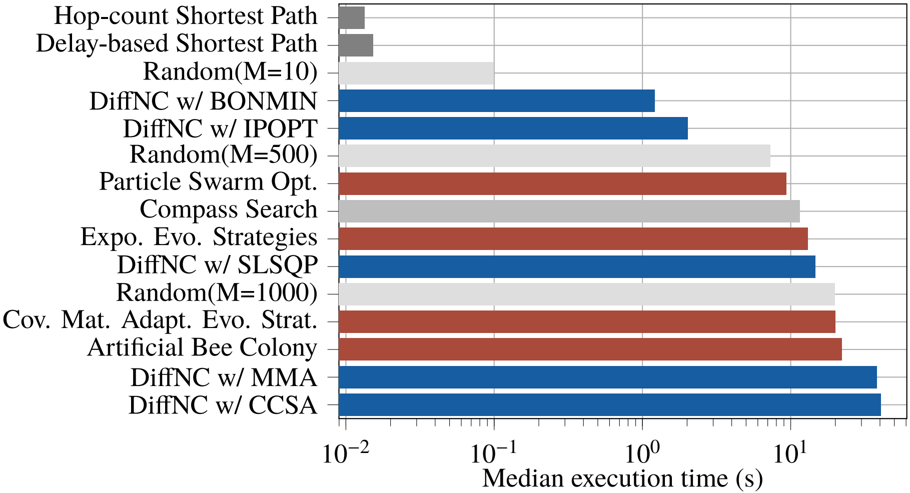
Numerical evaluation

Average relative gap to the best objective



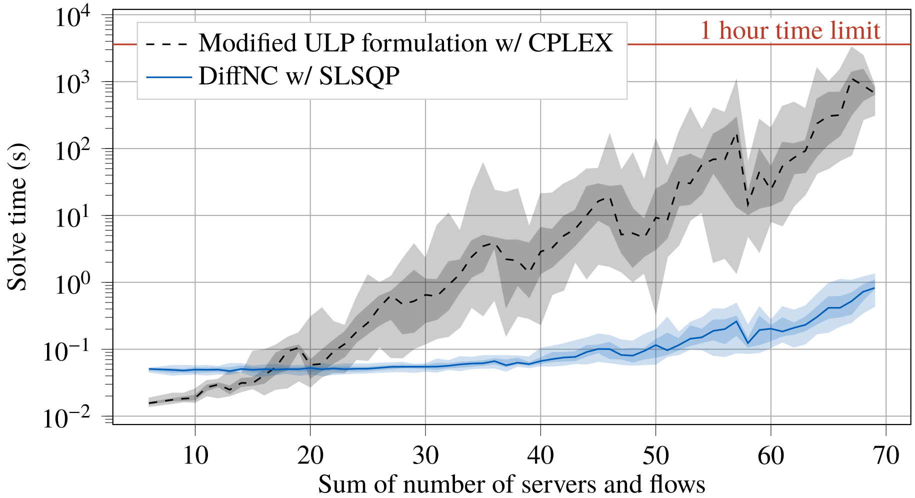
Numerical evaluation

Execution time



Numerical evaluation

DiffNC vs. MILP formulation (based on [Bouillard et al., 2010])

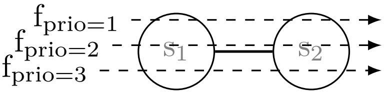


Conclusions

Applications and extensions

Priority assignment

One virtual flow for each potential priority class. DiffNC used for **differentiating w.r.t. the priority**



Packet scheduling parameters

DiffNC used for **differentiating w.r.t. scheduler weights** and other parameters of schedulers

Time-Sensitive Networking

Some flows are TDMA scheduled:

$$\beta_{\mathrm{TDMA}}(t) = R \cdot \max\left(\left\lfloor \frac{t}{c} \right\rfloor s, t - \left\lceil \frac{t}{c} \right\rceil (c - s)\right)$$

It is also possible to **differentiate w.r.t. the cycle parameters** *c* **and** *s*.

Deep learning

Recent interest for applying ML to Network Calculus [Geyer and Bondorf, 2019, Geyer et al., 2021, Mai and Navet, 2021]

DiffNC can also be used for **true end-to-end back propagation**

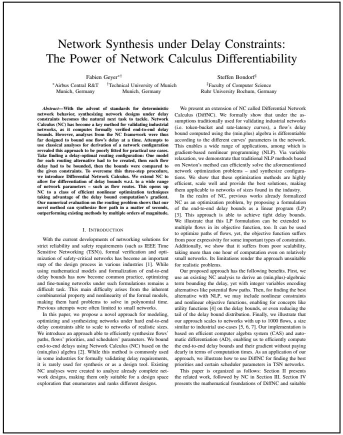
Conclusions

Contributions

- **Demonstrated how to differentiate network calculus delay bounds**
- Application to optimization of flows' path with nonlinear programming
- Methods and techniques for using it in practice and making it scale
- Dataset: `https://github.com/fabgeyer/dataset-infocom2022`

Future work

- Application to Time-Sensitive Networking
- Application to Deep Learning



Network Synthesis under Delay Constraints:
The Power of Network Calculus Differentiability

Fabien Geyer[*†]
[*]Airbus Central R&T, Munich, Germany
[†]Technical University of Munich, Munich, Germany

Steffen Bondorf[‡]
[‡]Faculty of Computer Science, Ruhr University Bochum, Germany

Abstract—**With the advent of standards for deterministic network behavior, synthesizing network designs under delay constraints becomes the natural next task to tackle. Network Calculus (NC) has become a key method for validating industrial networks, as it computes formally verified end-to-end delay bounds. However, analyses from the NC framework were thus far designed to bound one flow's delay at a time. Attempts to use classical analyses for derivation of a network configuration revealed this approach to be poorly fitted for practical use cases. Take finding a delay-optimal routing configuration: One model for each routing alternative had to be created, then each flow delay had to be bounded, then the bounds were compared to the given constraints. To overcome this three-step procedure, we introduce Differential Network Calculus. We extend NC to allow for differentiation of delay bounds w.r.t. to a wide range of network parameters – such as flow routes. This opens up NC to a class of efficient nonlinear optimization techniques taking advantage of the delay bound computation's gradient. Our numerical evaluation on the routing problem shows that our novel method can synthesize flow path in a matter of seconds, outperforming existing methods by multiple orders of magnitude.**

I. Introduction

With the current developments of networking solutions for strict reliability and safety requirements (such as IEEE Time Sensitive Networking (TSN)), formal verification and optimization of safety-critical networks has become an important step of the design process in various industries [1]. While using mathematical models and formalization of end-to-end delay bounds has now become common practice, optimizing and fine-tuning networks under such formulations remains a difficult task. This main difficulty arises from the inherent combinatorial property and nonlinearity of the formal models, making them hard problems to solve in polynomial time. Previous attempts were often limited to small networks.

In this paper, we propose a novel approach for modeling, optimizing and synthesizing networks under hard end-to-end delay constraints able to scale to networks of realistic sizes. We introduce an approach able to efficiently synthesize flows' paths, flows' priorities, and schedulers' parameters. We bound end-to-end delays using Network Calculus (NC) based on the (min,plus) algebra [2]. While this method is commonly used in some industries for formally validating delay requirements, it is rarely used for synthesis or as a design tool. Existing NC analyses were created to analyze already complete network designs, making them only suitable for a design space exploration that enumerates and ranks different designs.

We present an extension of NC called Differential Network Calculus (DiffNC). We formally show that under the assumptions traditionally used for validating industrial networks (i.e. token-bucket and rate-latency curves), a flow's delay bound computed using the (min,plus) algebra is differentiable according to the different curves' parameters in the network. This enables a wide range of applications, among which is gradient-based nonlinear programming (NLP). Via variable relaxation, we demonstrate that traditional NLP methods based on Newton's method can efficiently solve the aforementioned network optimization problems – and synthesize configurations. We show that these optimization methods are highly efficient, scale well and provide the best solutions, making them applicable to networks of sizes found in the industry.

In the realm of NC, previous works already formalized NC as an optimization problem, by proposing a formulation of the end-to-end delay bounds as a linear program (LP) [3]. This approach is able to achieve tight delay bounds. We illustrate that this LP formulation can be extended to multiple flows in its objective function, too. It can be used to optimize paths of flows, yet, the objective function suffers from poor expressivity for some important types of constraints. Additionally, we show that it suffers from poor scalability, taking more than one hour of computation even on relatively small networks. Its limitations render the approach unsuitable for realistic problems.

Our proposed approach has the following benefits. First, we use an existing NC analysis to derive an (min,plus)-algebraic term bounding the delay, yet with integer variables encoding alternatives like potential flow paths. Then, for finding the best alternative with NLP, we may include nonlinear constraints and nonlinear objective functions, enabling for concepts like utility functions [4] on the delay bounds, or even reducing the tail of the delay bound distribution. Finally, we illustrate that our approach scales to networks with up to 1000 flows, a size similar to industrial use-cases [5, 6, 7]. Our implementation is based on efficient computer algebra system (CAS) and automatic differentiation (AD), enabling us to efficiently compute the end-to-end delay bounds and their gradient without paying dearly in terms of computation times. As an application of our approach, we illustrate how to use DiffNC for finding the best priorities and certain scheduler parameters in TSN networks.

This paper is organized as follows: Section II presents the related work, followed by NC in Section III. Section IV presents the mathematical foundations of DiffNC and suitable

Bibliography

[Andersson et al., 2019] Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., and Diehl, M. (2019).
CasADi – A software framework for nonlinear optimization and optimal control.
Mathematical Programming Computation.

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

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

[Cruz, 1991] Cruz, R. L. (1991).
A calculus for network delay, part I: Network elements in isolation.
IEEE Trans. Inf. Theory, 37(1):114–131.

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

[Geyer et al., 2021] Geyer, F., Scheffler, A., and Bondorf, S. (2021).
Tightening Network Calculus Delay Bounds by Predicting Flow Prolongations in the FIFO Analysis.
In *Proc. of IEEE RTAS*.

[Johnson, 2020] Johnson, S. G. (2020).
The NLopt nonlinear-optimization package – version 2.7.0.

[Kraft, 1988] Kraft, D. (1988).
A software package for sequential quadratic programming.
Technical Report DFVLR-FB 88-28, DFVLR, Institut für Dynamik der Flugsysteme, Germany.

[Mai and Navet, 2021] Mai, T. L. and Navet, N. (2021).
Improvements to deep-learning-based feasibility prediction of switched Ethernet network configurations.
In *Proc. of RTNS*.

[Meurer et al., 2017] Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M. J., Terrel, A. R., Roučka, v., Saboo, A., Fernando, I., Kulal, S., Cimrman, R., and Scopatz, A. (2017).
SymPy: symbolic computing in Python.
PeerJ Computer Science.