On the Robustness of Deep Learning-predicted Contention Models for Network Calculus

Fabien Geyer\textsuperscript{1,2} and Steffen Bondorf\textsuperscript{3}

IEEE ISCC 2020
Tuesday 7\textsuperscript{th} July, 2020

\textsuperscript{1}Airbus Central R&T Munich, Germany
\textsuperscript{2}Chair of Network Architectures and Services Technical University of Munich, Germany
\textsuperscript{3}Faculty of Mathematics, Center of Computer Science Ruhr University Bochum, Germany
Motivation

Worst-Case End-to-End Performance Analysis

- Trade-off between computational effort and tightness
- **This talk:** network analysis method with good tightness and fast execution
Motivation

Network Calculus – Basics

**Basis:** Cumulative arrivals and services [Cruz, 1991a]

**Arrival curve** \( \alpha \): \( A(t) - A(t - s) \leq \alpha(s), \forall t \geq s \)

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

Network Calculus – Network Analysis

How to compute end-to-end performance?

TFA – Total Flow Analysis [Cruz, 1991b]

Step 1: Compute delay at each server on the path

Step 2: Sum delays

Server concatenation [Le Boudec and Thiran, 2001]

(min, +) algebra gives us:

→ Pay Bursts Only Once principle
Motivation
Network Calculus – Network Analysis

**SFA – Separate Flow Analysis**
[Le Boudec and Thiran, 2001]

*Step 1:* Compute per-server residual service

*Step 2:* Concatenate the servers

*Step 3:* Compute delay over concatenated server

---

**PMOO – Pay Multiplexing Only Once**
[Schmitt et al., 2008b]

*Step 1:* Concatenate the servers

*Step 2:* Compute residual service

*Step 3:* Compute delay over concatenated server
Motivation
Network Calculus – TMA

**TMA – Tandem Matching Analysis** [Bondorf et al., 2017]

- Main concept: apply concatenation only for some servers
- Exhaustive search to find which concatenations will result in the tightest end-to-end delay $\to O(2^{n-1})$

$$\begin{align*}
\alpha & \xrightarrow{\text{Cut}} \beta_1 \\
& \xrightarrow{\text{Cut}} \beta_2 \\
& \xrightarrow{} \beta_3 \\
& \xrightarrow{} \alpha'
\end{align*}$$

$$\begin{align*}
\alpha & \xrightarrow{\text{Cut}} \beta_1 \\
& \xrightarrow{} \beta_2 \\
& \xrightarrow{} \beta_3 \\
& \xrightarrow{} \alpha'
\end{align*}$$

Alternative 1

$$\begin{align*}
\alpha & \xrightarrow{} \beta_1 \otimes \beta_2 \\
& \xrightarrow{} \beta_3 \\
& \xrightarrow{} \alpha'
\end{align*}$$

Alternative 2

$$\begin{align*}
\alpha & \xrightarrow{\text{Cut}} \beta_1 \\
& \xrightarrow{} \beta_2 \otimes \beta_3 \\
& \xrightarrow{} \alpha'
\end{align*}$$
**Motivation**

Network Calculus – DeepTMA

**Approach:** Avoid TMA’s exhaustive search using ML

[Schmitt et al., 2008a][Bouillard et al., 2010]

→ **DeepTMA:**
- Main idea: use neural networks for predicting best cuts
- Even if the heuristic is wrong, the bounds are still valid

**Figure 1:** Approach
Motivation

Network Calculus – Contributions

[Geyer and Bondorf, 2019] introduced DeepTMA, but did not explore its scalability or robustness.

**New results:** Explore the robustness of DeepTMA

- Influence of network size (number of flows and servers) and topology type on accuracy and tightness?
- Scalability on larger networks (up to 10,000s of flows)?
- Importance of features used by the machine learning algorithm?
Outline

DeepTMA: Heuristic based on Graph Neural Networks

Numerical evaluation

Conclusion
DeepTMA: Heuristic based on Graph Neural Networks

Introduction

**Principle:** Replace exhaustive search by a fast heuristic [Geyer and Bondorf, 2019]

**Heuristic**
- Use Graph Neural Network
- Classification problem for cuts

**Graph formulation**
- Nodes: flows, servers, cuts
- Edges: relationships between elements
- Prediction if cut is applied or not
DeepTMA: Heuristic based on Graph Neural Networks

Problem formulation as graph
DeepTMA: Heuristic based on Graph Neural Networks

Graph Neural Networks – Introduction

Graph Neural Networks [Scarselli et al., 2009] and related architectures are able to process general graphs and predict feature of nodes $o_v$

**Principle**
- Each node has a *hidden* vector $h_v \in \mathbb{R}^k$
- ... computed according to the vector of its neighbors
- ... and are propagated through the graph

**Algorithm**
- Initialize $h_v^{(0)}$ according to features of nodes
- for $t = 1, \ldots, T$ do
  - $a_v^{(t)} = AGGREGATE \left( \left\{ h_u^{(t-1)} \mid u \in Nbr(v) \right\} \right)$
  - $h_v^{(t)} = COMBINE \left( h_v^{(t-1)}, a_v^{(t)} \right)$
- return $READOUT \left( h_v^{(T)} \right)$
DeepTMA: Heuristic based on Graph Neural Networks

Graph Neural Networks – Implementation

Implementation (simplified)

- Initialize $h_v^{(0)}$ according to features of nodes
- for $t = 1, \ldots, T$ do
  - $AGGREGATE \rightarrow a_v^{(t)} = \sum_{u \in Nbr(v)} h_u^{(t-1)}$
  - $COMBINE \rightarrow h_v^{(t)} = Neural \ Network \left( h_v^{(t-1)}, a_v^{(t)} \right)$
- $READOUT \rightarrow$ return $Neural \ Network \left( h_v^{(T)} \right)$

Training

- Using standard gradient descent techniques

Different approaches

- Gated-Graph Neural Network
- Graph Convolution Network
- Graph Attention Networks
- Graph Spatial-Temporal Networks
- …

→ Hot area of research in the ML community
Numerical evaluation

Previous results from [Geyer and Bondorf, 2019]

- We already showed that DeepTMA is a fast and accurate method
- Relative error: metric used for estimating tightness:

\[
\text{RelErr}_{f_i} = \frac{\text{Delay}^\text{DeepTMA}_{f_i} - \text{Delay}^\text{TMA}_{f_i}}{\text{Delay}^\text{TMA}_{f_i}}
\]

(1)
Numerical evaluation

Dataset generation for training

- Generation of 172,374 networks with tandem, tree or random graph topology
- Random generation of curve parameters for servers and flows
- Evaluation of each network using DiscoDNC and extract intermediary results of TMA
- Dataset available online: https://github.com/fabgeyer/dataset-deeptma-extension

| Parameter                              | Min | Max   | Mean  | Median |
|----------------------------------------|-----|-------|-------|--------|
| # of servers                           | 2   | 41    | 14.6  | 12     |
| # of flows                             | 3   | 203   | 101.2 | 100    |
| # of tandem combinations               | 2   | 197,196 | 1508.5| 384    |
| # of nodes in analyzed graph           | 10  | 2093  | 545.2 | 504    |
| # of tandem combination per flow       | 2   | 65,536| 19.4  | 4      |
| # of flows per server                  | 1   | 173   | 18.1  | 10     |

Table 1: Statistics about the generated dataset.
Numerical evaluation

Tightness vs. network size used for training

![Graph showing tightness vs. network size](image)

- Full dataset
- Networks up to 100 flows
- Networks up to 50 flows

Relative error to TMA (%) vs. Path length of flow
Numerical evaluation

Evaluation dataset

- Evaluated also on dataset from [Bondorf et al., 2017] with larger networks
- Up to 2 orders of magnitude larger in terms of number of servers and flows per network
- Neural network not trained on such large networks

| Parameter                                | Min  | Max    | Mean   | Median |
|------------------------------------------|------|--------|--------|--------|
| # of servers                             | 38   | 3626   | 863.0  | 693    |
| # of flows                               | 152  | 14 504 | 3452.0 | 2772   |
| # of tandem combinations                 | 2418 | 121 860| 24 777.6| 18 869 |
| # of nodes in analyzed graph             | 1358 | 113 162| 25 137.7| 19 518 |
| # of tandem combination per flow         | 2    | 512    | 7.3    | 8      |
| # of flows per server                    | 1    | 467    | 16.4   | 12     |

Table 2: Statistics about the set of networks from [Bondorf et al., 2017].
Numerical evaluation

Tightness in larger dataset

![Diagram showing numerical evaluation results for different network sizes and relative errors to TMA. The x-axis represents the number of servers in the network, while the y-axis shows the relative error to TMA in percentages. The lines represent different models: RND, RND_2, RND_4, RND_8, DeepTMA, DeepTMA_2, DeepTMA_4, and DeepTMA_8.]
Numerical evaluation

Feature importance

- Tandem networks
- Tree networks
- Random networks

ServiceRate
ArrivalRate
PathOrder
ArrivalBurst
ServiceLatency

Feature importance (%)
Conclusion

Contributions

- **Framework combining network calculus and graph-based deep learning**
- **Results show scalability on networks larger by 2 orders of magnitude**
- Feature importance will guide next iterations of the method
- Dataset available online for reproducing our results:
  → https://github.com/fabgeyer/dataset-deeptma-extension

Future work

- Applicability at other problems in Network Calculus
- Extension to other formal methods for network verification
[Bondorf et al., 2017] Bondorf, S., Nikolaus, P., and Schmitt, J. B. (2017).
Quality and cost of deterministic network calculus – design and evaluation of an accurate and fast analysis.
Proc. ACM Meas. Anal. Comput. Syst. (POMACS), 1(1):16:1–16:34.

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

[Cruz, 1991b] Cruz, R. L. (1991b).
A calculus for network delay, part II: Network analysis.

IEEE Trans. Inf. Theory, 37(1):132–141.

[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 INFOCOM.

[Scarselli et al., 2009] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009).
The graph neural network model.
IEEE Trans. Neural Netw., 20(1):61–80.

[Schmitt et al., 2008a] Schmitt, J. B., Zdarsky, F. A., and Fidler, M. (2008a).
Delay bounds under arbitrary multiplexing: When network calculus leaves you in the lurch.
In Proc. of IEEE INFOCOM.

[Schmitt et al., 2008b] Schmitt, J. B., Zdarsky, F. A., and Martinovic, I. (2008b).
Improving performance bounds in feed-forward networks by paying multiplexing only once.
In Proc. of GI/ITG MMB.