Predicting Latency Quantiles using Network Calculus-assisted GNNs

Max Helm
Technical University of Munich
Munich, Germany
helm@net.in.tum.de

Georg Carle
Technical University of Munich
Munich, Germany
carle@net.in.tum.de

ABSTRACT

Network digital twins commonly rely on Graph Neural Networks (GNNs) as functional models. They typically predict network performance metrics, such as latencies. Most approaches have one of the following restrictions: they use simulated data, predict mean values, or don't utilize formal method results as inputs. We introduce an approach that: (I) relies on data obtained from a hardware testbed, increasing realism, (II) predicts quantiles in addition to means, increasing flexibility and applicability, (III) uses the formal method of network calculus to obtain input features, increasing prediction accuracy. We show that latencies in hardware testbeds can be predicted at different quantiles with median relative errors between 8% and 29% using a simple GNN architecture. Furthermore, we show that network calculus bounds are especially useful for predicting higher quantiles and that they mostly correct large prediction errors.

CCS CONCEPTS

• **Networks → Network performance modeling**; *Network measurement*.

KEYWORDS

Graph Neural Network; Network Calculus; Delay Model; Latency Model; Hardware Measurement

ACM Reference Format:
Max Helm and Georg Carle. 2023. Predicting Latency Quantiles using Network Calculus-assisted GNNs . In *Proceedings of the 2nd Graph Neural Networking Workshop 2023 (GNNet '23), December 8, 2023, Paris, France.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3630049.3630173

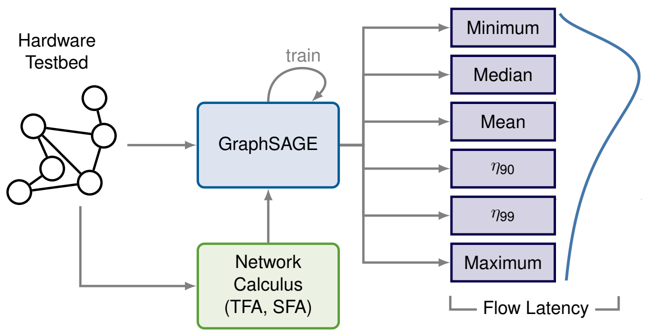

Figure 1: Overview of the approach. Different quantiles (minimum, median, η_{90}, η_{99}, maximum) and the mean are predicted using a GNN. The input consists of a network topology with flow- and node descriptions. Additionally, we derive network calculus worst-case bounds from the network description and use them as input features for the GNN.

1 INTRODUCTION

Network digital twins are an important tool in network management. Among other components, they consist of functional models [25]. Functional models for performance prediction typically take the network state as an input and predict network performance metrics. An important metric is the end-to-end latency of flows. The prediction of performance metrics can be done in a variety of ways, e.g., simulators, analytical methods, or machine-learning approaches.

A common approach in literature is the prediction of mean end-to-end latencies using Graph Neural Networks (GNNs). While scaling very well, related works in this area typically suffer of a subset of the following three disadvantages. First, they only predicts the mean of a latency distribution, while many applications are more reliant on other metrics, such as maximum latency. Second, additional information, such as results from formal methods, are rarely considered as input features. These results can provide additional information and are provably correct. Third, they rely on data obtained by network simulators while aiming to model the behavior of networks consisting of interconnected hardware devices.

We approach these three problems with the following methodology. The prediction target is extended to include more characteristic values of a latency distribution, namely a list of selected quantiles: minimum, median, the 90th percentile (η_{90}), the 99th percentile (η_{99}), and the maximum. Furthermore, we include worst-case latency bounds from the formal method network calculus (NC), as derived from network state, as input features for a GNN. Lastly, we use data obtained from optical wire tapping in a physical network with virtualized nodes where every packet traverses a Network Interface Card (NIC) and physical cable. Figure 1 shows an overview of the approach.

We provide access to our datasets and results.

The remainder of the paper is structured as follows. Section 2 gives an overview of the background, Section 3 surveys related work, Section 4 provides a detailed description of the methodology, Section 5 presents our results, the limitations are discussed in Section 6, before we conclude in Section 7.

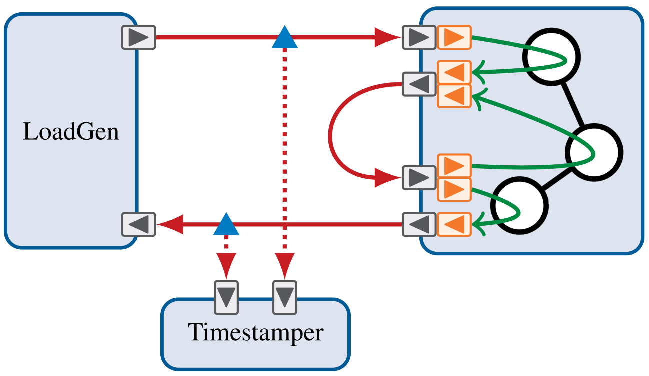

Figure 2: Measurement setup, adapted from [20]. Gray interfaces are physical, orange interfaces are virtual network functions. Red are physical cables, green is the flow path through virtual machines. We assume a single flow traversing from the top-, over the middle-, to the bottom node.

2 BACKGROUND

This section provides background information about virtualized latency measurements, network calculus, and GNNs.

2.1 Virtualized Latency Measurements

Latency measurements of packets can be done in a multitude of ways. They are typically more precise the closer the measurement point is to hardware. When measurements are done on large-scale topologies, a full hardware setup is costly. One approach tries to mitigate the scaling issue by utilizing virtual machines and tries to obtain precise results by staying close to hardware [20]. They provide a dataset of end-to-end flow latency measurements obtained by wiretapping optical cables in a virtualized topology [21]. A flow is defined as a sequence of UDP packets traversing the network on a static path from sender to receiver. We note that the employed virtualization technique, using virtual machines as nodes and virtualized network functions as NICs, has the benefit of sending each packet over a physical NIC and fiber optical cable. This means the nodes in the network are virtualized but each frame traverses physical network infrastructure. The measurement setup, consisting of three physical nodes, is shown in Figure 2. A load generator [6] produces traffic for all traffic sources in the network. The traffic is directed over different virtual network functions at the ingress and egress interfaces of the device hosting the virtualized topology, depending on its source and destination. The virtualized network functions are generated using SR-IOV [20]. All frames are timestamped using a dedicated timestamper, connected to the ingress and egress interface cables using optical splitters. We utilize this dataset to train and evaluate our model.

2.2 Network Calculus

Network Calculus (NC) is a mathematical framework for worst-case latency bounds in communication networks [5, 12]. It can be used to calculate flow-level worst-case upper-bound end-to-end latencies. The elemental objects of NC are arrival curves and service curves. An arrival curve is an envelope for the amount of data sent by a flow. A servive curve describes the amount of service a node can provide, i.e., how much data can be processed. These curves can

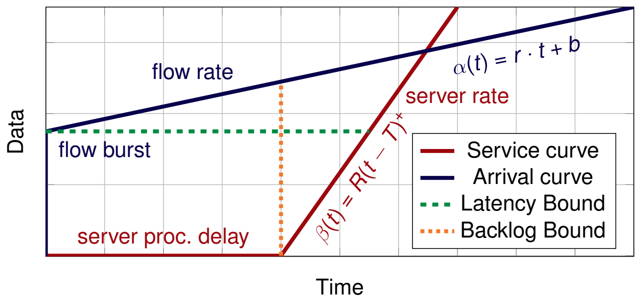

Figure 3: Arrival- and service curve, latency- and backlog bound

take arbitrary non-decreasing shapes, however, they are typically assumed to be concave and convex respectively. Simplified versions are described by the token-bucket arrival curve and the rate-latency service curve, piecewise linear functions with two parameters each. The parameters describe the flow burst (b) and long-term average rate (r), as well as the maximum processing delay (T) and minimum processing rate (R). A worst-case latency bound is calculated by determining the maximal horizontal deviation between the two curves. An example is shown in Figure 3. The close-form expression of the worst-case latency bound for token-bucket and rate-latency curves is given in Equation (1).

$$latency = T + \frac{b}{R} \quad | \quad r \leq R \tag{1}$$

Traversing multiple servers in series can be modeled using the convolution operator on all service curves to obtain a single service curve. Cross-traffic influences are modeled using left-over service curves, which are service curves from which cross-traffic is deducted. [12]

The tightness of the upper bound describes the accuracy of the method, i.e., a quantification of the overhead between the worst-case upper bound and the actual worst-case. This overhead is the result of simplifications of the real-world conditions (inputs to NC), or shortcomings of the method itself. There exist multiple NC methods, among others, Total Flow Analysis (TFA), Separate Flow Analysis (SFA), Pay Mulitplexing Only Once Analysis (PMOOA), Tandem Matching Analysis (TMA), Unique Linear Program (ULP). They typically provide a trade-off between tightness and computational cost. They form a partial order w.r.t. tightness due to the fact that the tightness between SFA and PMOOA is not decidable apriori [16]. The remaining methods form a total order. Note that especially SFA provides tighter bounds than TFA [1] because it utilizes the convolution- and left-over service curve operations.

2.3 Graph Neural Networks

Graph Neural Networks (GNNs) [14] are a neural network approach that can work directly on graph-structured data. They achieve better results on graph-structured data compared to other approaches by exploiting the permutation invariance property of graphs. This means that different representations of the same graph lead to the same results, which is not fundamentally true for other approaches, such as Convolutional Neural Networks for image processing. Geometric deep learning approaches, such as GNNs, can be considered generalizations of many of these other neural network

Work	Ref.	Year	GNN	Formal Method as Input	Data Source	Prediction Target
Rusek et al.	[13]	2020	✓	✗	Simulation	Normal, (gamma) distribution
Ferriol-Galmés et al.	[7]	2022	✓	✗	Simulation	Mean
Wang et al.	[19]	2022	✓	✗	Simulation	Mean per timestep
Yang et al.	[23]	2022	✓	✗	Simulation	Distribution (mean, η_{99} reported)
Zhang et al.	[24]	2023	✓	✓	Simulation	Mean
Suárez-Varela et al.	[17]	2023	✓	✗	Hardware Testbed	Mean
This Work	—	2023	✓	✓	Hardware Testbed	Mean + Quantiles

Table 1: An overview of related work in network performance prediction using GNNs

approaches [4]. An input graph is defined as $G = (V, E)$ where V is a set of vertices and E is a set of edges. Each vertex and edge can be associated with a vector of features. These inputs are encoded into two matrices, the feature matrix, and the adjacency matrix. During the training of GNNs, a message passing step and an aggregation step are performed. The message passing exchanges information along the edges of the graph, and the aggregation collects the exchanged information into a hidden state at each vertex. The aggregation is typically an invariant function, e.g., mean, sum, or maximum. There exist different architectures around this basic GNN concept, e.g., GCN [10], GraphSAGE [8], GAT [3, 18], GIN [22].

3 RELATED WORK

Rusek et al. proposed a GNN architecture to predict mean latencies as well as normal- and gamma distributions of latencies in SDN networks [13]. Ferriol-Galmés et al. improve on this by including multiple schedulers, traffic models, replaying captured traffic, and scaling the network size [7]. Wang et al. extend the prediction target from mean to mean per timestep by utilizing a factorization-based temporal approach [19]. Yang et al. predict latency distributions of simulated data and report mean and 99^{th} percentile results [23]. Zhang et al. extend the work of Ferriol-Galmés et al. by using NC results as inputs, reporting a decrease in prediction error of mean targets [24]. Suárez-Varela et al. provide a real-world dataset [17] and a GNN model architecture developed on simulation data. Table 1 provides an overview.

Our approach differs from related work by using a combination of formal-method-assisted GNNs, trained on measurements obtained from a hardware testbed, and predicting five different quantiles as point predictions in addition to mean values. Our approach does not consider complex scheduling mechanisms and different traffic models, like some related works do. Furthermore, we perform point predictions and do not predict distributions. While predicting distributions is generally the approach that should be preferred, our approach does not require us to make any assumptions about the target distribution, such as a normal distribution of values.

4 METHODOLOGY

This section describes the architecture, dataset, and training process of our approach.

4.1 Architecture

The architecture of our approach is shown in Figure 1. Starting from a network consisting of a topology, node specifications, and flow specifications, we derive NC worst-case latency bounds using the five methods TFA, SFA, PMOOA, TMA, and ULP. The network as well as the NC bounds are used as input to a GNN, in our case a GraphSAGE or GAT module. The GNN is trained to predict descriptors of the latency distribution of each flow. The descriptors are the mean as well as five quantiles: minimum, median, the 90^{th} percentile (η_{90}), the 99^{th} percentile (η_{99}), and the maximum.

Metric	Min.	Max.	$\sum$
Number of networks	—	—	100
Number of flows	20	60	—
Number of egress interfaces	5	25	—
Flow length	2	9	—
Number of measured latency values	—	—	14,012 M

Table 2: Metrics of the dataset

4.2 Dataset

The dataset [20] consists of 100 networks with unique topologies, node configurations, and flow configurations. For each network, there are measurements of the end-to-end flow latency of UDP flows. Measurements are performed three times per configuration, resulting in 175 data points (not all measurements were completed successfully). The latencies are obtained by optical wiretaps directly after the first egress interface and directly before the last node on the flow path. Table 2 provides an overview of metrics from the dataset. Figure 4 shows three example topologies and flow configurations.

Figure 5 shows a High Dynamic Range (HDR) histogram of the measured end-to-end latencies over all networks. We can observe stable latencies for lower quantiles and large outliers in latency for higher quantiles. This indicates that the prediction of quantiles above the median is a more complex task than predicting mean values. Especially the maximum latencies can reach extreme values that might not directly correlate with the topology or flow configurations of the network.

The NC worst-case latency bounds have been obtained by using the NCorg DNC [2, 15] with a custom parsing backend. Bounds for TFA and SFA have been obtained using the FIFO multiplexer, while bounds for PMOOA, TMA, and ULP have been obtained using the arbitrary multiplexer. Analysis of the latency bounds between methods has shown that there was no increase in tightness for any of the networks while using PMOOA, TMA, or ULP. Therefore, the remainder of the paper concentrates on using TFA and SFA bounds.

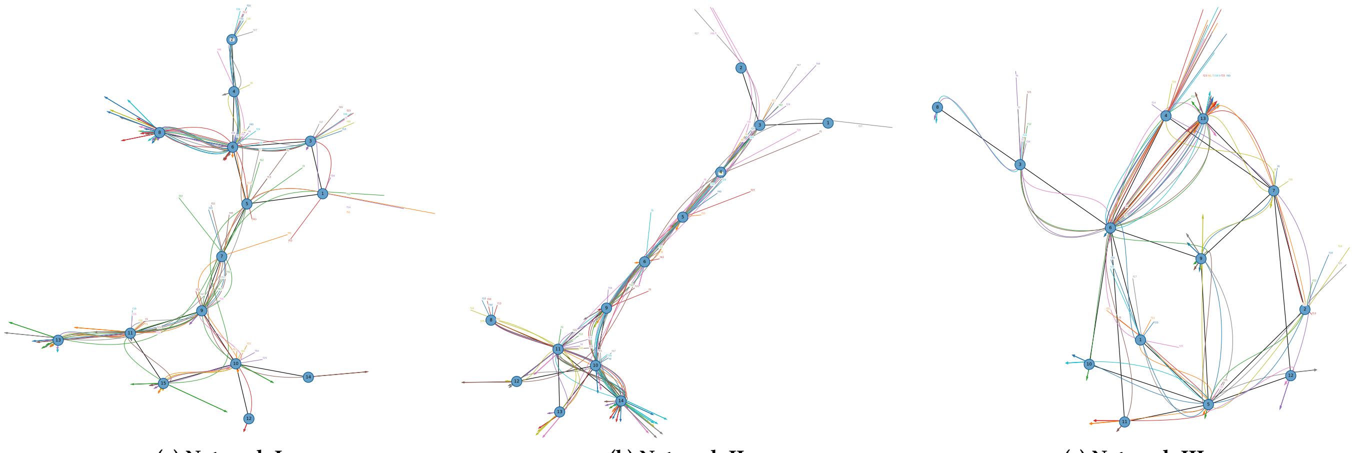
(a) Network I (b) Network II (c) Network III

Figure 4: Three example networks from the dataset, each line indicates the path of a single flow. Each node is a virtual machine connected to each adjacent node via a NIC and physical cable.

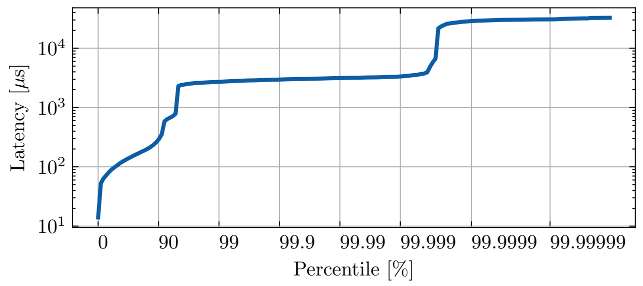

Figure 5: High Dynamic Range (HDR) histogram of end-to-end flow latencies in the dataset

4.3 Training

The training process is challenging because of the small size of the dataset. Therefore, we do not expect to obtain prediction errors that are comparable to predictions on simulated datasets in related work.

The training and inference are done by performing a 10-fold cross-validation. During training, we observed that GraphSAGE consistently provided better results compared to GAT. Therefore, all results in Section 5 are based on GraphSAGE. A list of all hyper-parameters can be found in Appendix A.

5 EVALUATION

This section presents and discusses our results.

5.1 Prediction Accuracy

Figure 6 shows the relative error for different prediction targets with and without NC bounds support. We can observe that lower quantiles are easier to predict while higher quantiles are harder to predict as we see an increase in relative error. The inclusion of NC bounds has no effect on the prediction of minimum and median latencies. However, mean, η_{90}, η_{99}, and maximum latency predictions benefit from the inclusion of NC bounds. The inclusion reduces the maximum as well as the third-quartile prediction error.

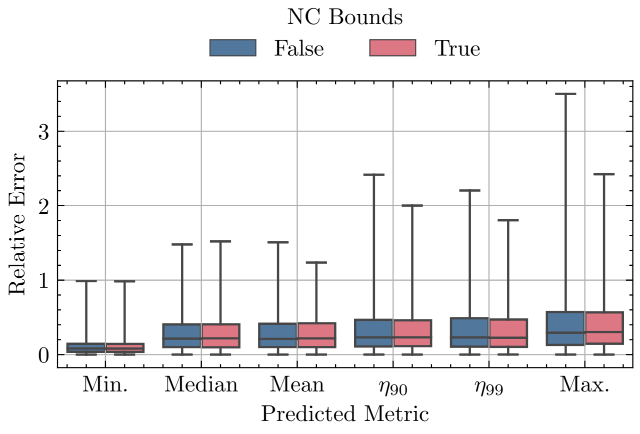

Figure 6: Relative error for different prediction targets with and without NC bounds as input parameters. Lower- and upper whiskers show minimum and maximum values respectively.

To find a reason for this, we look at the error behavior w.r.t different network parameters. Figure 7 shows the behavior of the relative error (over all prediction targets) for different flow lengths. We can observe that the relative error decreases with increasing flow length. This is caused by the instability of latencies for shorter flows. What we can further observe is that the inclusion of NC bounds especially helps in reducing the error for longer flows.

Figure 8 shows the relative error scaling with the number of flows in a network. We can observe that the number of flows has no correlation with the relative error except for occasional spikes. These spikes are mostly caused by latency outliers in the datasets of single networks. The large latency values are a result of interrupts in the measurement setup that are needed for operating the virtual machines [20]. These outliers heavily skew the maximum and percentile prediction targets without any correlation to the network configuration and state. They are therefore hard to predict without additional information.

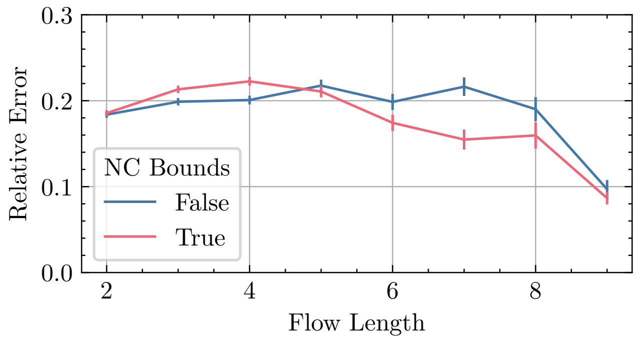

Figure 7: Relative error of predictions as they relate to flow length (in number of hops)

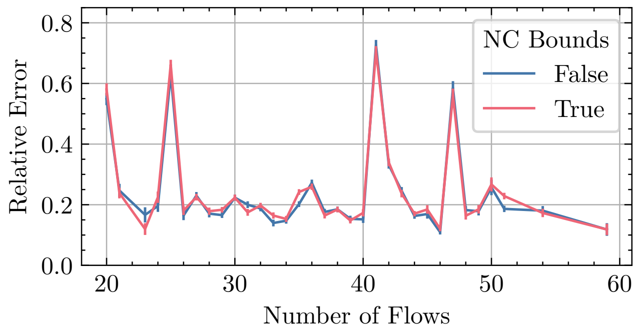

Figure 8: Relative error of predictions as they relate to the number of flows in a network

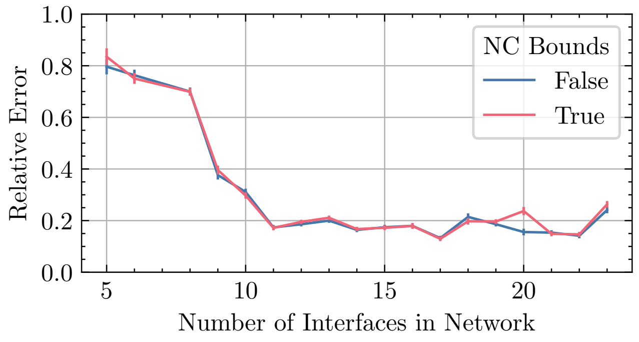

Figure 9: Relative error of predictions as they relate to the number of egress interfaces in a network

Figure 9 shows the behavior of the relative error for varying numbers of egress interfaces in a network. We can observe that the error is larger for smaller networks and that the error gets smaller the larger the network gets. This is again caused by latency instabilities in smaller networks, similar to the flow length.

Next, we want to quantify the influence of different input parameters on the prediction accuracy. Figure 10 shows the feature importance of each input parameter split into minimum, mean, and maximum prediction targets. Other quantiles did not differ significantly. We can observe that the egress interface link rate and the flow burst have a relatively large influence on the predictions. This

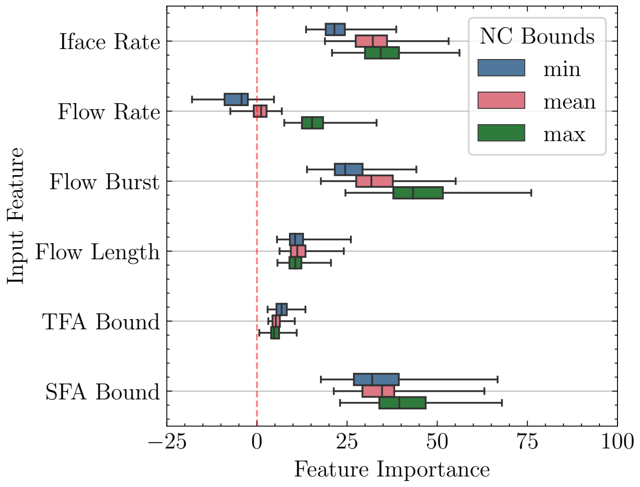

Figure 10: Feature importance of different input features of the GNN. Split by prediction targets (minimum, mean, maximum). Feature importance scores obtained using [11].

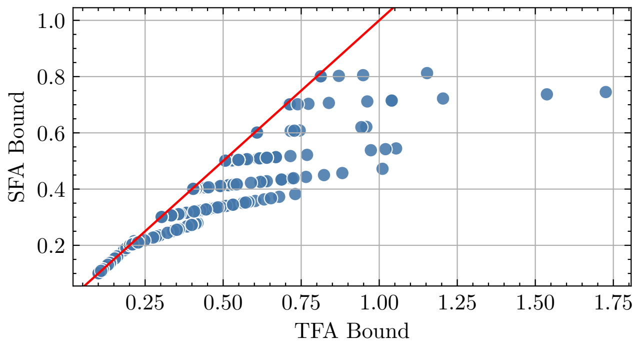

Figure 11: Comparison between obtained TFA- and SFA bounds. SFA bounds are always at least as tight as TFA bounds (red line).

makes sense since they are the main contributors to queue buildup and consequently latency increase. We can also see that their influence increases when predicting maximum values compared to mean or minimum. The NC bounds show an interesting behavior: TFA bounds are mostly ignored for predictions while SFA bounds have a large influence. Additionally, we can see that SFA bounds are more important for predicting maximum latencies, as expected.

To identify why TFA bounds are largely ignored, we look at the relationship between them and SFA bounds on a per-flow basis. This is shown in Figure 11. We can observe that TFA bounds consistently overestimate the worst-case latency bounds. This is a known concept called Pay-Bursts-Only-Once which leads to the tighter latency bounds of SFA [12].

5.2 Negative Results

Using other loss functions did not prove helpful in decreasing relative errors. Specifically, we used a quantile loss function for the six prediction targets with quantile weights of 0.01, 0.5, 0.5, 0.90, 0.99, and 0.999. This approach did not improve the results compared to using the loss specified in Appendix A.

6 LIMITATIONS

We have a dataset of only 100 networks, which is due to the expensive nature of obtaining measurement data from hardware testbeds. Therefore, the reported relative error values are not able to compete with related work which is based on simulated network latencies. This leads our focus to the relative differences between errors of different prediction targets and NC-bound inclusion. Furthermore, we only consider a stateless transport protocol (UDP), we refer to Jaeger et al. [9] for models of a stateful transport protocol (TCP). Due to the nature of NC, it is only possible to apply this method to networks that don't contain links with a utilization larger than 100%. To include such networks, it would be necessary to include a binary input feature that indicates infinite latency bounds.

7 CONCLUSION

We provide an approach to predict end-to-end flow latency quantiles, as measured in a hardware testbed, using a simple GNN. Additionally, we incorporate worst-case end-to-end latency bounds from the network calculus framework as input features to the GNN. Showing that predicting higher quantiles becomes increasingly difficult and that network calculus bounds can remedy this to some degree. We provide access to our dataset and to our results[1].

ACKNOWLEDGMENTS

European Union Horizon 2020 (project SLICES-SC, 101008468, and SLICES-PP, 101079774). Bavarian Ministry of Economic Affairs, Regional Development and Energy (project 6G Future Lab Bavaria). German Federal Ministry of Education and Research (project 6G-life, 16KISK002, and project 6G-ANNA, 16KISK107).

REFERENCES

[1] Steffen Bondorf. 2016. Worst-Case Performance Analysis of Feed-Forward Networks–An Efficient and Accurate Network Calculus. (2016).

[2] Steffen Bondorf and Jens B. Schmitt. 2014. The DiscoDNC v2 – A Comprehensive Tool for Deterministic Network Calculus. In *Proc. of the International Conference on Performance Evaluation Methodologies and Tools (ValueTools '14)*. 44–49. https://dl.acm.org/citation.cfm?id=2747659

[3] Shaked Brody, Uri Alon, and Eran Yahav. 2021. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491* (2021).

[4] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velickovic. 2021. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. *CoRR* abs/2104.13478 (2021). arXiv:2104.13478 https://arxiv.org/abs/2104.13478

[5] Rene L Cruz. 1991. A Calculus for Network Delay. I. Network Elements in Isolation. *IEEE Transactions on information theory* 37, 1 (1991), 114–131.

[6] Paul Emmerich, Sebastian Gallenmüller, Daniel Raumer, Florian Wohlfart, and Georg Carle. 2015. MoonGen: A Scriptable High-Speed Packet Generator. In *Internet Measurement Conference 2015 (IMC'15)*. Tokyo, Japan.

[7] Miquel Ferriol-Galmés, Krzysztof Rusek, José Suárez-Varela, Shihan Xiao, Xiang Shi, Xiangle Cheng, Bo Wu, Pere Barlet-Ros, and Albert Cabellos-Aparicio. 2022. Routenet-Erlang: A Graph Neural Network for Network Performance Evaluation. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2018–2027.

[8] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[9] Benedikt Jaeger, Max Helm, Lars Schwegmann, and Georg Carle. 2022. Modeling TCP Performance Using Graph Neural Networks. In *Proceedings of the 1st International Workshop on Graph Neural Networking* (Rome, Italy) *(GNNet '22)*. Association for Computing Machinery, New York, NY, USA, 18â€“23. https://doi.org/10.1145/3565473.3569190

[10] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[11] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, et al. 2020. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896* (2020).

[12] Jean-Yves Le Boudec and Patrick Thiran. 2001. *Network Calculus: a Theory of Deterministic Queuing Systems for the Internet.* Vol. 2050. Springer Science & Business Media.

[13] Krzysztof Rusek, José Suárez-Varela, Paul Almasan, Pere Barlet-Ros, and Albert Cabellos-Aparicio. 2020. Routenet: Leveraging Graph Neural Networks for Network Modeling and Optimization in SDN. *IEEE Journal on Selected Areas in Communications* 38, 10 (2020), 2260–2270.

[14] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The Graph Neural Network Model. *IEEE transactions on neural networks* 20, 1 (2008), 61–80.

[15] Alexander Scheffler and Steffen Bondorf. 2021. Network Calculus for Bounding Delays in Feedforward Networks of FIFO Queueing Systems. In *Proc. of the 18th International Conference on Quantitative Evaluation of Systems (QEST '21)*. 149–167. https://link.springer.com/chapter/10.1007/978-3-030-85172-9_8

[16] Jens B Schmitt, Frank A Zdarsky, and Markus Fidler. 2008. Delay bounds under arbitrary multiplexing: When network calculus leaves you in the lurch.... In *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*. IEEE, 1669–1677.

[17] José Suárez-Varela et al. 2021. The graph neural networking challenge: a worldwide competition for education in AI/ML for networks. *ACM SIGCOMM Computer Communication Review* 51, 3 (2021), 9–16.

[18] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. 2017. Graph attention networks. *stat* 1050, 20 (2017), 10–48550.

[19] Mowei Wang, Linbo Hui, Yong Cui, Ru Liang, and Zhenhua Liu. 2022. xNet: Improving Expressiveness and Granularity for Network Modeling with Graph Neural Networks. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*. 2028–2037. https://doi.org/10.1109/INFOCOM48880.2022.9796726

[20] Florian Wiedner et al. 2022. HVNet: Hardware-Assisted Virtual Networking on a Single Physical Host. In *IEEE INFOCOM WKSHPS CNERT 2022*. Virtual Event.

[21] Florian Wiedner et al. 2022. HVNet: Hardware-Assisted Virtual Networking on a Single Physical Host. https://mediatum.ub.tum.de/1638129

[22] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[23] Qingqing Yang, Xi Peng, Li Chen, Libin Liu, Jingze Zhang, Hong Xu, Baochun Li, and Gong Zhang. 2022. DeepQueueNet: Towards Scalable and Generalized Network Performance Estimation with Packet-level Visibility. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 441–457.

[24] Lianming Zhang, Benle Yin, Qian Wang, and Pingping Dong. 2023. Graph Neural Network-based Delay Prediction Model Enhanced by Network Calculus. In *2023 IFIP Networking Conference (IFIP Networking)*. IEEE, 1–7.

[25] Cheng Zhou, Hongwei Yang, Xiaodong Duan, Diego Lopez, Antonio Pastor, Qin Wu, Mohamed Boucadair, and Christian Jacquenet. 2023. *Digital Twin Network: Concepts and Reference Architecture.* Internet-Draft draft-irtf-nmrg-network-digital-twin-arch-04. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-irtf-nmrg-network-digital-twin-arch/04/ Work in Progress.

A APPENDIX

Parameter	Value
Batch size	8
Dropout Linear	0.3
Dropout GRU	0.1
Epochs	300
Hidden size	4
Jumping knowledge	True
Learning rate	0.0005
Loss function	HuberLoss
LR scheduler	ReduceLearningRateOnPlateau
LR scheduler factor	0.7
Model	GraphSAGE
Number of message passing steps	4
Train-test split	0.8
Weight decay	0.0

Table 3: Hyperparameters for all models

[1] https://gitlab.lrz.de/gnnet2023/quantile-nc-prediction