DIGITAL TWIN OPPORTUNITIES WITH LEVERAGING GRAPH NEURAL NETWORKS ON REAL NETWORK DATA

Kaan Aykurt[1], Maximilian Stephan[1], Serkut Ayvasik[1], Johannes Zerwas[1], Wolfgang Kellerer[1]
[1]Chair of Communication Networks, Technical University of Munich

{kaan.aykurt, maximilian.stephan, serkut.ayvasik, johannes.zerwas, wolfgang.kellerer}@tum.de

Abstract – *Sixth-generation networks propose integrating multiple networks while ensuring seamless network performance. Hence, networks are becoming increasingly complex while the traditional methods to manage networks are facing significant challenges as the topology sizes, traffic patterns, and network domains are changing. Autonomous network management solutions, which are often built on digital twins, are emerging as possible candidates for addressing these challenges.*

Machine learning models are widely used for realizing digital twins. Among many neural network structures, graph neural networks are a subclass of promising machine learning methods that perform well in graph-structured data such as network topologies. In this paper, we explore GNN performance on real network data and present our solution to per-flow mean delay prediction which achieves a MAPE of 35.39%, improving the baseline solutions by over 20% together with additional findings and further improved models for Graph Neural Networking Challenge 2023.

Keywords – Autonomous network management, digital twins, graph neural networks

1. INTRODUCTION

Networks serve as the backbone of our interconnected world, ranging from social networks and transportation systems to communication infrastructures and the Internet itself. Understanding and autonomously managing these networks is crucial for optimizing performance, ensuring robustness, and proactively addressing potential challenges. The traditional approaches to network management often face problems while capturing the relationships and dynamic behaviors inherent in emerging novel forms of networks (e.g. body area networks, vehicular networks, and satellite networks). The increasing number of nodes, distinct network characteristics, and topologies render the existing network management solutions unsuitable for ensuring network guarantees in real time.

To address this challenge, the current trend in the literature is shifting the focus from manual configurations to self-driving and autonomously-managed networks. State-of-the-art Machine Learning (ML) methods power autonomous network management paradigms. An example of the emerging ML methods widely considered is Graph Neural Networks (GNNs), a subclass of ML models tailored for tasks involving graph-structured data. GNNs perform well in learning representations of nodes and edges within a graph, making them well-suited for modeling real-world networks [1]. However, while GNNs have demonstrated promising capabilities when trained on synthetic data, this approach has inherent limitations. Network simulators and synthetic datasets, while useful for initial model training and evaluation, do not fully capture the complexities and unpredictabilities of real-world network environments. Simulated data often lacks the intricate details of real network traffic, such as unexpected congestion patterns, hardware-induced latencies, and environmental variables that can significantly impact network performance [2]. Our paper investigates the possibility of creating a network digital twin that is based on real-world data by leveraging GNNs. By using actual network statistics, traffic patterns, and performance metrics, our approach aims to bridge the gap between theoretical models and the unpredictable nature of real networks.

Specifically, this article explores the creation of a network digital twin with a specific emphasis on leveraging GNNs to model and evaluate real network data. The remainder of this paper is structured as follows: Section 2 provides an overview of the background. Section 3 describes the state of the research in the literature. Section 4 discusses our approach and Section 5 presents the findings. We finally conclude and present the digital artifacts in Section 6.

2. BACKGROUND

This section provides an overview of the challenge and basic information on GNNs.

2.1 Challenge description

This work is developed in the context of the Graph Neural Networking Challenge 2023 *Creating a Network Digital Twin with Real Network Data* [3, 4]. The goal of this challenge is to create a real network digital twin based on a GNN model that can accurately estimate flow level delay. In recent years, modern GNN architectures enabled building lightweight and accurate network digital twins that can operate faster than real time, hence enabling *what-if* analysis. However, since these models were mainly based on synthetic data, existing ML-based network digital twins were mainly developed and trained using simulation data. The synthetic nature of training datasets puts a strong limitation against our understanding of the real network characteristics that lay behind real datasets. Hence, this challenge aims to explore the possibility of building a network digital twin based on real network data.

2.2 Requirements and restrictions

This challenge involves two datasets generated from a real network testbed, featuring various topologies, routing configurations, and traffic flows. The first dataset uses Constant-Bit Rate (CBR) traffic, while the second employs Multi-Burst Rate (MBR). The testbed emulates realistic network conditions, capturing the nuances of real traffic behavior and network interactions. Each dataset consists of samples labeled with average per-flow performance measurements. The objective is to predict the mean per-flow delay.

A test dataset, with similar distributions but lacking delay information, is also provided for ranking solutions based on the Mean Absolute Percentage Error (MAPE) score. The test dataset excludes flow performance measurements such as delay, jitter, and loss, prohibiting their use as input for proposed solutions. RouteNet-Fermi [5], a GNN architecture, serves as a baseline for the challenge with an open-source implementation [6].

2.3 Digital twins

In the era of rapid technological advancements, the concept of a digital twin is emerging as a powerful paradigm that promises to provide a virtual representation of physical entities. While digital twins have found widespread application across various domains, in recent years, their integration into the networking domain is gaining increased attention due to modern networks' complex interdependencies and dynamic nature.

Digital twins represent the mapping of a physical world into a digital space, which offers a real-time digital counterpart of a physical object or a process. By mirroring a physical entity in a digital realm, digital twins facilitate the modeling of an entity to predict its future states and enable remote control. Hence, the networking community's interest in digital twins for network modeling and management has grown enormously in the last few years [7]. The potential applications of a digital twin can help to troubleshoot network problems, detect anomalies, and apply *what-if* analysis faster than real time [8]. Hence, the operations, that are unfeasible in a physical network become viable with digital twins.

Representing the physical world in its digital counterpart require accuracy and preferably fast execution. Hence, ML models are emerging as one of the possible candidates for realizing digital twins. For network performance modeling, the recent developments in GNN-based ML architectures are gaining popularity. Since the networks can essentially be modeled as graphs, the state-of-the-art network performance analysis implementations, such as RouteNet [9] and xNet [2] rely on GNN-based neural network architectures.

2.4 Graph neural networks

In recent years, GNNs [10] have become increasingly popular for representing data in the form of graphs. Unlike traditional neural networks that assume a fixed-dimensional input space and predetermined connection structures, GNNs dynamically determine their architecture based on input graphs. This flexibility allows them to leverage the inherent graph structure of input data, facilitating the modeling of relationships between elements in the graph.

From a network management perspective, the ultimate goal for network managers is to oversee actions taken by the autonomous network management framework. GNN-based ML models offer the potential to accurately predict network characteristics, serving as a powerful tool for network analysis and autonomous network management. The structural similarity of GNNs to computer networks ensures that predicted results are meaningful and suitable for analysis. Additionally, GNNs exhibit promising capabilities to generalize to unseen data, topologies, and traffic characteristics [11], which make them a suitable candidate for a successful autonomous network management framework.

3. RELATED WORK

Recent advancements in ML have led to its extensive use in network analysis, as detailed in Fadlullah et al.'s survey [12]. This research highlights various ML methods for tasks like traffic classification and latency analysis. The literature shows diverse ML model applications, from support vector regression for analyzing queuing and latency [13] to causal Bayesian networks [14]. These methods are versatile but recent studies are exploring graph-based data structures in networks more deeply.

GNNs, introduced by Gori et al. [1], represent graph-structured data through message passing between graph nodes. GNNs have been applied in various fields, from object localization to webpage ranking, and now in network analysis. They offer a faster alternative to traditional network analysis methods. Deep-Q [15] and RouteNet [9] are examples of GNN implementations

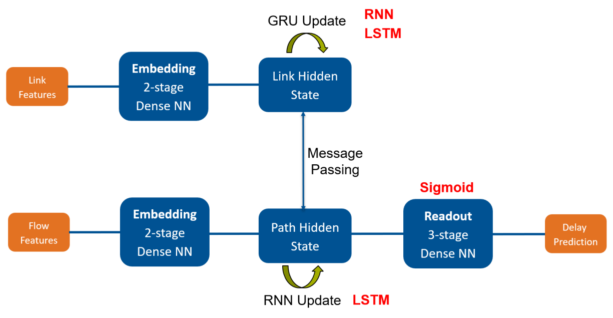


Figure 1 – Architecture of our proposed model.

in network analysis, focusing on path delay prediction and key network characteristics, respectively. Another recent development, xNet [2], aims to replace simulation studies for Data Center Network (DCN) topologies. However, current GNN approaches are limited as they rely on simulation data, which may not accurately reflect hardware characteristics and challenges in modeling complex topologies.

The potential for digital twins in network management is greatly enhanced by the rapid execution of GNNs. Ferriol et al. [16] and Almasan et al. [8] have discussed the possibilities of using GNNs for digital twins.

In this context, our work aims to develop a GNN architecture trained on real hardware data to create an autonomous network management framework aided by the digital twin paradigm. Our approach addresses the current limitations of GNN-based network analysis by exposing them to real network data on training.

4. METHODOLOGY

This section provides an overview of our approach, GNN model, and the training testbed.

4.1 Model

Our GNN model offers significant opportunities for advancing digital twin implementations in network management. By accurately predicting flow delays and understanding network behavior under various conditions, the model can support what-if analysis, allowing network operators to simulate potential scenarios and their impacts without affecting the live network. This capability is particularly valuable in proactive network management, where potential issues can be anticipated and mitigated before they affect users. Moreover, the integration of real network data enhances the model's applicability to real-world scenarios, making it a powerful tool for dynamic and autonomous network management.

Our model is based on the baseline model RouteNet-Fermi [5]. Fig. 1 shows a block diagram that represents the architecture of the GNN algorithm. The algorithm processes two types of input features: `Link Features` and `Flow Features`. Both sets of inputs are first processed by separate `Embedding` blocks, each described as a `2-stage Dense NN`, indicating two stages of dense neural networks used for embedding. Although both embeddings share a similar structure, the different input dimensions of link and flow features require separate embedding stages.

The embedded features from the `Link Features` are then fed into a `Link Hidden State` block, where they undergo a `Gated Recurrent Unit (GRU) Update`. Here, GRU captures the temporal relationship between the link features. Simultaneously, the embedded `Flow Features` are passed to a `Path Hidden State` block. The state of this block is updated using an `RNN Update`.

The `Link Hidden State` and `Path Hidden State` are then involved in a `Message Passing` process, where information is passed along the edges of the graph. The result of the message passing is then fed into a `Readout` block described as a `3-stage Dense NN`. Finally, the output of the readout block is processed for computing the delay.

Additionally, alternatives to the blocks, which are presented in red, indicate the different functions to improve the model. Mainly, our work focused on capturing the temporal dependencies in the link and path hidden states. For this reason, we used RNNs and LSTMs for the hidden state updates, and a Sigmoid activation function in the readout layer instead of a Softplus activation. However, our final model selection was adapted from the provided baseline model RouteNet-Fermi [5].

4.2 Features and hyperparameters

Table 1 lists the hyperparameters chosen for the training of the GNN. The hyperparameters are chosen from RouteNet [9] training settings without any fine tuning. It specifies the dimensions of the link and path state representations as well as the size of the readout layer, each set to 64. The model performs eight rounds of message passing. The learning rate is set to 0.001 and is adaptive, decreasing by 99% after each epoch. The Adam optimizer is used for its efficiency in handling sparse gradients and adaptive learning rate capabilities. The loss function employed is the MAPE. Early stopping is implemented to prevent overfitting, with the condition that training will stop if the change in validation loss is less than 0.0001.

Table 2 outlines the set of features utilized in the training of the model. These features capture various aspects of network traffic and topology, such as the average bandwidth per flow, the number of packets per flow, and the size of the packets. It also includes more structural characteristics like flow type, flow length, link capacity, and the topology of the network represented by lists of

Table 1 – Hyperparameters used for training the graph neural network model.

Parameter	Value
Link State Dimensions	64
Path State Dimensions	64
Readout Layer Size	64
Message Passing Rounds	8
Learning Rate	0.001 (Adaptive)
Optimizer	Adam
Loss	MAPE
Early Stopping	Δvalidation_loss $< 1e-4$

Table 2 – Input features used for the graph neural network model.

Features
Average bandwidth per flow (normalized)
Number of packets per flow (normalized)
Size of the generated packets per flow (normalized)
Flow type
Flow length (physical path)
Link capacity
List of links traced per flow
List of flows per link
Inter packet gap per flow (μ, σ^2, **percentiles)**
Percentage of packet losses per flow

links per flow and flows per link. All of these features are present in the provided baseline for the challenge.

The last two features, highlighted in bold, represent an extension to the baseline feature set. The `Inter gap per flow` is defined as the time interval between the transmission of consecutive packets within a single flow of data in a network. It is described by its mean, variance, and percentiles. The inclusion of every 10^{th} percentile reveals the underlying distribution of the inter-packet gap. Therefore, the inter-packet gap helps to capture the burstiness, and hence the likeliness of a flow to suffer from congestion, which acts as an explaining factor to the flow level delay. The `Percentage of packet losses per flow` is computed by calculating the percentage of lost packets in a flow. Since packet losses are a good indicator of congestion which majorly contributes to delay, it also serves as an explanatory factor for the flow level delay.[1]

4.3 Testbed

Fig. 2 illustrates the architecture of the testbed environment designed for training ML jobs. The setup consists of four servers, labeled from Server 1 to Server 4, each equipped with two GPUs. These GPUs are indicated as GPU0 and GPU1 within each server, and training tasks are containerized using Docker, hence making the deployment and reproducibility of our work far easier. All servers are interconnected through a centralized switch, facilitating communication and data exchange with the

[1]Packet loss information was not available in the challenge test dataset. Hence, it is not included in this paper.

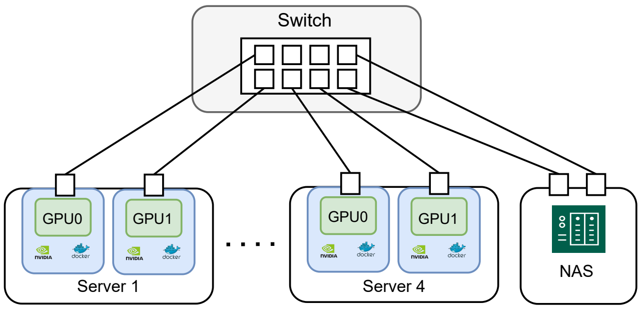


Figure 2 – Architecture of the testbed environment for training ML jobs.

Network Attached Storage (NAS). We offload the training tasks to our compute testbed to facilitate the training time. For the scope of this work, our approaches do not employ training parallelism.

5. EVALUATION

This section compares the alternative models considered and investigates the importance of normalized features.

5.1 Model performance

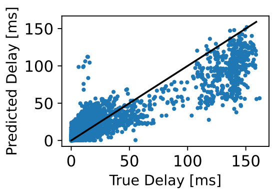
Table 3 provides a performance comparison of three distinct models based on the MAPE metric. The compared models include *Baseline Model*, *Model with IPG*, and *Extended Model with Packet Loss*. The *Model with IPG* indicates the adapted baseline model with the `Inter gap per flow` feature. Finally, the *Extended Model with Packet Loss* refers to the extended version of the *Model with IPG*. The extension of the model is done with the inclusion of the `Percentage of packet losses per flow` feature.

The *Baseline Model*, which serves as a reference point, achieved a MAPE of 45.00%. *Model with IPG*, which includes the extended inter-packet gap feature, showed improved performance with a MAPE of 35.39%. Finally, the results show that the *Extended Model with Packet Loss*, outperformed the other models with a MAPE of 23.25%, indicating a substantial enhancement in the target loss metric, MAPE.

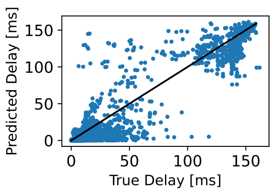
Fig. 3 presents a scatter plot of model predictions against the true delay values. In an ideal scenario without any prediction error, the points are expected to lie on the diagonal axis. Fig. 3(a) shows the scatter plot for *Model with IPG*, and Fig. 3(b) illustrates *Extended Model with Packet Loss*. The plots show that the predictions of the *Extended Model with Packet Loss* lie closer to the correct values on average, which is also reflected with a lower MAPE of 23.25% in comparison to 35.39%.

Table 3 – Comparison of model performances based on MAPE.

Model	MAPE
Baseline Model	45.00%
Model with IPG	35.39%
Extended Model with Packet Loss	23.25%



(a) Model with IPG.



(b) Extended Model with Packet Loss.

Figure 3 – Scatter plot of delay predictions of the models against the ground truth.

5.2 Feature importance

Analyzing feature importance is crucial for understanding the mechanics of the model performance. Unlike traditional ML methods, assessing the feature importance of GNNs is not viable via traditional Principal Component Analysis (PCA). Therefore, to analyze the feature importance, we follow a method to introduce noise to inputs and check the decline or improvement in the prediction performance as described in [17].

Specifically, we introduce a random Gaussian noise $\mathcal{N}(0, 1)$ to the inputs and check the percentage change in test MAPE. Since the range of values in the non-normalized features differ, we only introduce the noise to the normalized features. Fig. 4 shows the horizontal bar plot of feature importance for the size of generated packets per flow, number of packets per flow, and average bandwidth per flow features. The analysis shows that perturbing any of the normalized inputs to the model results in a decrease in the test MAPE. Quantitatively, introducing noise to the size of the generated packets per flow leads to a 7.41% increase in MAPE, whereas the number of generated packets per flow leads to 3.74%, and average bandwidth per flow leads to a 17.40% increase in MAPE due to the increased congestion in the network. This indicates that the most important feature contributing to a better model performance among the normalized features is the average bandwidth per flow.

6. CONCLUSION

This paper presented an extended GNN model designed to enhance flow delay prediction. Our models with extended features have improved the baseline MAPE from 45% to 35.39% and 23.25% respectively, resulting in an improvement of 48.3%. The feature importance analysis highlighted the criticality of certain features, with the average bandwidth per flow being the most significant.

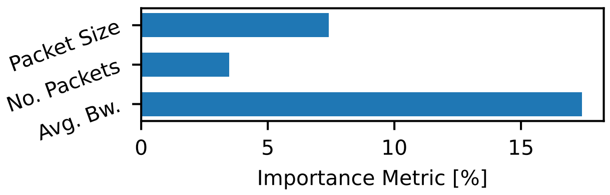


Figure 4 – Bar plot of feature importance for the normalized features. Feature labels are shown on the y-axis, whereas the target importance metric which is the percentage change in test MAPE is shown on the x-axis.

Our results contribute valuable insights to the field, potentially guiding the development of more effective autonomous network management frameworks aided by GNNs exposed to real network data. Finally, for reproducibility reasons, our code and artifacts are available under GitHub[2].

ACKNOWLEDGEMENT

We acknowledge the financial support by the Federal Ministry of Education and Research of Germany (BMBF) in the program of "Souverän. Digital. Vernetzt." joint project 6G-life, project identification number 16KISK002.

REFERENCES

[1] Marco Gori, Gabriele Monfardini, and Franco Scarselli. "A new model for learning in graph domains". In: *Proc. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 2. IEEE. 2005, pp. 729–734.

[2] Mowei Wang, Linbo Hui, Yong Cui, Ru Liang, and Zhenhua Liu. "xnet: Improving expressiveness and granularity for network modeling with graph neural networks". In: *IEEE INFOCOM 2022*. IEEE. 2022, pp. 2028–2037.

[3] Nov. 2023. URL: `https://bnn.upc.edu/challenge/gnnet2023/`.

[4] José Suárez-Varela et al. "The graph neural networking challenge: a worldwide competition for education in AI/ML for networks". In: *ACM SIGCOMM Computer Communication Review* 51.3 (2021), pp. 9–16.

[5] Miquel Ferriol-Galmés, Jordi Paillisse, José Suárez-Varela, Krzysztof Rusek, Shihan Xiao, Xiang Shi, Xiangle Cheng, Pere Barlet-Ros, and Albert Cabellos-Aparicio. "RouteNet-Fermi: Network Modeling With Graph Neural Networks". In: *IEEE/ACM Transactions on Networking* (2023).

[6] Jan. 2024. URL: `https://github.com/BNN-UPC/GNNetworkingChallenge/tree/2023_RealNetworkDT`.

[2]https://github.com/ITU-AI-ML-in-5G-Challenge/lkn-tum-gnnchallenge2023

[7] Yiwen Wu, Ke Zhang, and Yan Zhang. “Digital Twin Networks: A Survey”. In: *IEEE Internet of Things Journal* 8.18 (2021), pp. 13789–13804. DOI: 10.1109/JIOT.2021.3079510.

[8] Paul Almasan, Miquel Ferriol-Galmés, Jordi Paillisse, José Suárez-Varela, Diego Perino, Diego López, Antonio Agustin Pastor Perales, Paul Harvey, Laurent Ciavaglia, Leon Wong, et al. “Digital twin network: Opportunities and challenges”. In: *arXiv preprint arXiv:2201.01144* (2022).

[9] Miquel Ferriol-Galmés, Krzysztof Rusek, José Suárez-Varela, Shihan Xiao, Xiang Shi, Xiangle Cheng, Bo Wu, Pere Barlet-Ros, and Albert Cabellos-Aparicio. “Routenet-erlang: A graph neural network for network performance evaluation”. In: *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE. 2022, pp. 2018–2027.

[10] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. “The graph neural network model”. In: *IEEE transactions on neural networks* 20.1 (2008), pp. 61–80.

[11] José Suárez-Varela, Sergi Carol-Bosch, Krzysztof Rusek, Paul Almasan, Marta Arias, Pere Barlet-Ros, and Albert Cabellos-Aparicio. “Challenging the generalization capabilities of graph neural networks for network modeling”. In: *Proc. ACM SIGCOMM Conference Posters and Demos*. 2019, pp. 114–115.

[12] Zubair Md Fadlullah, Fengxiao Tang, Bomin Mao, Nei Kato, Osamu Akashi, Takeru Inoue, and Kimihiro Mizutani. “State-of-the-art deep learning: Evolving machine intelligence toward tomorrow’s intelligent network traffic control systems”. In: *IEEE Communications Surveys & Tutorials* 19.4 (2017), pp. 2432–2455.

[13] Mariyam Mirza, Joel Sommers, Paul Barford, and Xiaojin Zhu. “A machine learning approach to TCP throughput prediction”. In: *ACM SIGMETRICS Performance Evaluation Review* 35.1 (2007), pp. 97–108.

[14] Mukarram Bin Tariq, Kaushik Bhandankar, Vytautas Valancius, Amgad Zeitoun, Nick Feamster, and Mostafa Ammar. “Answering “what-if” deployment and configuration questions with WISE: Techniques and deployment experience”. In: *IEEE/ACM Transactions on Networking* 21.1 (2013), pp. 1–13.

[15] Shihan Xiao, Dongdong He, and Zhibo Gong. “Deep-q: Traffic-driven qos inference using deep generative network”. In: *Proc. 2018 Workshop on Network Meets AI & ML*. 2018, pp. 67–73.

[16] Miquel Ferriol-Galmés, José Suárez-Varela, Jordi Paillissé, Xiang Shi, Shihan Xiao, Xiangle Cheng, Pere Barlet-Ros, and Albert Cabellos-Aparicio. “Building a digital twin for network optimization using graph neural networks”. In: *Computer Networks* 217 (2022), p. 109329.

[17] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. *Captum: A unified and generic model interpretability library for PyTorch*. 2020. arXiv: 2009.07896 [cs.LG].

AUTHORS

Kaan Aykurt received his M.Sc. degree in Communications Engineering from the Technical University of Munich (TUM), Germany, in 2022. He joined the Chair of Communication Networks at the TUM as a research and teaching associate in May 2022. His research is focused on data center networks, multi-domain autonomous network management, and applications of machine learning in communication networks.

Maximilian Stephan received his M.Sc. degree in Electrical and Computer Engineering from the Technical University of Munich (TUM), Germany, in 2021. He joined the Chair of Communication Networks at TUM as a research and teaching associate in 2021. His research is in the area of network traffic monitoring and analysis. Here, he focuses on stochastic network state representations for anomaly detection and network behavior characterization

Serkut Ayvasik received his Bachelor of Science degree in Electrical and Electronics Engineering at Middle East Technical University (METU), Turkey in 2016. He obtained the M.Sc. Degree in Communications Engineering in February 2019 with high distinction from Technical University of Munich (TUM). In March 2019, he joined the Chair of Communications Networks at TUM as a research and teaching associate. His current research interests are design and evaluation of radio resource management for URLLC applications and machine learning in 5G/6G wireless networks.

Johannes Zerwas received the M.Sc. degree in Electrical Engineering and Information Technology in 2018 and the Dr.-Ing. degree from TUM in 2024. He joined the Chair of Communication Networks at the TUM as a research and teaching associate in February 2018. His research is focused on reconfigurable network topologies for data center and wide area networks, and data-driven networking algorithms.

Wolfgang Kellerer (M'96, SM'11) is a Full Professor with the Technical University of Munich (TUM), heading the Chair of Communication Networks at the Department of Electrical and Computer Engineering. Before, he was for over ten years with NTT DOCOMO's European Research Laboratories. He currently serves as an associate editor for IEEE Transactions on Network and Service Management and as the area editor for Network Virtualization for IEEE Communications Surveys and Tutorials.