DEVELOPING A LIBRARY OF PROBLEM SCENARIOS FOR TIME-SENSITIVE AND REAL-TIME NETWORKING

Alexej Grigorjew[1], Viktoria Vomhoff[1], Stefan Geißler[1], Philip Diederich[2], Tobias Hoßfeld[1], Wolfgang Kellerer[2]

[1]University of Würzburg, Chair of Communication Networks, Würzburg, Germany
Email: <firstname>.<lastname>@uni-wuerzburg.de

[2]Technical University of Munich, Chair of Communication Networks, Munich, Germany
Email: <firstname>.<lastname>@tum.de

ABSTRACT

The lack of realistic problem instances is one of the major challenges when it comes to evaluating and comparing mechanisms and approaches in the area of real-time networking. To this end, we aim at establishing a library of general as well as use case specific scenarios including different network topologies and traffic streams with the goal of promoting comparability among research works and facilitate the development of innovative approaches. We present our foundational framework for the generation of these synthetic problem instances and call on the community to contribute their experiences and insights.

1 Introduction

As with every major class of technology, the area of real-time networking comes with an excessive list of challenges, options, and problems that must be solved prior or during successful network operation. These challenges may include different optimization problems, different protocols or configurations of networking equipment, or even new mechanisms and protocols that must be invented in order to solve a problem. Popular examples include latency models, path finding problems, and placement of critical components. When conducting research in these areas, and when comparing different options to a specific problem in theory, it is vital to select problem scenarios that closely represent the realistic requirements during real-world operation. However, generating such problem scenarios and arguing for their relevance is typically not a primary objective of researchers.

As a result, the availability of well-designed problem scenarios has an influence on the quality of the attained results from many types of problems. Having a common baseline for optimization problems makes their results comparable. In addition, a sophisticated library of problem scenarios can help to enhance the desired algorithms and protocols during their development, and it prevents authors from cherry-picking only those use cases where they reportedly have an advantage. Further, less time and effort must be spent on crafting and validating problem input over and over again.

This abstract takes the first steps towards creating such a library of well established problem scenarios. It aims to serve as a source of information for existing use cases, as well as an effort to abstract their characteristics, predict ongoing future developments, and provide the means to generate new use cases that will likely represent some future requirements. More importantly, it serves as a call for participation for all researchers and network operators who want their use case to be included in that library.

In the following, a brief overview of well known use cases of real-time networks is presented in Section 2, particularly regarding network topologies and traffic requirements. The existing analysis mainly focuses on wide area networks, industrial networks, automotive networks, and data centers. In Section 3, their characteristics are abstracted and a problem scenario generator is proposed. Its core features include more heterogeneous networks, a combination of different topologies in a single network, and a simple framework to generate traffic demands for any topology. Further, Section 4 highlights the need for pre-defined problem profiles. Finally, Section 5 concludes this abstract.

2 Problem Scenarios in Literature

Innovations in next-generation real-time networking will bring about new, highly efficient topologies. However, implementing these innovations takes time, and they are often inspired by existing solutions. This section presents a brief taxonomy of topologies used across different domains.

WAN. Wide Area Networks (WAN) vary in form and size, influenced by both performance considerations and geographical features. The structure of WAN topologies depends on various parameters [1], such as whether it is a research, access, or backbone network. Thus, making general assumptions about WAN networks is challenging. However, datasets with in-use and archived WAN topologies, such as the Internet Topology Zoo [1] and SNDlib [2], provide valuable resources.

Data center. Data center topologies are categorized as Switch-Centric and Server-Centric [3]. Switch-Centric topologies employ intelligent switches for packet routing, while Server-Centric topologies use servers to forward packets. Notable topologies in each category include VL2, Clos Network, FatTree, JellyFish, DCell, BCube, HyperBcube, Flecube, and FiConn. LaScaDa, a novel data center topology, offers superior scalability, average path length, bisection bandwidth, and aggregated bottleneck throughput compared to other topologies [3]. Additionally, reconfigurable data center topologies improve performance by establishing links between frequently communicating racks [4].

Industrial. Industrial automation utilizes star, linear, and clustered line topologies [5]. The star topology has a single controller and multiple I/O nodes connected to a central switch with consistent latencies. In the linear topology, each node connects to two neighbors, with latencies increasing over distance. The clustered linear topology combines characteristics of star and line topologies, with groups of streams exhibiting linearly increasing latencies. Multiple industrial automation use cases, including the significance of ring topologies for switch-over and redundancy, are discussed in [6]. When multiple production cells are merged, interconnected rings form the architecture. The importance of ring, star, and linear topologies for global players can be observed in [7]. Finally, the required flexibility of Industry 4.0 deployments calls for complex and heterogeneous architectures, combining industrial and data center/edge cloud use cases. This, in return, results in the interconnection of industrial linear network segments and (fat) tree topologies.

Automotive. The authors of [8] summarize timing requirements for automotive network traffic. The authors state different algorithms and methods and describe the used topologies, including stars, trees, rings and linear topologies. In [9] the authors present the development of automotive architecture, including rings or multiple rings alongside point-to-point and star topologies. Latency requirements are anticipated to decrease to microseconds.

3 Problem Generation Framework

We propose a problem generator methodology and implementation for generating problem instances in data center, industrial, and automotive scenarios. Our goal is to create a parameterizable platform for synthetic problem generation in different application contexts. We discuss the topology types considered, their configuration parameters, and the proposed generation mechanism. Additionally, we cover stream generation for resource reservation and provide example instances generated with our tool. Please note that the tool is still in development and serves as a foundation for future approaches. We welcome community feedback on relevant topologies and stream profiles for real-time networking problems. Contributions are welcome through our public GitHub repository.[1]

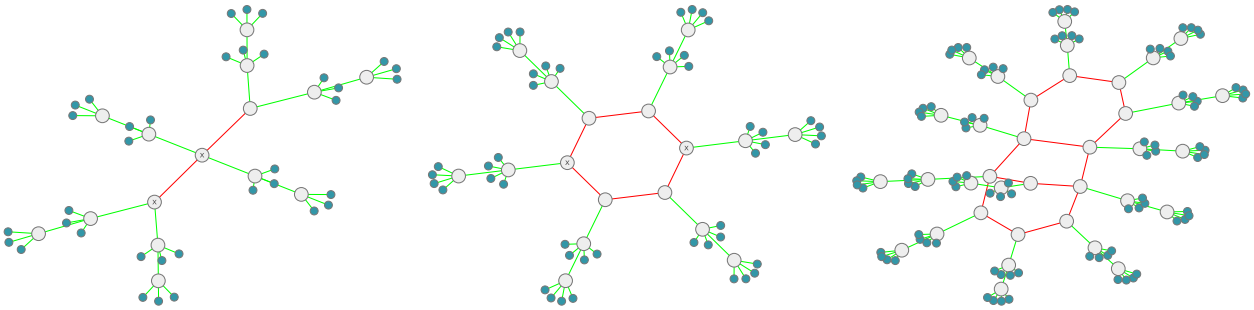
Topology Factories.

Building on the taxonomy mentioned earlier, we focus on three key topology types: linear, rings, and two-layer trees. Furthermore, based on the topologies covered in literature [6], we introduce a mechanism to combine these topologies, forming mixture topologies by integrating two or more individual types.

The initial topologies, linear and ring, are primarily composed of core switches, branch switches, and hosts, drawing inspiration from industrial network layouts [6]. The distinction lies in whether the first and last switch of a linear chain are connected, forming a ring. Figure 1 illustrates an example of each topology type, highlighting red links between core switches and bright green links for branches. Gray represents switches, while dark green represents hosts.

These topologies can be easily generated and parameterized using the following inputs. *main_length* describes the number of core switches, *branches_per_main_switch* and *branch_length* describes the number of branches for each core switch and their length, *hosts_per_branch_switch* defines the number of hosts per branch. Link speeds can be configured using *main_link_speed* and *branch_link_speed*, which determine the available bandwidth for core and branch links. *connect_to_ring* defines if the core should form a ring or a linear chain. Finally, *num_join_points*

[1] https://github.com/lsinfo3/tsn_problem_generator



(a) Example linear topology with three core switches, two branches per core switch with two branch switches each and three hosts per branch switch.

(b) Example ring topology with 6 core switches, one branch per core switch with two branch switches each and four hosts per branch switch.

(c) Example mixture topology consisting of multiple, interconnected rings with linear branches connected to each core switch.

Figure 1: Example topologies of type linear, ring and one mixture topology.

provides the number of attachment points used for the generation of mixture topologies. These attachment points are marked with an X in the figures.

The third topology currently available for generation is a two-layer tree which is inspired by data center topologies such as Fat Trees. Similar to the previous topologies, the two-layer tree consists of core and branch switches as well as hosts and be configured using the parameters *num_layer1_switches*, *num_layer2_switches* and *hosts_per_l2switch*. For the tree topology, all layer 1 switches are automatically considered join points when generating mixture topologies.

Combination of Topologies.

Lastly, we demonstrate the implementation of mixture topologies, combining multiple base topologies into a single architecture. Individual base topologies are defined and interconnected using join points (X in Figure 1). Figure 1c shows an example of a mixture topology with interconnected rings and linear branches attached to each core element. Configuring the joining of topologies involves specifying the maximum number of joins (*maxJoins*) and the behavior of join points (*removeJoinPointsUsed*). This enables the generation of complex scenarios, including redundant interconnects with multiple join points and industrial deployments combining linear, ring, or star topologies with a tree layout for the data center. In the automotive domain, gateways can interconnect application-specific topologies to the core system, such as bus gateways connected to a redundant core ring. Additionally, the framework supports importing empirical topologies, e.g., from the Topology Zoo, for stream generation in subsequent steps.

Stream Factory.

In addition to the generation of the network topology, every problem instance requires a set of streams that represent the traffic flowing within the topology. These are essentially traffic flows between two nodes in the topology, enriched with additional characteristics required for reservation protocols in real-time networks. In general, we generate endpoints through random sampling of all hosts available in the network and select the following additional characteristics based on pre-configured value ranges. *burst_range* defines the minimum and maximum burst size, meaning the amount of data that can be transmitted at one point in time, values are then drawn uniformly between the provided minimum and maximum. *rate_range* similarly defines the minimum and maximum for the mean data rate. *prio_range*, *min_pathlen* and *max_pathlen* determine the range of priorities as well as the minimum and maximum path lengths of this type of stream. Finally, *num_streams* configures the total number of streams of this type. Note that a single set of streams can consist of an arbitrary number of different stream types, each with their own value ranges. Note also that these properties are simply the baseline which we can build upon. The framework is designed to easily extend these characteristics by adding, for example, a required maximum end-to-end latency, inter-arrival time distributions or upper bounds for packet loss probabilities in the future.

4 Pre-configured Problem Profiles

In order to provide a suite of problem instances to the community that can be used to evaluate different mechanisms and approaches in relation to real-time networks, we are looking at defining specific problem profiles that can be used to generate general and reproducible problem instances. To this end, we suggest two example problem profiles and

explicitly invite the community to contribute profiles and configurations from various use cases to include in this suite of problem instances. Two such example profiles are already implemented and can be found in the repository.[2]

In addition, we provide an exemplary set of streams, consisting of six different classes of shorter and longer streams, to showcase the combination of different stream classes. Here, we also specifically invite comments and contributions from the community to establish a suite of use cases and their corresponding types of streams, to be able to synthetically generate realistic problem instances.

5 Conclusion

This abstract highlights the critical importance of a standardized library of problem instances for advancing real-time networking research. In order to address this need, we introduce a brief taxonomy of existing use cases, providing a solid foundation for further exploration. Additionally, we propose a foundational framework capable of generating synthetic problem instances across diverse scenarios, serving as a valuable resource for researchers. We delve into the current state of our framework implementation, outlining its fundamental features and capabilities. Our primary objective is to encourage active participation by the community, by contributing their opinions, experiences, and expertise to establish a comprehensive library of problem scenarios. This library should ideally encompass a wide range of network topologies and stream profiles, accommodating both generic and specific use cases. By fostering the creation of a standardized set of evaluation scenarios, our ultimate goal is to promote comparability among research works and foster the development of innovative approaches in real-time networking research.

References

[1] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011. DOI: 10.1109/JSAC.2011.111002.

[2] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," English, in *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, http://sndlib.zib.de, extended version accepted in Networks, 2009., Apr. 2007. [Online]. Available: http://www.zib.de/orlowski/Paper/OrlowskiPioroTomaszewskiWessaely2007-SNDlib-INOC.pdf.gz.

[3] Z. Chkirbene, R. Hadjidj, S. Foufou, and R. Hamila, "Lascada: A novel scalable topology for data center network," *IEEE/ACM Transactions on Networking*, vol. 28, no. 5, pp. 2051–2064, 2020.

[4] M. Pacut, W. Dai, A. Labbe, K.-T. Foerster, and S. Schmid, "Improved scalability of demand-aware datacenter topologies with minimal route lengths and congestion," *ACM SIGMETRICS Performance Evaluation Review*, vol. 49, no. 3, pp. 35–36, 2022.

[5] J. Diemer, D. Thiele, and R. Ernst, "Formal worst-case timing analysis of ethernet topologies with strict-priority and avb switching," in *7th IEEE International Symposium on Industrial Embedded Systems (SIES'12)*, IEEE, 2012, pp. 1–10.

[6] IEC/IEEE 60802, *TSN Profile for Industrial Automation*, Use Cases v1.3, 2018. [Online]. Available: https://www.ieee802.org/1/files/public/docs2018/60802-industrial-use-cases-0918-v13.pdf.

[7] Huawei, *Smart Oil and Gas All-Optical Network*, Accessed: 2023-06-21. [Online]. Available: https://e.huawei.com/at/solutions/enterprise-transmission-access/intelligent-oil-gas.

[8] S. Tuohy, M. Glavin, C. Hughes, E. Jones, M. Trivedi, and L. Kilmartin, "Intra-vehicle networks: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 534–545, 2014.

[9] Helge Zinner, Julian Brand, Daniel Hopf, *Automotive E/E Architecture evolution and the impact on the network*, IEEE802 Plenary, March 2019, 802.1 TSN. [Online]. Available: https://www.ieee802.org/1/files/public/docs2018/60802-industrial-use-cases-0918-v13.pdf.

Acknowledgments

This work was funded by the Deutsche Forschungsgesellschaft DFG (German Reasearch Agency) under Grant No 316878574, project SDN-App.

[2] https://github.com/lsinfo3/tsn_problem_generator/blob/main/python/z_test.py