Simulating and Emulating the Characteristic Packet Delay of Logical 5G TSN Bridges

Lucas Haug, Frank Dürr, Simon Egger
Institute of Parallel and Distributed Systems
University of Stuttgart
Stuttgart, Germany
{lucas.haug,frank.duerr,simon.egger}@ipvs.uni-stuttgart.de

Lorenz Grohmann
Institute of Parallel and Distributed Systems
University of Stuttgart
Stuttgart, Germany
st161568@stud.uni-stuttgart.de

James Gross, Gourav Prateek Sharma
School of Electrical Engineering & Computer Science
Royal Institute of Technology (KTH)
Stockholm, Sweden
{jamesgr,gpsharma}@kth.se

Joachim Sachs
Ericsson Research
Bremen, Germany
joachim.sachs@ericsson.com

***Abstract*—Due to the demand of many time-sensitive mobile applications, there is currently a strong trend to extend Time-Sensitive Networking (TSN) to the wireless domain. In particular, 3GPP has speficied a logial 5G TSN bridge, which implements the same functions as wireline TSN bridges (e.g. the Time-Aware Shaper of IEEE 802.1Qbv). However, a logical bridge has fundamentally different port-to-port delay properties than its wired counterpart. In this paper, we present two open-source tools and open latency data sets for simulating and emulating the characteristic port-to-port delay of logical 5G TSN bridges.**

***Index Terms*—simulation, emulation, time-sensitive networking, mobile communication, 5G, delay**

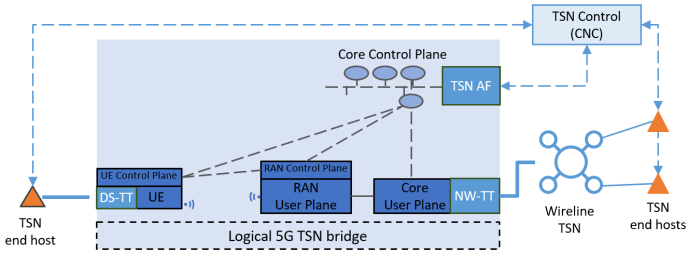


Fig. 1: Logical 5G TSN bridge

I. Introduction

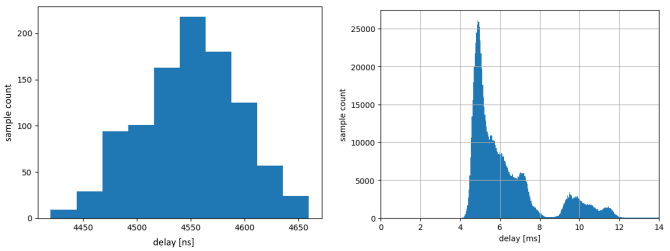
Many safety-critical cyber-physical systems rely on the timely delivery of data over a real-time communication network. Moreover, many of these systems include mobile devices and, therefore, require wireless real-time communication. Some examples of use cases include the coordination of the movement of automated guided vehicles on a factory shop floor, or smart farming, with drones monitoring the area in front of harvesters to ensure the safety of animals in the field (see [1] for a description of these and other use cases).

The requirement for real-time communication with strict bounds on end-to-end packet delay (PD) and packet delay variation (PDV) has been well recognized by major standardization bodies. IEEE has specified standards under the umbrella term *Time-Sensitive Networking* to support "deterministic" real-time communication over standard Ethernet. This includes different so-called shapers to schedule traffic at TSN bridges along the end-to-end path between talker (sender) and listener (receiver). In particular, we will consider the so-called Time-Aware Shaper (TAS) in this paper, which uses a time-driven gating mechanism at each TSN bridge to control when egress queues for different traffic priorities forward packets.

TSN has been extended from the wired domain (Ethernet) to wireless domain (mobile 5G networks) by 3GPP by specifying the so-called *logical TSN bridge* depicted in Figure 1. A logical (wireless) TSN bridge implements the same functions as its wired counterpart. In particular, the gating mechanism of the TAS is implemented by the TSN Translators (TT). The mobile User Equipment (UE) is connected to the bridge at the Device-Side TT (DS-TT); the Network-TT (NW-TT) connects to wired TSN bridges. Note that the wireless link between UE and base station (gNB) is inside the bridge, i.e., internally connecting the ports of the logical bridge. This has fundamental consequences on the non-functional properties of the logical bridge, specifically, the port-to-port delay (also called bridge delay in the standard).

We have measured and compared the port-to-port delay of wired TSN bridges and wireless logical bridges using 5G technology (commodity of the shelf 5G equipment and Open-RAN-based hardware). To measure delays in the 5G system, PTP-based time synchronization was implemented over an out-of-band wired network, providing a time reference for all nodes. The 5G network operated in the n78 band at 3.5 GHz, utilizing TDD mode with 40 MHz bandwidth, 30 kHz subcarrier spacing and 106 physical resource blocks. All occuring errors were corrected by 5G retransmission mechanisms (HARQ)[1].

[1]We refer the interested reader to [2] for a more detailed latency analysis.



(a) Wired bridge (b) Logical 5G bridge

Fig. 2: Port-to-port delay distributions

Figure 2 shows fundamental differences: (1) the port-to-port delay of a logical bridge is orders of magnitude greater (milliseconds vs. microseconds); (2) the port-to-port delay of a logical bridge follows a heavy-tailed bi-modal distribution. 5G's time-slotted architecture and HARQ explain the multi-peak structure. Moreover, for a heavy-tailed distribution, the probability of large values does not decrease exponentially, in contrast to the exponential latency distribution of the wired TSN bridge.

This fundamental difference in internal bridge timing raises many questions like: (1) What is the impact of the stochastic port-to-port onto time-driven scheduling (TAS), which relies on precise timing? Will existing gate schedules calculated for wired bridges still work? Note that the TAS only controls the queuing delay. The port-to-port delay is a parameter to the algorithms for calculating schedules. Therefore, it is expected that port-to-port delay should have an impact either on the safety-guarantees of non-robust schedules or the efficiency of robust schedules. (2) What is the impact onto the performance of applications communicating through a logical bridge? Take as an example a networked control system with sensors and actuators at the mobile devices and a controller on an edge cloud server, i.e., closing the control loop over a logical 5G TSN bridge. Delay is expected to have an impact onto the stability and performance of the control system.

To answer these and other questions, we present an overview of two tools that support the validation and evaluation of systems including logical TSN bridges: (1) *6GDetCom Simulator*: We added an implementation of a logical bridge to the popular OMNeT++/INET network simulation framework. This simulation model is focused on the realistic simulation of port-to-port delay by integrating our data from measurements in real 5G networks. Since the design of the logical bridge inherits all TSN functions like the TAS from the existing TSN bridge model, we can test the impact of port-to-port delay onto existing TAS scheduling algorithms and new wireless-friendly scheduling algorithms. (2) *6GDetCom Emulator*: The network delay emulator is a Linux tool—implemented as Queuing Discipline (QDisc)—to emulated the characteristic network delay of a 5G network from measurement data. This allows for testing the impact of the characteristic delay onto real applications under test.

Both, the simulator and the emulator are open source [3], [4]. Moreover, we also provide our latency measurements from 5G networks as open data [5].

The remainder of this paper is structured as follows: In Section II, we present an overview of the Det6G Simulator. In Section III, we present an overview of the 6GDetCom Emulator. Finally, we conclude this paper in Section IV.

II. 6GDetCom Simulator

In order to simulate the characteristic delay of a logical TSN bridge, we implemented a so-called DetCom node for the OMNeT++/INET framework. The DetCom contains one NW-TT and multiple DS-TTs which inherit all TSN functions from the TSN bridge model of INET, e.g., its TAS implementation.

To simulate the characteristic delay, there are two basic design options: (1) Integrating a sophisticated simulation model of the 5G system, including anything that might introduce delay in the data path, such as HARQ, physical layer simulation models of the wireless medium, etc. We decided against this option due to the high complexity of implementing a realistic 5G simulation model. (2) A data-driven approach, treating the logical-bridge-internal 5G system as a "black box" that delays packets passing through the 5G system based on realistic latency models from measurements in real 5G networks. We implemented this data-driven option using latency histograms from our measurements in real 5G networks [5].

To this end, the DetCom node is extended by a so-called Delayer component, which is located before the egress queues of the TSN bridge. The Delayer is either configured using latency histograms or delay traces from measurements. Different configurations can be applied in upstream and downstream direction or, in general, between different port pairs of the logical 5G TSN bridge (DetCom node). Note that a packet from one UE to another UE passes the wireless link to/from the gNB two times, whereas a packet from/to the NW-TT only traversed the internal wireless link once.

While our design allows the replay of delay traces to simulate delay correlations between different links within the DetCom node, one drawback remains: other configuration parameters in the simulation, such as packet size and sending rate, do not influence the resulting DetCom delays.

Figure 3 shows that the simulated delay of packets passing through a DetCom node closely follows the original input histogram from the real network.

III. 6GDetCom Emulator

Figure 4 shows the architecture of the 6GDetCom network delay emulator for Linux. It consists of two major parts: (1) A custom Linux QDisc in kernel space emulating the delay on the network data path between ingress and egress network interface. (2) A user-space application generating the delays for the QDisc.

The QDisc is attached to the data path at the egress port. For each packet to be forwarded through the port, it reads a delay value from a FIFO queue (delay queue). This queue is populated with delay values in advance to ensure that packets can be processed at high speed. Delayed packets are stored

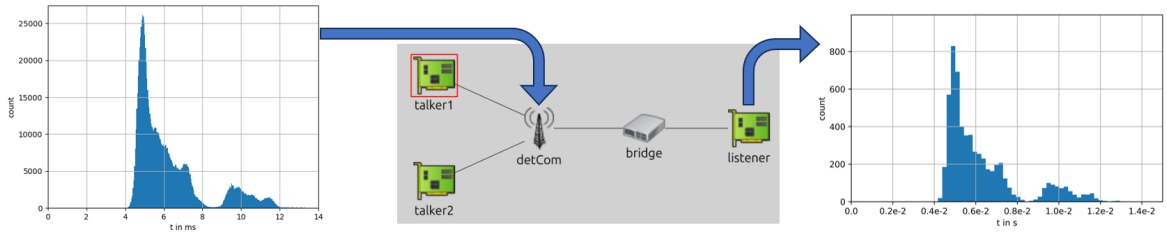


Fig. 3: DetCom node applying delay to packets

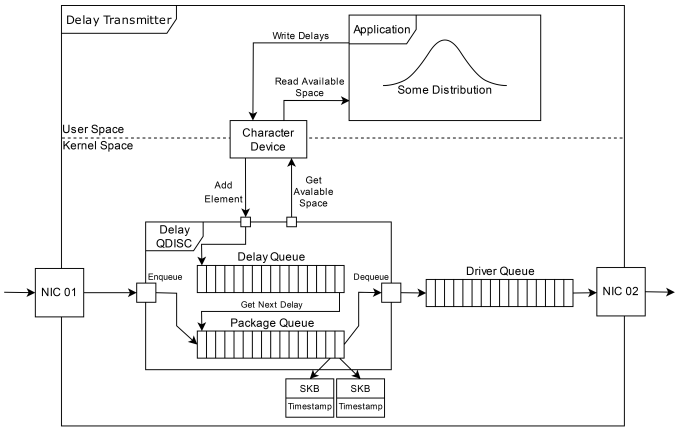


Fig. 4: Det6G network delay emulator architecture

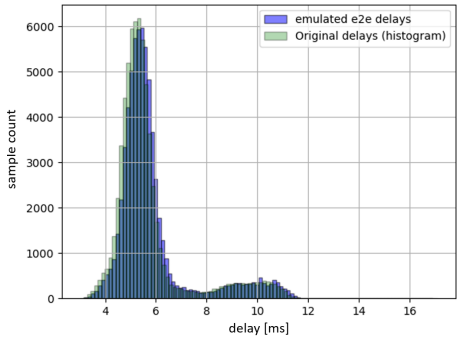


Fig. 5: Emulated vs. original delay

in the packet queue until a timer signals the release of the delayed packet towards the network driver.

The user-space application pushes delay values via a character device to the delay queue, depending on the fill level of this queue. The application can use any algorithm or data set to create delay values. In particular, we can use our delay histograms based on latency measurements in real 5G network [5].

This QDisc-based design can be integrated easily with other Linux features such as virtual bridges to emulate a whole network on a single Linux host, or other QDiscs to provide different delays to different traffic classes using QDisc filters and dedicated delay QDiscs per class.

A drawback of this design is its limitation to independent identically distributed (i.i.d.) latency values since delays are pre-calculated for performance reasons.

Our evaluations show that the emulated delay of a real 5G distribution closely follows the given distribution from measurements in a real 5G network with an offset of about 81 µs. Considering the very short delays of wired bridges in the range of few microseconds (cf. Figure 5), we observe another limitation: we cannot emulate the delay of wired bridges with this software-based approach. However, wireless logical bridges with delay values in the range of milliseconds pose no problem.

Further information how to use the network emulator can be found in our software repository [4].

IV. Conclusion and Future Work

We have presented two tools for simulating and emulating the characteristic packet (port-to-port) delay of a logical 5G TSN bridge.

Next steps include the application of these tools to the validation of our algorithms for calculating wireless-friendly TAS schedules that are robust to high packet delay variation. Moreover, we will use the emulator to evaluate the impact of characteristic delay onto applications, such as the quality of control of a remotely controlled exo-skeleton (networked control system).

Acknowledgments

This work was supported by the European Union's Horizon Europe project DETERMINISTIC6G under grant agreement No. 101096504.

References

[1] D. Patel, E. M. de Oca, H. N. Nguyen, J. Costa-Requena, J. Gross, G. P. Sharma, L. Grosjean, J. Sachs, J. Harmatos, O. Höftberger, F. Profelt, D. Puffer, D. Houatra, F. D. Simio, G. Bigoni, F. Giovacchini, and G. Bag, "DETERMINISTIC6G use cases and architecture principles – Deliverable D1.1 of DETERMINISTIC6G project," https://deterministic6g.eu/images/deliverables/DETERMINISTIC6G-D1.1-v1.0.pdf, Jun. 2021.

[2] S. Mostafavi, M. Tillner, G. P. Sharma, and J. Gross, "EDAF: An end-to-end delay analytics framework for 5G-and-beyond networks," in *Proceedings of IEEE INFOCOM 2024 – 11th International Workshop on Computer and Networking Experimental Research using Testbeds (CNERT 2024)*, Vancouver, Canada, May 2024, DOI: 10.1109/INFOCOMWKSHPS61880.2024.10620853.

[3] "6GDetCom simulator," https://github.com/DETERMINISTIC6G/6GDetCom_Simulator, last accessed March 2025.

[4] "6GDetCom emulator," https://github.com/DETERMINISTIC6G/6GDetCom_Emulator, last accessed March 2025.

[5] "Deterministic6G measurement data," https://github.com/DETERMINISTIC6G/deterministic6g_data, last accessed March 2025.