PerfVis: Visualization of Timings in Network Transmission

Marlene Böhmer
Saarland Informatics Campus
Saarland University
Saarbrücken, Germany
boehmer@cs.uni-saarland.de

Thorsten Herfet
Saarland Informatics Campus
Saarland University
Saarbrücken, Germany
herfet@cs.uni-saarland.de

***Abstract*—Analysing and optimizing the networking performance of protocol stacks requires accessible tools capable of presenting precise measurements in intuitive visualizations. Real-time presentation of live data further increases the efficiency in gaining insights, e.g. into cross-stack phenomena and configuration changes of highly configurable networking technologies such as 5G or 6G. In this context, we introduce PerfVis, a tool designed to visualize network transmission timings. By one-way measurements, the insights can be focused on direction-dependent effects of single links or short paths. Case studies involving a 5G campus network demonstrate how PerfVis can identify timing effects on lower layers of the protocol stack and highlight its capability to adapt measurements and visualizations for specific analysis tasks.**

***Index Terms*—Network communications, Performance evaluation, Visualization, Real time, 5G**

I. Introduction

Emerging networking technologies like 5G, 6G or Open Ran are evolving to open and highly configurable network functions and setups. This brings the benefit of customizable networks but requires the tools and knowledge to optimize configurations. While round trip times and delays provide important information, they shed only limited light on cross-layer and absolute timing effects. Understanding those is important when optimizing, monitoring or troubleshooting high-performance networks and low-latency or real-time applications. Additionally, detailed time and delay information can help to find good configurations in highly configurable wireless networks like 5G.

A common approach to analysing lower layers of the protocol stack is measuring various parameters and visualizing them in an accessible way, e.g. by eye diagrams or spectrograms. On higher layers, it is rather common to do statistical measurements, e.g. of throughput or latency, and visualize the statistical properties, e.g. by distributions. Both approaches are important to highlight the characteristics of their metrics, but timing is a characteristic that spans and is influenced by all layers. Thus we suggest complementing measurement methodologies with timing measurements from various layers and accessible visualizations to ease gaining insights. Depending on what kind of timestamps are available, a visualization can give an application layer perspective or allow effects to be narrowed down to specific layers. Visualizations also need to span multiple orders of magnitude as the order of magnitude and the change rate of timing effects highly differ.

With this vision in mind, we developed PerfVis[1] as a tool to visualize the timing of packet transmissions and to get insight into effects that distort timing during network transmissions. The main contribution of PerfVis is the visualization of timestamps from a network transmission measurement. With a live mode, PerfVis can provide insights in real-time while also supporting downstream analysis by saving measurement data. Timestamps taken on the application layer include effects from the full network protocol stack, thus they give an application layer view. Timestamps from lower layers provide a different view and enhance analysis. With its visualization, PerfVis complements other tools and methodologies by providing a quick visual insight into transmission timing effects at any layer and can even be used to investigate effects at lower layers when access to lower-layer logs is not available.

II. Related Work

Different tools exist for measuring network latency, employing different methodologies. Traditional tools such as Ping and TWAMP [1] measure round-trip time (RTT), while OWAMP [2] focuses on one-way latency measurements. Fabini et al. [3] highlight that end-to-end delay measurements containing subsequent time-slotted links need special measurement methodologies to give representative statistical results. In a simple case, this already applies to an RTT measurement of a 5G link. We do not introduce the suggested start-time randomness in the test setup but choose one-way measurements to not obscure effects occurring in one direction with effects on the return path. A side effect of one-way measurements is that synchronization plays a major role in how timestamps drawn from different clock sources relate. Grigorjew et al. [4] show that the source of the timestamps, i.e. whether the source is the system time or an external PTP (Precision Time Protocol) hardware clock, has—apart from their different semantics—also influence on the accuracy of the timestamps, which is important especially when investigating low latencies.

[1] https://git.nt.uni-saarland.de/research-projects/independent/performance-visualization/-/tree/enmeth25?ref_type=tags

Furthermore, increasing the number of measurement points on a path allows for more fine-grained analysis. In the same philosophy, Orosz et al. [5] deal with multi-layer timestamping including software and hardware timestamps with the help of an FPGA. Reif et al. [6] present a mechanism of fine-grained timing analysis within a network layer by gaining timestamps from the instrumentation of the protocol code. All those timestamps from different clock sources (system or PTP hardware clocks) and different levels (application layer, transport layer, ...) are valid for our visualization but come with different efforts to obtain them.

Research has shown that a fitting visualization can enhance the understanding of complex data and make it easier for users to learn patterns, trends, and anomalies [7]. Reduced waiting time from collecting the data to the visualization being available plays a significant role in how efficient it is to work with the data [8]. Thus we provide our tool with a live mode. Standard tools for measuring network latency provide valuable statistics but lack an intuitive visual presentation of the results. When results are visualized, it is usually a simple line or bar graph that has time on the abscissa and latency or RTT on the ordinate. Gregg et al. [9] propose heat maps as a powerful method for visualizing latency distributions over time to highlight patterns that simple line graphs might miss, such as the clustering of latency spikes or the impact of specific events on overall performance. Di Bartolomeo et al. [10] compare how different timeline layouts with recurrent events affect readability. A slightly modified timeline layout is the basis for our visualization of timestamps.

III. Tool Design and Implementation

PerfVis is designed to provide intuitive visual insight into the timing of network transmissions including judging delays and jitter and identifying timing structures. It is flexible in the protocol stack and traffic pattern to adapt to different analysis scenarios and further aims to support different usage scenarios by variable setup and viewing modes. Before diving into the implementation and details of PerfVis, the following two subsections introduce the visualization and its underlying data.

A. Timestamps

The visualizations are created from timestamps. For a complete view of the timing of packets traversing a network, timestamps from both transmission ends are required. Timestamps from multiple sources give insight into the effects on specific network protocol layers or sections of network transmission. To be able to focus on single links and effects that only occur in one direction, one-way measurements are used. PerfVis uses built-in measurements but timestamps from external sources or tools can be added as long as all timestamps are consistent, so computing differences between them is valid. If this is not the case for all timestamps, sets of timestamps that fulfil the consistency requirement can be grouped for separate visualization. As timestamps might originate from different clocks that are not well synchronized, the question of time reference for visualization arises. The sender-side clock of the highest layer timestamps is chosen as a reference because the first timestamp of each packet transmission originates from that clock.

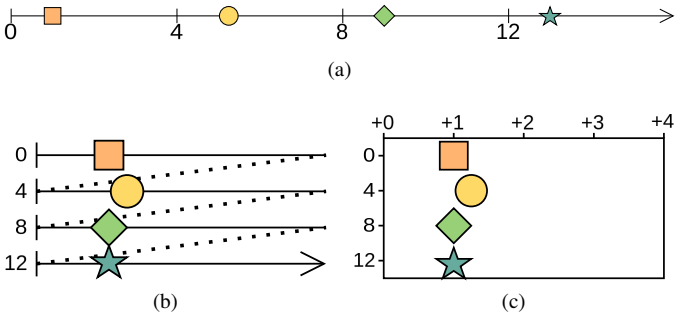


Fig. 1. Presentation of a one-dimensional timeline (a) in a two-dimensional plot (c) by splitting the timeline into parts (b). This highlights periodic events and deviations from the periodicity.

Due to different clocks being involved in network transmission, clock effects and clock drifts will be part of the visualization. To isolate timing effects occurring during network transmission from the clock effects of the sender and receiver clocks, these two clocks should be synchronized. Furthermore, there are clocks in other parts of the transmission system, e.g. the 5G RAN, that influence the transmission timing and the effects of those clocks will show in visualizations.

B. Visualization Idea

As timing effects have completely different orders of magnitude and change rates, the visualization has to be capable of depicting multiple orders of magnitude and conveying the notion of time. Video is a suitable format for that as it inherently includes a notion of time and allows grasping timing effects in different orders of magnitude, i.e. small-scale effects within the presentation of a video frame and effects of a larger scale over the frames of a video. The following explanation assumes that a data set consisting of the send and receive timestamps for each measurement packet is available and assumes that the measurement packets were sent periodically with interval i.

The key idea of the visualization is to sort all timestamps on a timeline (Fig. 1a), split the timeline into parts (Fig. 1b) and arrange subsequent parts as rows into images (Fig. 1c). Subsequent images then yield an animation or video. The dimensions of the image are characterized by the time chosen to be in a row and the number of rows put into one image n_{rows}. To support the periodicity of the measurements it makes sense to choose a row as a multiple of the interval i, such that the time of a row is $t_{row} = ipr \cdot i$, where ipr is the number of intervals per row. The interval i can be chosen freely, the reasoning of choice could be to match application behaviour, to match the periodicity of some timing effect or to produce visualizations that are easy to interpret, e.g. when i is chosen greater than the delay, all timestamps belonging to one packet transmission will fall into the same row.

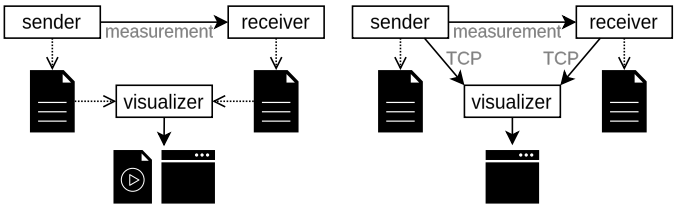


Fig. 2. (a) In intermitted mode, sender and receiver save their measurement data to a file, afterwards the visualizer reads the files and saves a video or shows an animation. (b) In live mode, sender and receiver send the measurement data directly to the visualizer which views an animation.

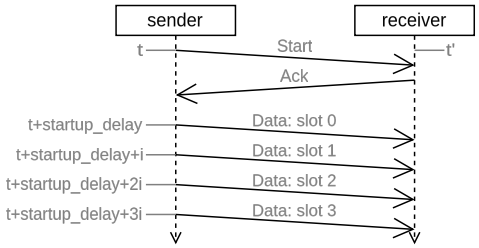


Fig. 3. Packet sequence of a PerfVis measurement.

Visualizations created this way can give quick and intuitive insights into timing effects of different kinds and orders of magnitude. In the case of a continuous, periodic packet transmission without effects that differ between the packets or change over time, vertical lines are expected that stay in their position over the images of a video. Deviations thereof can easily be visually identified. Transmission timing effects that change over time show up as shifts between rows and between frames in the visualization. By choosing the time of a row, also the order of magnitude of effects seen between rows is defined. Equally, by choosing the number of rows, the order of magnitude of effects between video frames is fixed. For example, when a visualization is configured with 1000 rows of $1\,\mathrm{ms}$ and a shift of $1\,\mu\mathrm{s}$ per video frames is identified, then this change is 6 orders of magnitude smaller than the time over which it changes but the shift is recognizable even in the presence of static jitter with a higher order of magnitude.

C. Implementation

PerfVis is implemented in Python and consists of multiple modules supporting different modes. The modes include an intermitted mode (Fig. 2a) and a live mode (Fig. 2b). The sender, receiver and visualizer modules are described in the following.

1) Sender and Receiver: These two modules send measurement packets from sender to receiver and collect timestamps whenever a packet is sent or received on their side independent of each other. The current state of the implementation is that timestamps from two levels are collected automatically, firstly application layer timestamps taken in the Python code called Python timestamps and secondly kernel software timestamps.[2] The kernel timestamps are enabled by setting the software timestamping flags in the socket options of sockets used for measurement. When sending or receiving packets via these sockets, the kernel timestamps can be read from ancillary data and give information about when a packet left or arrived in the kernel. Depending on the mode, the sender and receiver store data to file or, in the case of live mode, additionally send it via TCP to the visualizer. Sending to the visualizer is done from separate processes to not disturb the timing of sending or receiving measurement traffic.

[2]https://www.kernel.org/doc/html/next/networking/timestamping.html

At the beginning, a handshake makes sure that the connection itself is functional and provides information about periodicity and sender configuration to the receiver. This allows for better timeout settings at the receiver without manual configuration via the command line. After a startup delay that is meant for the handshake to complete, the sender periodically sends the measurement packets. Their relevant content is only a slot number that specifies how many interval times after starting the measurement the packet is supposed to be sent (target send time) and a sequence number to identify the packets and with those the sender and receiver side timestamps that belong to the same packet (Fig. 3). Optionally measurement packets can be padded for increased size. The sender tries to send the packets close to the target timestamp specified by the slot. The Python timestamps will not perfectly hit the target due to scheduling of the operating system, but as this information is included it can be part of later analysis.

2) Visualizer: The core task of the visualizer module is to use the data from the sender and receiver to generate animations. To achieve this, the visualizer converts all timestamps into x and y coordinates in images according to the visualization as presented in Fig. 1c. Internally, an animated matplotlib scatter plot enables showing the animation or saving a video with a frame rate to view data in real-time. This frame rate depends on the choice of the parameters i, ipr and n_{rows} which define how much time is represented in one image. The real-time frame rate $r = 1/(t_{row} \cdot n_{rows})$ shows each image as long as the time that is represented in that image. This might lead to high or low frame rates that are not comfortable to view but they ensure awareness of the actual timing and can be adjusted to a comfortable speed in a video player of choice. For live animations, when data processing and rendering of the visualization happen in parallel, the effective frame rate might not be definable as the real-time frame rate r. Depending on the tools used, the effective frame rate will be limited or even vary. To still provide real-time visualizations, the PerfVis live mode adapts to the effective frame rate by repeating or skipping images when necessary and reporting the effective and real-time frame rates. For cases of live animation where the real-time frame rate is low and the effective frame rate is much higher, PerfVis provides an *incremental* option that updates the image partly with the most recent data but this can lead to visual breaks of continuously changing effects at the point up to which the data is updated.

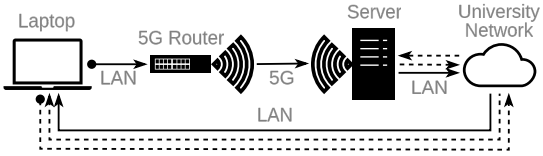


Fig. 4. Setup of a small cell 5G Campus Network connected to the wired university network. Two paths have been measured: the LAN path (⇢) from the laptop via the university network to the server and the same way back and the 5G path (→) from the laptop via the 5G router to the server and back via the university network.

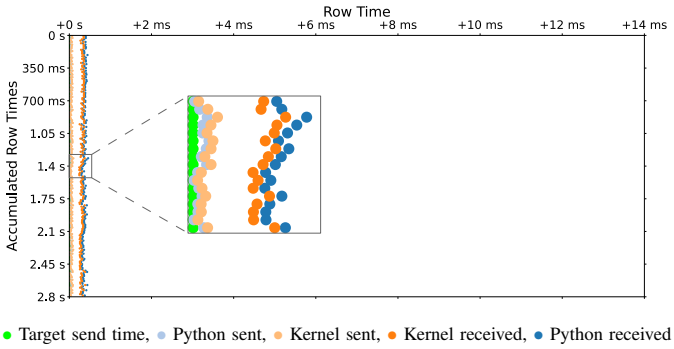


Fig. 5. PerfVis visualization showing the send and receive timestamps of a measurement over the LAN path (Fig. 4). The visualization is constructed as described in Fig. 1.

	Ping RTT				TWAMP RTT			
Format	avg	min	max	mdev	avg	min	max	jitter
LAN	1.015	0.689	184.294	4.930	0.998	0.809	1.617	0.064
5G	7.759	5.005	61.487	2.545	7.697	4.634	22.790	2.655

	TWAMP FWD			TWAMP RTD		
Format	avg	min	max	avg	min	max
LAN	0.396	0.288	1.012	0.599	0.506	0.822
5G	4.421	1.592	16.729	3.273	2.819	12.258

Table 1. Ping and TWAMP results for the LAN and 5G paths (Fig. 4) also evaluated with PerfVis. The TWAMP result can be split up into the forward delay (FWD) and return delay (RTD).

IV. Evaluation

In the case of small network setups with only a few hops, the influence of the timing effect of single links on the overall timing is large, especially the influence of wireless links in an otherwise wired network. This allows the analysis of a wireless link, e.g. of a 5G link, from an end-to-end measurement collecting timestamps. Our tool can help to analyse its influence and get insight into the structure of timing effects that statistical performance metrics hide. The following evaluations give examples of PerfVis usage and visualization capabilities.[3] Even though the evaluations focus on 5G, PerfVis is not limited or tight to that and could equally be used to analyze 4G, Wifi or any other network connection. However, the effects seen in the PerfVis visualizations of other use cases might be different of course.

A. 5G Campus Network Slot Structure

The following evaluations are performed to analyse the network transmission over a 5G campus network that is integrated into the university's network infrastructure (Fig. 4). PerfVis sender and receiver have been executed on a laptop which connected the measurement path to a loop with the help of network namespaces. As—in this very special setup—they draw their timestamps from the same clock, a perfect synchronization between them is achieved. PerfVis has in all cases been configured to transmit 5000 UDP (User Datagram Protocol) measurement packets, save sender and receiver side timestamps to file and save the animation to a video file with video frames containing images with 200 rows. It should be noted, that the interval between PerfVis packets has been chosen to be 14 ms after an initial test and estimation of delays, as this time allows all receive times to be placed into the same row as the send times. The images of PerfVis results show just one frame from the video that the visualization creates.

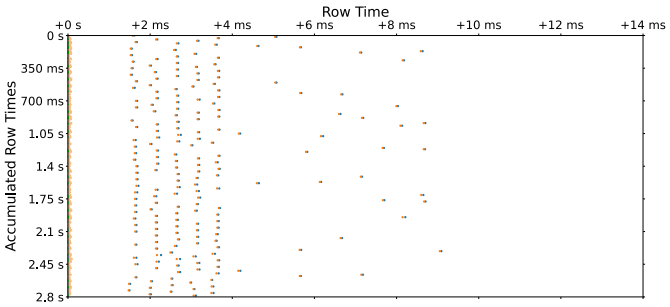
As timing effects that the 5G network introduces should be investigated, first, a baseline from the wired university network is generated (Fig. 4). This is done by setting a route that forces the traffic from the laptop over the university network to the server and back over the university network to the laptop. The result in Fig. 5 shows that the delay is below 0.5 ms and shows little jitter. Ping and TWAMP[4] commands have been executed using the same path, configured with 5000 packets and the same packet interval of 14 ms (Table 1). When considering that Ping and TWAMP measure the RTT, while PerfVis only includes one way, their average RTT around 1 ms support the same order of magnitude in delay and little jitter.

The next test spans the full loop starting and ending at the laptop and including the 5G network in the uplink direction. The PerfVis visualization (Fig. 6a) clearly shows that the timings have a structure that was not seen in Fig. 5. Before investigating the structure seen in the PerfVis output, it is worth comparing it to the Ping and TWAMP results (Table 1). As the RTTs include both ways they are not expected to include such a clear structure. However, the TWAMP implementation outputs forward and backward delays, similar to an OWAMP test, which can be expected to show a closer similarity with the PerfVis timings. Yet, the TWAMP forward delay covering the same path as the PerfVis test does not show a similarly recognizable structure (Fig. 7).

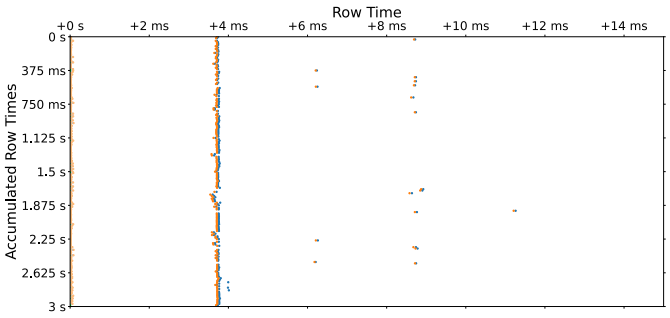
A plausible hypothesis for the structure seen in the PerfVis output is the 5G frame structure, which is shortly explained in the following. The 5G time domain structure consists of frames that are 10 ms in length and, as our 5G campus network operates with 30 kHz subcarrier spacing, each frame contains 20 slots of 0.5 ms [11]. Furthermore, our 5G Campus Network operates in TDD (Time Division Duplex) which means that the slots are assigned to be uplink, downlink or again contain some portion of uplink and downlink symbols. The slot assignment in our 5G campus network iterates over the following slot

[3]Code snippets, data and visualization videos can be found in the project repository.

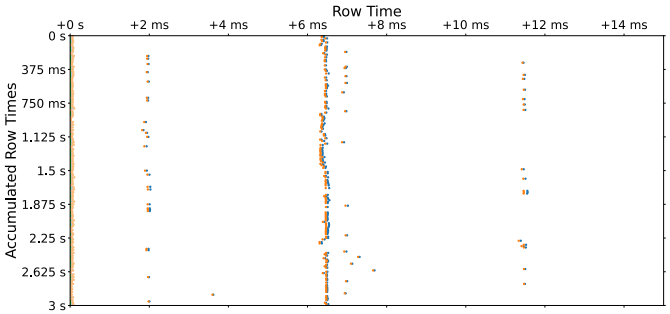
[4]TWAMP implementation used: https://github.com/emirica/twamp-protocol



(a)



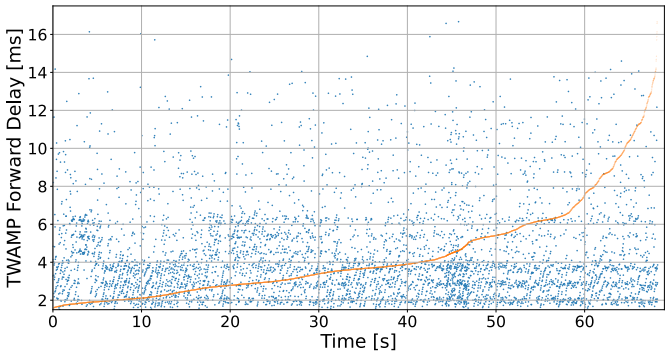
(b)



(c)

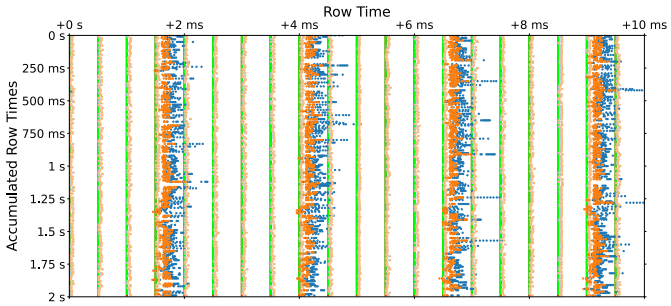
● Target send time, ● Python sent, ● Kernel sent, ● Kernel received, ● Python received

Fig. 6. PerfVis visualization of measurements over the 5G path (Fig. 4). (a) is configured with a packet interval of 14 ms, (b) with a packet interval of 15 ms and (c) with a packet interval of 15 ms and a changed 5G slot pattern.



● delays in measurement order, ● delays sorted by value

Fig. 7. Forward delay for a TWAMP measurement over the 5G path (Fig. 4). Structures like the vertical lines seen in Fig. 6a should show as horizontal lines in this plot.



● Target send time, ● Python sent, ● Kernel sent, ● Kernel received, ● Python received

Fig. 8. PerfVis visualization of measurements over the 5G path (Fig. 4) configured with a packet interval of 0.5 ms and showing the length of a 5G frame in the rows to highlight the 5G frame structure.

assignment 'DDDSU', where 'D' are downlink slots, 'U' are uplink slots and 'S' are special slots that contain 10 downlink, 2 guard and 2 uplink symbols. This slot assignment leads to 5G frames containing this slot structure 4 times and results in an uplink possibility every 2.5 ms.

To better identify a periodicity of 2.5 ms, the PerfVis evaluation has been repeated with an interval of 15 ms and the result indeed shows that except for a few artefacts the timestamps are closely lying on lines that are spaced by 2.5 ms (Fig. 6b). For verification of the hypothesis, the slot assignment has been changed to 'DDDDDDDSUU' which repeats two times in a frame and thus uplink is possible every 5 ms. The PerfVis result in Fig. 6c shows nicely not only the 5 ms periodicity but also the two uplink slots following each other. This confirms the hypothesis, that the structure in the PerfVis visualization originates from the 5G time domain structure.

In this example of the 5G time domain structure, it can be shown that the PerfVis measurement and visualization can be tweaked to deliberately highlight specific structures (Fig. 8). As we know from our 5G configuration that the slots are 0.5 ms long, the PerfVis packet interval is chosen as 0.5 ms to identify the number of available uplink slots. This can then be visualized with 20 intervals in a row, thus a row contains the 5G frame length of 10 ms. The result of that configuration clearly shows a slot format with uplink slots available every 2.5 ms, where the reception of transmitted packets accumulates.

B. 5G HARQ Retransmissions

On a different setup whose software gives out more details about the 5G transmissions, we could verify that PerfVis can visualize 5G HARQ (Hybrid Automatic Repeat Request) retransmissions. The hardware setup can be seen in Fig. 9. The Open Air Interface (OAI) software suite enables 5G transmissions via the USRP B210 SDRs (Software-Defined Radios) by a PC running the OAI 5G Core and a stand-alone 5G gNB and a mini-PC running a 5G stand-alone UE. The PerfVis measurement was configured with a packet interval of 15 ms and an increased packet size of 1500 B additional padding. Fig. 10 shows two lines on the left of

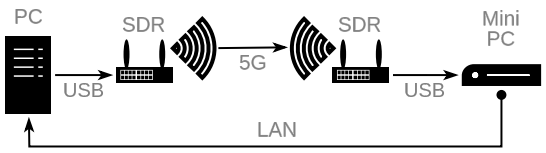


Fig. 9. Setup that uses software-defined radios (SDR) controlled by an open-source 5G software running on the connected PCs.

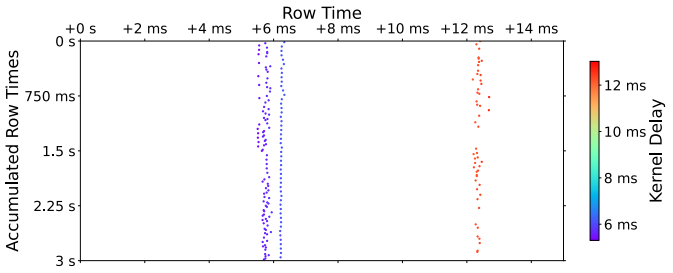


Fig. 10. PerfVis visualization of a measurement with transmission from the mini-PC via LAN to the PC and then via 5G back to the mini-PC. The visualization only shows the kernel receive timestamps coloured with the kernel delay, computed as the difference between the kernel send timestamps and the kernel receive timestamps.

two subsequent downlink slots and outliers on the right which are receptions after a HARQ retransmission. We know that as 92 out of 500 measurement packets had a delay higher than 8 ms and the OAI software output reported 105 transmissions that required exactly one HARQ retransmission until the end of the PerfVis measurement. That the OAI software states a few more HARQ retransmissions, can be explained by the handshake and packets not stemming from PerfVis that required retransmissions. This example shows that the PerfVis visualization based on kernel software timestamps can indicate the occurrence of retransmissions in lower protocol layers. PerfVis is not bound to the specific setup, but the OAI software provides the lower-layer logs with which we could verify that the visualization indeed shows HARQ retransmissions.

V. Discussion and Future Work

The capabilities of PerfVis do not extend to identifying sources or causes of timing effects on its own. The interpretation of the visualization has to be done with all layers in mind. An empirical series of tests in a structured way can help to identify and verify the causes of timing effects. To support this, taking timestamps from other layers than the current application layer Python timestamps and the kernel timestamps will be explored. If possible, PerfVis should take these additional timestamps automatically, otherwise, ways of obtaining and integrating them in the final visualization have to be documented and implemented. If timestamp accuracy becomes an issue, different ways of traffic generation can be explored. Planned improvements for usability are adding a GUI and providing more preconfigured plots and evaluations for choosing a suitable visualization for the analysis task. Furthermore, PerfVis is supposed to be used to investigate the influence of 5G settings and different protocol stacks on the application layer timing of packet transmission.

VI. Conclusion

Investigating network transmission timings is a complex task that can, depending on the objective, require very different approaches. PerfVis highlights that visualization is a crucial part of a tool that makes measurement data more accessible. By visualizing timestamps rather than only delays of packet transmissions PerfVis can give detailed insights while still being easy to use as an application. With the live mode and extensibility to different protocol stacks and traffic patterns, it can be used for latency analysis as well as network optimization. The usage examples analysing 5G links show that the tool helps to identify timing effects introduced by lower layers in the protocol stack, even with timestamps from the higher layers and that measurement and visualization can be adapted to work out specific effect characteristics.

Acknowledgment

We thank our students Julius Herrmann for realizing the tests on the Open Air Interface setup and Moritz Miodek for plotting and analysing the Ping and TWAMP tests.

References

[1] K. Hedayat, R. Krzanowski, A. Morton, K. Yum, and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)," Internet Requests for Comments, RFC Editor, RFC 5357, Oct. 2008. [Online]. Available: www.rfc-editor.org/rfc/rfc5357.txt

[2] S. Shalunov, B. Teitelbaum, A. Karp, J. Boote, and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)," Internet Requests for Comments, RFC Editor, RFC 4656, Sep. 2006. [Online]. Available: www.rfc-editor.org/rfc/rfc4656.txt

[3] J. Fabini and M. Abmayer, "Delay measurement methodology revisited: Time-slotted randomness cancellation," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 10, Oct. 2013.

[4] A. Grigorjew, P. Diederich, T. Hoßfeld, and W. Kellerer, "Affordable measurement setups for networking device latency with sub-microsecond accuracy," *Würzburg Workshop on Next-Generation Communication Networks (WueWoWas'22)*, 2022.

[5] P. Orosz, T. Skopko, and J. Imrek, "A netfpga-based network monitoring system with multi-layer timestamping: Rnetprobe," in *2012 15th International Telecommunications Network Strategy and Planning Symposium (NETWORKS)*. IEEE, Oct. 2012.

[6] S. Reif, A. Schmidt, T. Hönig, T. Herfet, and W. Schröder-Preikschat, "X-LAP: A systems approach for cross-layer profiling and latency analysis for cyber-physical networks," *ACM SIGBED Review*, vol. 15, no. 3, pp. 19–24, 2018.

[7] A. Protopsaltis, P. Sarigiannidis, D. Margounakis, and A. Lytos, "Data visualization in internet of things: tools, methodologies, and challenges," in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ser. ARES 2020. ACM, Aug. 2020.

[8] Z. Liu and J. Heer, "The effects of interactive latency on exploratory visual analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2122–2131, Dec. 2014.

[9] B. Gregg, "Visualizing system latency," *Communications of the ACM*, vol. 53, no. 7, pp. 48–54, Jul. 2010.

[10] S. Di Bartolomeo, A. Pandey, A. Leventidis, D. Saffo, U. H. Syeda, E. Carstensdottir, M. Seif El-Nasr, M. A. Borkin, and C. Dunne, "Evaluating the effect of timeline shape on visualization task performance," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, ser. CHI '20. ACM, Apr. 2020.

[11] E. Dahlman, S. Parkvall, and J. Sköld, *5G NR*. Elsevier, 2021.