2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) | 979-8-3503-5805-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/IDAACS58523.2023.10348913

The 12th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications
7-9 September, 2023, Dortmund, Germany

End-to-end Latency Measurements with Black Box Components

Abdullah Muslim, Stephan Recker
University of Applied Sciences and Arts Dortmund, EFS-42 C.3.33 44227,
abdullah.muslim@fh-dortmund.de,
stephan.recker@fh-dortmund.de

Abstract— **Edge computing has gained substantial attention in both academia and industry due to its ability to facilitate cloud services provision in close proximity to the user and data source, thereby effectively reducing latency. This is particularly important for latency-sensitive or *tactile* Internet services that cannot tolerate delays caused by transferring workloads to distant cloud computing data centers. User equipment generating workloads for offloading to the edge encompasses a wide range of device types, from small sensors to specialized integrated devices like augmented reality wearables, and even powerful mobile computing devices like smartphones, tablets, or laptops. Since low latency is the foremost factor for utilizing edge computing services, it is imperative to identify and resolve any latency problems that might occur between the user equipment and the edge server. Nevertheless, accurately measuring latency becomes a challenging task when the edge service setting comprises closed black box components or entire systems. For instance, closed user equipment, such as augmented reality wearables, typically do not allow intrusive measurement techniques, and synchronization with other components of the setting is often not possible. To address these challenges, this article proposes a pragmatic end-to-end latency measurement approach specifically designed for edge computing settings with black box components. This approach leverages performance data provided by the Real-Time Control Protocol (RTCP) to obtain accurate latency measurements. The results obtained using this approach are validated through reference scenarios with only white box components, and exemplary latency measurements within an edge computing setting deploying a private 5G network infrastructure are presented.**

Keywords — edge computing, webRTC, Real Time Protocol, latency measurements, TWAMP

I. MOTIVATION

The development of advanced technologies required for creating and manufacturing intelligent mobile devices and providing mobile internet solutions has enabled a wide range of new and increasingly intricate applications, such as augmented or virtual reality-based applications, unmanned or self-driving vehicles, and image recognition and manipulation applications. In the context of the cloud-edge continuum, use cases for edge computing typically involve processing data in close proximity to its source, resulting in low latency between the user equipment and the computation server processing the data. Augmented or virtual reality applications, services involving Vehicle-to-Everything communication or autonomous driving, and a multitude of IoT services [1] are the most frequently examples. Research often mentions the deployment of Microsoft's HoloLens 2 (HL2) for Augmented Reality use cases. One specific use case involves Mixed Reality medical applications, such as surgical telementoring, which facilitates organ transplantation with mixed reality support for a remote explantation team receiving in-depth visual information from the on-site implantation team [2]. In such scenarios, ensuring an appropriate level of *Quality of Service* (QoS) is crucial, with latency being a key metric to consider. When it comes to surgical telemonitoring, a moderate level of latency is usually sufficient, unlike real- time distributed processes like autonomous driving that require ultra-low latency. However, excessively high latencies would negatively affect the efficacy and timeliness of surgical actions [2]. Therefore, it is necessary to obtain accurate information on the actual path latency between the local and remote surgical teams to assess the feasibility of the telemonitoring system. To refer to the same baseline, it's necessary to synchronize all measurement points while measuring latency [3]. However, obtaining system access for a device and synchronizing all devices, including end devices owned by different entities in public networks, is generally not feasible. Moreover, wearable devices like HoloLens 2 are considered black box devices as they don't offer system access to implement *Operating System* (OS) level measurement probes. This article examines the options available for latency measurements in a general setting where devices are not synchronized, and measurement probes are only possible on specific white box devices. To determine network level latencies, reference scenarios with synchronized white box devices are used, where packet capturing is employed to assess the accuracy of possible measurement approaches. During the experiment parallel measurement were also done using *Two Way active Measurement Protocol* (TWAMP) for reference in the same network infrastructure. Furthermore, the inclusion of these measurements would make a substantial contribution towards improving the realism of the environmental model on emulation platforms. The

structure of this article is as follows: Section II will review the related work on edge computing and latency measurements. Then, in Section V, the article will describe the experimental set-up, including the baseline reference scenarios, and present the measurement results. Finally, Section VI will conclude the article and provide a glimpse of future work.

II. Related Work

Several studies have been conducted on latency measurements in various settings. Aslanpour et al provide a taxonomy of possible metrics for evaluating the performance of edge computing [4]. They highlight the significance of latency distribution tails, which have the most significant impact on user experience, although these events are rare. Chen et al conducted an empirical study on latencies for wearable device applications [5]. However, they avoided a practical set-up using such wearables and instead used a self-made integrated edge computing and application offloading solution that also provided integrated measurement probes for latency. Pogel et al explored passive measurement approaches for scarce bandwidth links, such as cellular network links [6]. Their findings support the use of inband measurement approaches, such as the RTCP-based methodology applied in this article, to avoid additional payload on low bandwidth links. The limitations of the most prominent latency measurement tools, such as *ping* and *traceroute*, have already been highlighted in previous research, such as by Pelsser et al [7]. While Corneo et al use traceroute to obtain a latency-topology graph with latencies in the range of several 10ms for edge and cloud data centers [8], it can be concluded that these tools would not provide sufficiently accurate results for the comparatively small expected latencies in the ms range in the scenarios considered in this article. Besides network latency, processing delays in compute nodes have also been measured. For example, Friston and Steed proposed a framework for latency measurements in virtual environments [9]. While a round trip time between systems inherently includes at least the processing time at the responder system, their work focuses on the latency between the arrival time of a video frame at a server and the user's visual perception of that frame.

III. Latency Measurements based on RTCP

For real-time audio and video streaming, WebRTC utilizes a combination of Real-Time Protocol (RTP) and Real-Time Control Protocol (RTCP). Real-time audio and video data are sent using RTP, which works at the transport layer alongside UDP [10]. RTP header contains a sequence number for the detection of packet loss in a real-time media stream. RTCP is used to monitor and deliver statistical information about the media stream to RTP session participants, such as packet counts, packet delay variation, transmitted octet, and round trip delay time. The aforementioned information enables the rate of data transmission in the session.

RTP uses a dedicated UDP port to provide real-time audio and video data, whereas RTCP uses a separate UDP port to send control information. This design results in the use of distinct and separate UDP ports for transmitting RTP and RTCP packets for every audio and video stream, thereby ensuring efficient and reliable communication during a WebRTC session [11].

WebRTC uses RTP packets to provide real-time, bidirectional video communication between peers. After the connection is established, video streaming is initiated using RTP packets, and each receiver periodically transmits RTCP packets to provide feedback on the media stream quality to the other participants. The frequency of RTCP messages is determined by the number of participants in the session.

The RTCP packet header comprises a payload type that serves to identify the type of packet being used. The RTCP protocol supports five distinct types of payload, which are specified in [10]. These include *Sender Reports* (SR), *Receiver Report* (RR), *Source Description* (SDES), *Goodbye* (BYE), and *Application-specific* (APP) functions. To facilitate the provision of feedback on the quality of received data, RTP receivers rely on RTCP reports, which are categorized into two distinct types: *Sender Reports* (SR) and *Receiver Reports* (RR). The RR packet type is used to determine the *Round-Trip Time* (RTT) between the sender and receiver. The format of an RR packet features 32-bit size fields containing timestamps from the last received SR (LSR) and the delay since the last received SR (DLSR) [10]. These timestamps are utilized to calculate the RTT between the sender and receiver of the media stream.

Fig 1 illustrates how the RTT between peers is calculated. The sender transmits an SR RTCP packet that incorporates a 32-bit Network Time Protocol (NTP) timestamp. This timestamp is stored by the receiver in the LSR field of the subsequent RR RTCP packet. Additionally, the amount of time that has passed since the last sender report was received by the receiver is recorded in the DLSR field of the RR RTCP packet. The RR packet features both the LSR and DLSR fields.

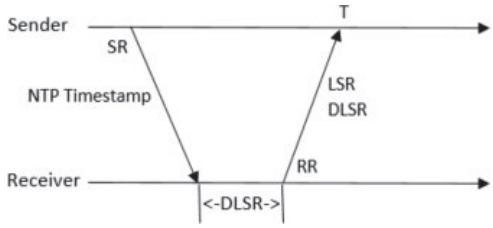

Figure 1. RTT Measurement Using RTCP.

On the sender's side, the RTT is calculated using the equation outlined in (1), where T represents the time at which the RR packet is received by the sender.

$$Delay = T - LSR - DLSR \quad (1)$$

IV. Latency Measurement Based On TWAMP

Two way Active Measurement Protocol (TWAMP) enables the measurement of service quality between two IP network devices. It is a modern standard that supports layer-3 protocols of the OSI reference model. TWAMP is composed of two protocols:

- TWAMP-Control
- TWAMP-Test

The TWAMP-Control is responsible for initiating, starting, and stopping test sessions, while the TWAMP-Test is used to exchange test packets between the devices to mea- sure network metrics. The TWAMP architecture consists of four logical roles, which are *Control-Client*, *Server*, *Session-Sender*, and *Session-Reflector*. These roles can be implemented on different devices, but in practice, the Control-Client and Session-Sender are implemented on a single device as a Controller, while the Server and Session-Reflector are implemented on the other device as a Responder [12] as shown in Fig 2.

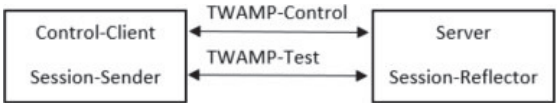

Figure 2. RTT measurement using TWAMP.

The TWAMP-Test protocol is designed to facilitate the computation of performance metrics by enabling the exchange of test packets between two network points, utilizing the UDP protocol for packet transmission. Session- Sender sends the test packet to the Session-Reflector. The Session-Reflector, on the other hand, is responsible for sending back the test packets sent by the Session-Sender without collecting any packet information. Session-Sender collects and records information communicated by the Session-Reflector to measure two-way metrics accurately [12].

V. Experiment Set-Up and Results

The experiment setup is similar to the surgical monitoring use case. The setup consists of an HL2 device which represents the explantation team capturing device. The HL2 sends video streams to the edge server, where a Python script captures video frames, performs image processing, and forwards the processed stream to a laptop display as well as back to the HL2 for visualization as shown in the fig 3. Private and public networks such as WiFi, 5G, and wired Ethernet are used for connectivity between the HL2, server, and display. To measure latency, the common approach is, by capturing the timestamps of packets, sent from the source device (HL2) and received by the destination device (display). However, as the HL2 does not allow access to implement latency measurement

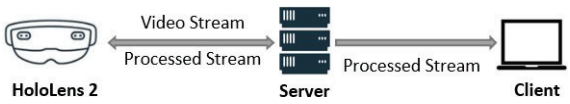

Figure 3. Video Streaming Using WebRTC.

To provide a measurement reference, selected scenarios were replicated by substituting the HL2 with a laptop as a video source, equipped with the packet-capturing software tool Wireshark [13]. Additionally, packet capturing was enabled on the server to allow the recording of the times- tamps of the RTCP packets, which is used in WebRTC to calculate RTT as shown in fig 4. Similarly, during the RTCP measurement, the Python scripts for TWAMP- Controller and TWAMP-Responder run concurrently on the laptop and server that are also used for RTCP measurements, providing a parallel approach to measuring latency. The Controller sends the test packets to the Responder, which then sends the test packets back to the Controller. The Controller python Script then measures the RTT. Latency is measured by taking the half of RTT.

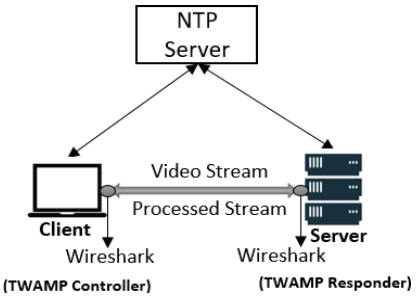

Figure 4. NTP Synchronization.

Additionally, 5G mobile Samsung S21 called Nemo Handy [14] provided by Keysight which supports the TWAMP protocol is utilized to measure the latency in a campus private 5G network from 5G mobile to server. A Docker container was deployed on the server to act as a TWAMP Session-Reflector for 5G mobile TWAMP test packets. The Keysigth enables on-device measurements to be performed at the User Equipment (UE), where all data protocols and services are operational [14].

The experiments were conducted for a duration of 100 minutes, and the resulting data was used to generate a Cumulative Density Function (CDF) graph of the latency.

A. *Scenario 1*

This scenario is being carried out in internal campus network, consisting of a laptop connected wirelessly to an Access Point, a switch, and a server as demonstrated in Fig 5. The laptop is transmitting a video stream to the server for image processing, which then transmits it back to the laptop. This setup serves as a standard point of reference to authenticate the results of the RTCP Latency measurement.

Figure 5. Illustration of measurement setup.

The results show in fig 6 that the CDF for all three measurements is close to each other. The graph shows that 95% of the latency values for the Wireshark are approximately below 5ms. For the RTCP and the TWAMP, 95% of the values are approximately below 6ms. The mean value obtained through Wireshark is relatively low, likely due to the capturing of packet timestamps at the physical layer. In contrast, the TWAMP protocol measures the Round-Trip Time (RTT) at the Transport layer, resulting in a slightly higher mean value.

Finally, the mean value for RTCP is higher than the other two measurements due to the additional delay caused by application processing as shown in table I.

Table I. Statistical Measurements of Scenario 1

Statistics	RTCP	TWAMP	Wireshark
Mean	4.621ms	3.697ms	3.577ms
Standard Deviation	1.751ms	2.508ms	1.229ms

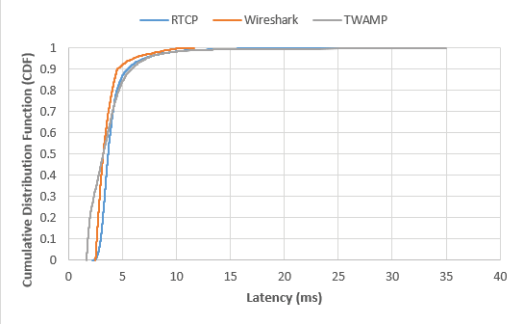

Figure 6. CDF of Latency, measured using RTCP, Wireshark and TWAMP.

B. *Scenario 2*

The experimental configuration shown in fig 7 is similar to deploying an edge server in an urban network, with the remote server placed at a distance of 8 km. The setup consists of a switch, an externally hosted server, and an HL2 connected to a wireless access point. The server is situated in a data center that is connected to the campus network on layer 2, and there are no firewalls present along the path to the external server. A WebRTC application is utilized to transmit the video stream from the HL2 device to the server, while the network latency is monitored using RTCP measurement.

Figure 7. Simulating as an Edge server scenario in our local network.

The measurement in the fig 8 shows that approximately 95% of the values are below 15ms. The mean and standard deviation values are 7.743ms and 5.247ms, respectively. The reason for high standard deviation value is due to the hardware limitation of HL2 and having a wireless connection.

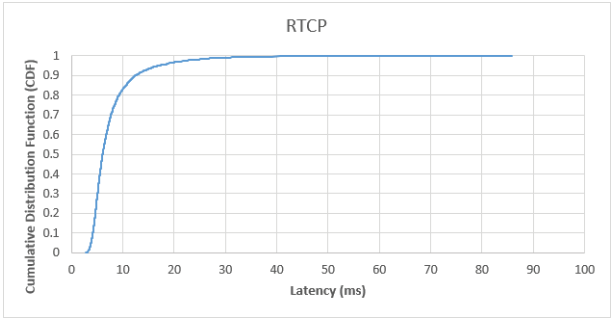

Figure 8. CDF of Latency measured in Scenario 2.

C. *Scenario 3*

The measurement depicted in the fig 9 was conducted within a private 5G campus network and serves as a reference measurement. The setup involved a laptop connected to a 5G mobile (S21 Nemo Handy) through WiFi tethering. The 5G mobile was then linked to a 5G base station, which in turn connected to a switch in the campus network and the 5G core. Moreover, the switch was connected to a server. In this configuration, alongside the laptop sending RTCP and TWAMP packets for measurement, 5G mobile also transmits the TWAMP test packet for the RTT measurement as described in section V.

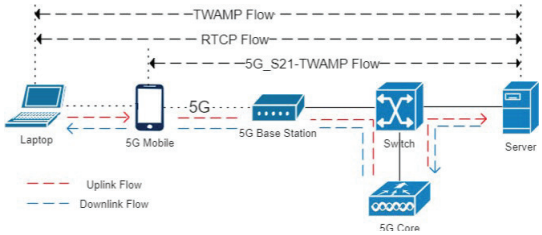

Figure 9. Latency measurement set-up with private 5G Network.

The fig 10 displays the CDF of the latency measurements. The 95% latency value from 5G mobile to the server is approximately below 15ms, with a mean value of 9.881ms as indicated in the table II. The dispersion of latency values from the mean value is 2.79ms, which is relatively small because the 5G mobile is placed near to the base station during the experiment. Similarly, the graph indicates that for roughly 95% of the time, the latency values in RTCP and TWAMP measurements is less than 40ms. However, the TWAMP shows a few higher latency peaks, resulting in a greater standard deviation than the others, as shown in the table II. These peaks could be caused by temporary network congestion. The mean value for RTCP is higher than TWAMP due to additional processing delay from the WebRTC application.

Table II. Statistical Measurements of Scenario 3

Statistics	RTCP	TWAMP	5G-TWAMP
Mean	22.396ms	14.775ms	9.881ms
Standard Deviation	7.492ms	10.019ms	2.792ms

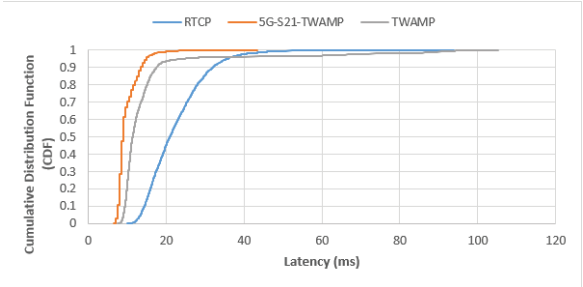

Figure 10. CDF of Latency measured in 5G private Network using RTCP, TWAMP, and 5G-S21-TWAMP.

D. *Scenario 4*

Fig 11 illustrates a setup for measuring latency in a private 5G network. The setup consists of HL2, which is connected via a wired USB link to a 5G router. The 5G router is then linked to a 5G base station or cell, which is further connected to a switch within the campus network and the network's core. Additionally, the switch is connected to a server. HL2 transmits the video stream to the server to examine the impact of different network components on latency during real-time communication. The CDF in fig 12 shows that approximately 95% of latency values from HL2 to server are below 20ms. The mean value is 12.063ms, and the standard deviation is 4.388ms. However, when the same experiment is carried out using a laptop as a video source, the mean value drops to roughly 10.642ms. This is because HL2 adds 1ms to 2ms of latency in comparison to the laptop due to its low hardware quality.

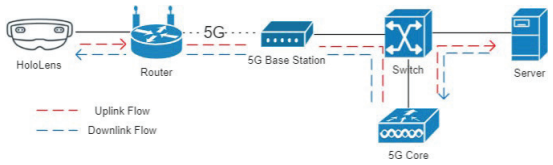

Figure 11. Latency measurement set-up with private 5G Network.

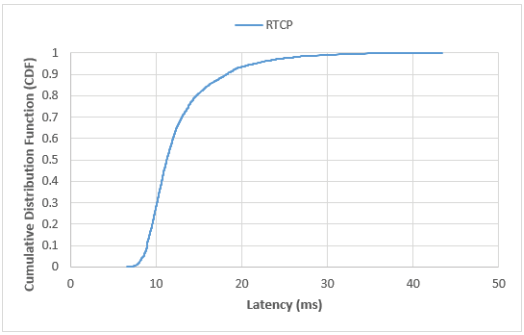

Figure 12. CDF of latency, measured in Scenario 4.

E. *Scenario 5*

The scenario illustrated in fig 13 aims to evaluate the latency of Vodafone's public 5G network at the University Hospital Dusseldorf. This particular setup has been designed to closely simulate real-world conditions for the surgical mentoring use case. The setup contains HL2 connected to a 5G mobile device via a wired USB link, and the 5G mobile device accessing Vodafone's public 5G network. The HoloLens device transmits real-time video streams to the public edge server located in Dortmund, spanning a distance of approximately 100km through the Vodafone core network and public internet links.

The CDF illustrated in fig 14 indicates that 93% of the latency values for this setup are below 20ms, with a mean value of 16.744ms and a standard deviation of 2.07ms. The relatively low deviation of latency values from the mean can be attributed to the wired connection between the HL2 and 5G mobile, as well as the experiment being conducted close to the 5G base station, which aids in low latency.

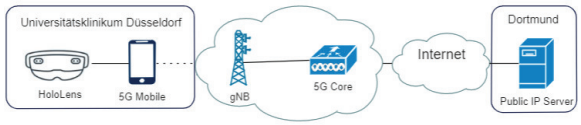

Figure 13. Latency measurement set-up with public 5G network from Düsseldorf to Dortmund.

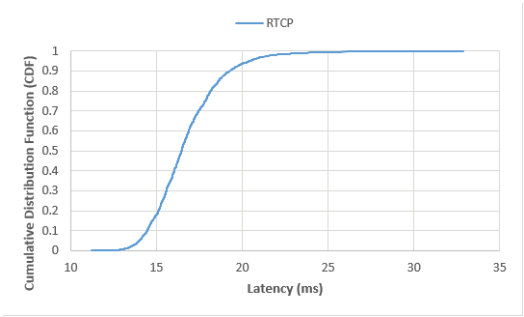

Figure 14. Latency measurement set-up with private 5G Network.

VI. Conclusion and Future Work

This paper conducts a statistical analysis of latency measurement results in an Edge computing setting using the HoloLens device as User equipment. The validation of RTCP measurements is performed using TWAMP protocol and packet capturing tool Wireshark, and Nemo Handy provided by Keysight is also used to validate measurements in 5G scenarios. The study finds that in a simulated Edge server scenario where the server is located at a distance of 10km and HL2 is connected to the Access Point wirelessly. The result shows, 96% of the latency values are below 20ms. Moreover, in private 5G network scenarios, the latency values from 5G mobile to the server range from 7.5ms to 16.5ms 97% of the time. Additionally, when the HL2 is connected wired with the 5G router, the latency from HL2 to the server indicates that 95% of the latency values are below 22ms, which is higher due to radio signal transmission through Ethernet between a base station and the 5G core network. In a public 5G network scenario, 83% of the latency values range from 15ms to 20ms, 16% are below 15ms, and 7% are above 20ms. The deviation of latency is less in the public 5G network due to the base station's proximity to the User Equipment. Based on the results, it can be inferred that a wired connection with the UE device and conducting the experiment in close proximity to the base station results in low latency values. Conversely, when the connection is wireless with the UE, high latency values are expected with a high standard deviation. The overall assessment of the deployed measurement approach utilizing RTCP packets reveals consistent and reliable results, which have been verified through additional validation measurements using TWAMP and packet-capturing tool to confirm accuracy. Ultimately, the findings from these results would be valuable in developing a highly realistic model on the emulation platform.

Future research could involve conducting measurements in a laboratory setting that utilizes more sophisticated network models, featuring multiple clients. Furthermore, considering the device's mobility in these measurements would yield a more complete understanding of its impact on latency. In addition, the mathematical model for latency would take into consideration and enable the comparison with the obtained measurement results.

ACKNOWLEDGMENT

This work was conducted as part of project EMULATE funded by the German Federal Ministry of Economics and Climate Action under research grant 13IPC012.

REFERENCES

[1] ETSI, "Multi-access edge computing (mec); phase 2: Use cases and requirements," ETSI GS MEC 002 V2.2.1, March, Tech. Rep., 2022.

[2] B. Dewitz, R. Bibo, S. Moazemi, S. Kalkhoff, S. Recker,

[3] A. Liebrecht, A. Lichtenberg, C. Geiger, F. Steinicke, H. Aubin, and F. Schmid, "Real-time 3d scans of cardiac surgery using a single optical-see-through head-mounted display in a mobile setup," *Frontiers in Virtual Reality*, vol. 3, 2022.

[4] R. Durairajan, S. K. Mani, J. Sommers, and P. Barford, "Time's forgotten: Using ntp to understand internet latency," in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, 2015, pp. 1–7.

[5] "Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research," *Internet of Things*, vol. 12, p. 100273, 2020.

[6] Z. Chen, W. Hu, J. Wang, S. Zhao, B. Amos, G. Wu, K. Ha,

[7] K. Elgazzar, P. Pillai, R. Klatzky, D. Siewiorek, and M. Satyanarayanan, "An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, ser. SEC '17. New York, NY, USA: Association for Computing Machinery, 2017.

[8] T. Po¨gel, J. Lu¨bbe, and L. Wolf, "Passive client-based bandwidth and latency measurements in cellular networks," in *2012 Proceedings IEEE INFOCOM Workshops*, 2012, pp. 37–42.

[9] C. Pelsser, L. Cittadini, S. Vissicchio, and R. Bush, "From paris to tokyo: On the suitability of ping to measure latency," in *Proceedings of the 2013 conference on Internet measurement conference*, 2013, pp. 427–432.

[10] L. Corneo, N. Mohan, A. Zavodovski, W. Wong, C. Rohner,

[11] P. Gunningberg, and J. Kangasharju, "(how much) can edge computing change network latency?" in *2021 IFIP Networking Conference (IFIP Networking)*. IEEE, 2021, pp. 1–9.

[12] S. Friston and A. Steed, "Measuring latency in virtual environments," *IEEE transactions on visualization and computer graphics*, vol. 20, pp. 616–25, 04 2014.

[13] H. Schulzrinne, S. L. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550, Jul. 2003. [Online]. Available: https://www.rfc-editor.org/info/rfc3550

[14] C. Perkins and J. Ott, "Guidelines for Extending the RTP Control Protocol (RTCP)," RFC 5968, Sep. 2010. [Online]. Available: https://www.rfc-editor.org/info/rfc5968

[15] R. Krzanowski, K. Haydayat, A. Morton, K. Yum, and J. Babiarz, "A two-way active measurement protocol (twamp)," Internet Engineering Task Force (IETF), 2008, rFC 5357. [Online].

[16] Available: https://tools.ietf.org/html/rfc5357

[17] Y. Orzach, *Network analysis using Wireshark cookbook: over 80 recipes to analyze and troubleshoot network problems using Wireshark*, ser. Quick answers to common problems. Birmingham: Packt Publ., 2013.

[18] KEYSIGHT, "Nemo handy fast and efficient on the go network measurement and troubleshooting," 2023, flyer.