



# User Plane Positioning in 5G Networks: A Design Space Study for Implementation

1<sup>st</sup> Nico Kalis 

*Institute of Applied Microelectronics  
and Computer Engineering  
University of Rostock  
Rostock, Germany  
nico.kalis@uni-rostock.de*

2<sup>nd</sup> Christian Haubelt 

*Institute of Applied Microelectronics  
and Computer Engineering  
University of Rostock  
Rostock, Germany  
christian.haubelt@uni-rostock.de*

3<sup>rd</sup> Frank Golatowski 

*Institute of Applied Microelectronics  
and Computer Engineering  
University of Rostock  
Rostock, Germany  
frank.golatowski@uni-rostock.de*

**Abstract**—Fast and accurate positioning of medical service robots is a challenging task. Therefore, this paper presents a performance analysis of various implementation designs around the Location Services User Plane Protocol (LCS-UPP), which is used in fifth generation (5G) networks to transport position-related data. More precisely, the transport time of the LCS-UPP is examined depending on the transported LTE Positioning Protocol (LPP) payload size, different Transport Layer Security (TLS) 1.3 implementations and the successive movement of the data path to the underlying Linux operating system. For this purpose, a Raspberry Pi Zero 2 W in combination with the Quectel RM520N-GL modem as a 5G user equipment (UE) as well as three additional Raspberry Pi 5 and the Ettus USRP B210 Software Defined Radio (SDR), which form the radio access and the core network, are used. The results show that the transport time of the LCS-UPP can be reduced by 42.01%, if the QUIC-based MsQuic user space library is used instead of the wolfSSL library that based on the Transmission Control Protocol (TCP). Also the complete movement of the LCS-UPP data path to the Linux operating system decreases the transport time by 5.84% and 16.95% compared to the wolfSSL library and the in-kernel TLS implementation of Linux, respectively.

**Index Terms**—5G, positioning, LCS-UPP, networking performance

## I. INTRODUCTION

IN terms of time, the positioning of targets is a challenging task on resource-constrained devices. This can be observed, for example, in hospitals, where medical service robots and self-driven patient beds have to localize themselves in a time-critical and very accurate manner. Due to limited resources at the target, for instance, the medical service robot, the calculation of its current position can be moved to more powerful computer hardware. Therefore, a possible solution to realize positioning for both use cases could be the deployment of a distributed system, where the target initially records measurement data, such as signal strength metrics or transmission times. Afterwards, these measurements are sent to a server that calculates the target's current position using the received data.

In practice, the usage of wireless communication technologies, which are summarized in [1], would be suitable to exchange position-related data with respect to both use cases. Due to existing positioning features, this paper focuses on 5G

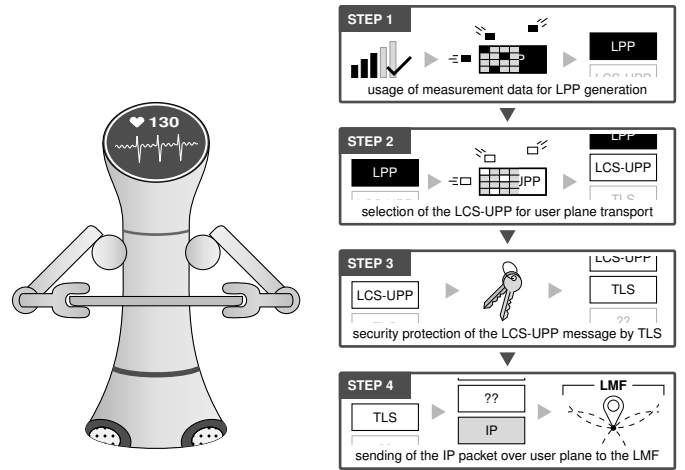


Fig. 1. 5G positioning in hospitals: A medical service robot records measurement data, generates a LPP message (step 1) and sends it to the LMF of the 5G core network (step 4) by using the user plane positioning feature that based on the LCS-UPP (step 2) and TLS protection (step 3).

networks to investigate positioning performance in terms of time that is required to exchange position-related data.

With the introduction of Release 18 of the 5G mobile communication standard, which is published by the *3rd Generation Partnership Project (3GPP)*, the LCS-UPP has been defined ([2]). In this context, Figure 1 shows a general scenario of how this protocol could be integrated into the described use case of a medical service robot. Therefore, after the robot has completed the measurement recording, for instance, of signal strength parameters, it can put them in a LPP message that is defined by the 3GPP in [3]. To transport this message to a server, the LCS-UPP realizes this by using the *user plane* of a 5G network to address the *Location Management Function (LMF)* directly, which is the key enabler for positioning.

Figure 1 also shows that the LPP transport over the user plane is encapsulated in *Internet Protocol (IP)* packets. This is an advantage that simplifies the development of 5G positioning systems on top of an operating system and makes the implementation largely independent of specific 5G hardware. However, these characteristics only apply to experimental 5G

test environments because the usage of LCS-UPP must first be negotiated between the LMF and the UE over the *control plane* ([4]). This is currently, to the best of the author's knowledge, not supported by either commercial or open source solutions.

Additionally, the LCS-UPP data transmission should be protected by the TLS protocol, as shown in Figure 1. Due to the facts that the 3GPP does not specify the concrete TLS implementation in [2] and [5] as well as that there are several TLS implementation approaches, for instance, using different transport layer protocols, such as the TCP or the QUIC protocol on top of the *User Datagram Protocol (UDP)*, the time behavior of data exchange can be affected differently.<sup>1</sup>

In combination with the impact of *context switches* between the application and the underlying operating system, the transport of measurement data can be impacted negatively. Therefore, the paper at hand presents two contributions:

- ① **implementation** of two LCS-UPP procedures to enable the transport of LPP messages within a 5G network according to the 3GPP in [2],
- ② **performance investigation** of the transport time  $t_t$  of the LCS-UPP depending on the LPP payload size  $n$  by using different TLS 1.3 implementations and the successive movement of the *data path* to the underlying Linux operating system, respectively.

In the following, Section II gives an overview of related work. Afterwards, Section III describes the theoretical approach of how the performance of the of LCS-UPP is investigated and which implementation designs are considered. Based on this, the experimental setup and the achieved results are considered in the Sections IV and V, respectively. Finally, Section VI summarizes the key points of this paper.

To improve the readability of each section, Table I summarizes selected abbreviations of the paper at hand.

TABLE I  
OVERVIEW OF SELECTED ABBREVIATIONS

Short	Long
3GPP	3rd Generation Partnership Project
DL	DownLink
DTLS	Datagram Transport Layer Security
IP	Internet Protocol
LCS-UPP	LoCation Services User Plane Protocol
LMF	Location Management Function
LPP	LTE Positioning Protocol
QUIC	QUick Internet Connections
SDR	Software Defined Radio
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UE	User Equipment
UL	UpLink
XDP	eXpress Data Path

<sup>1</sup>The transport protocol layer in Figure 1 is represented by ??.

## II. RELATED WORK

The performance within 5G networks has already been investigated in research from a variety of perspectives. For instance, Barbosa *et al.* examine the performance of various open source 5G core network solutions in [6]. By comparing *Open5GS*, *OpenAirInterface* and *free5GC*, they can show that the former provides the best control plane implementation in terms of latency, OpenAirInterface offers the highest throughput and Free5GC has the lowest resource consumption. In contrast to the user plane investigation in [6], the paper at hand does not focus on the user plane performance of a 5G core network directly. Instead, the impact of context switches and different TLS implementations on the time behavior of the transport of the LCS-UPP over the user plane is considered.

Håkegård *et al.* investigate the performance in terms of coverage, throughput and latency of a 5G network in [7]. By using the open source software systems Open5GS and srsRAN in combination with a workstation and the Ettus USRP B210 SDR, the authors show that the coverage of their selected 5G platform is limited to a few meters. On the other hand, the throughput can be improved by increasing the bandwidth as well as using 256-QAM instead of the 64-QAM modulation scheme. Additionally, Håkegård *et al.* consider the round trip time metric depending on different frame structures to analyze the latency of their 5G platform. In contrast, the paper at hand investigates the transport time of the LCS-UPP depending on, for instance, the LPP payload size. Furthermore, the used 5G test environment of the paper at hand based on the same software systems as in [7]. However, three Raspberry Pi 5 are used instead of a workstation to evaluate the time behavior of the LCS-UPP transport in a more realistic worst case scenario.

Zeidler *et al.* examine the impact of different TLS versions and their encryption algorithms on the performance of 5G core networks in [8]. More precisely, they consider the time overhead of TLS 1.2 and 1.3 for two selected core network procedures depending on different cipher suites and the corresponding resource consumption in warm and cold start scenarios. The authors can show that TLS performs generally better in warm start scenarios. In this case, the memory and CPU consumption of TLS can be considered negligible. In contrast to their investigations, the paper at hand applies TLS to the 5G user plane to determine the impact of different TLS 1.3 implementations on the LCS-UPP transport time by using the `TLS_AES_128_GCM_SHA256` cipher suite.

To improve the TLS performance, the QUIC protocol can be used. Due to the fact that the latter based on UDP instead of TCP, QUIC performs much better in terms of throughput. This is shown by Wang *et al.* in [9], where the authors propose an in-kernel solution of the QUIC protocol. Their performance evaluation shows that QUIC is more robust than TCP in terms of throughput, if the packet loss rate increases. This paper applies the findings of the approach in [9] to the LCS-UPP by investigating its performance depending on context switches between user and kernel space as well as the usage of TLS in combination with TCP and QUIC, respectively.

In addition to performance investigations of 5G networks, Zanini *et al.* present the current state of research with respect to available 5G positioning systems and propose an LMF implementation in [10]. In their work, the authors point out that 5G positioning features have so far only been supported to a limited extent by open source and commercial systems. However, they only address the usage of 5G positioning procedures and protocols within the control plane. Therefore, the paper at hand extends the current state of research by contributing an implementation of two LCS-UPP procedures that enable positioning over the user plane.

### III. THEORETICAL APPROACH

In order to transmit position-related data as LPP messages over the user plane by using the LCS-UPP, the 3GPP defines the following two procedures in [2]:

- ① **UL LCS-UP Transport**, which primarily enables the transmission of LPP messages in *uplink* (UL) direction from a UE to the 5G core network.
- ② **DL LCS-UP Transport**, which primarily enables the transmission of LPP messages in *downlink* (DL) direction from the 5G core network to the target UE.

Both procedures are used to evaluate the time behavior of the LCS-UPP. More precisely, the *transport time*  $t_t$  is investigated, which represents the time difference between the generation of the UL LCS-UP message and the complete processing of the corresponding DL LCS-UP message. In this regard, Figure 2 shows the sequence of steps that are performed to determine the investigated metric. In particular, the third and the twelfth step are crucial for the calculation of the transport time  $t_t$ .

Before the time measurement is started, a data path implementation type is set on UE and LMF side, respectively.<sup>2</sup> As shown in the first step of Figure 2, three different types are distinguished. These can be characterized as follows:

- ① **Type 1** addresses different TLS user space implementations. In particular, the impact of TLS in combination with TCP and QUIC, respectively, on the transport time  $t_t$  is primarily examined.
- ② **Type 2** addresses the movement of the TLS data path to the underlying operating system.<sup>3</sup> It is investigated, if the transport time  $t_t$  can be reduced in comparison to a TCP-based user space implementation of Type 1.
- ③ **Type 3** corresponds to a complete in-kernel solution. Due to this fact, the impact of missing context switches between user and kernel space on the transport time  $t_t$  is examined compared to Type 1 and Type 2.

After one of these types has been selected, the UE generates a LPP message of size  $n$ , where  $n$  is defined by the subset

$$n \in \{2^i, 2^{16} - 1 \mid i \in \mathbb{N}_0, i \leq 15\}. \quad (1)$$

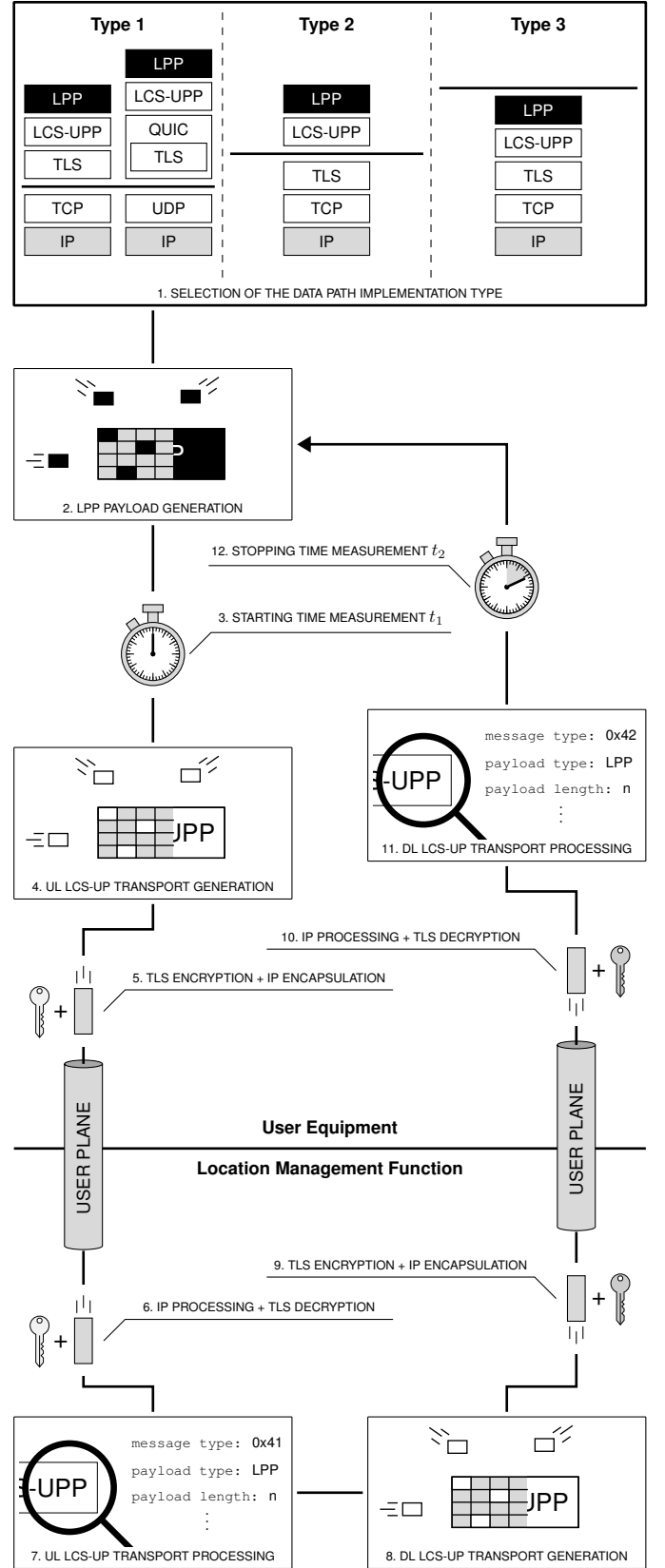


Fig. 2. Performance investigation of the LCS-UPP: Based on a selected data path implementation type (step 1) on UE and LMF side, the LCS-UPP transport time  $t_t$  is calculated by the timestamps  $t_1$  (step 3) and  $t_2$  (step 12).

<sup>2</sup>The same data path implementation type is set on both sides.

<sup>3</sup>The kernel space is located below the horizontal line in Figure 2.

While the impact of the doubling of  $n$  on the transport time  $t_t$  is examined, the upper bound of  $n$  is set to the maximum payload size of a LCS-UPP message, as described in [2].

In the third step, the timestamp  $t_1$  is taken. This corresponds to the start time, which is required to calculate the transport time  $t_t$  at the end of a single iteration. Afterwards the UE generates the UL LCS-UP Transport message that includes the created LPP message from the second step.

Before the LCS-UPP message is sent to the LMF over the user plane, the former is encrypted by using TLS, as shown in Figure 2. Within the paper at hand, TLS 1.3 is generally used together with the `TLS_AES_128_GCM_SHA256` cipher suite for each encryption and decryption of LCS-UPP data.

After the UL LCS-UP Transport message has been successfully received and decrypted, the LMF verifies the request message syntactically and semantically. As illustrated in the seventh step of Figure 2, the message type as well as the payload type and its length are checked. However, Figure 2 does not show *session-related* parameters, which are also taken into account by the LMF. According to [2], this parameter set has a total length of 256 Byte. To investigate a more realistic worst case scenario, the corresponding parameters are set to their maximum values in terms of data size.

If the verification of the UL LCS-UP Transport message is successful, the LMF generates a corresponding DL LCS-UP Transport message that also includes a LPP message of size  $n$  to *echo* the UE's request. However, the LPP message generation on LMF side is not shown in Figure 2.

When the encrypted DL LCS-UP Transport message has passed the user plane of the 5G network and was successfully decrypted on UE side, the latter also verifies the received LCS-UPP data in the eleventh step of Figure 2. If the verification is successfully completed, a second timestamp  $t_2$  is taken. Afterwards, the investigated transport time  $t_t$  can be determined using both timestamps  $t_1$  and  $t_2$ . Within the scope of this paper, the transport time  $t_t$  is calculated as follows:

$$t_t = t_2 - t_1 \text{ with } [t_t] = [\text{ms}] . \quad (2)$$

For the evaluation of the transport time  $t_t$ , statistical methods are used to implicitly quantify the *uncertainty* as well. More precisely, the sequence of steps shown in Figure 2 are repeated  $m$  times for each data path implementation type and value of  $n$ . The value of  $m$  is set to

$$m = 10^3 = 1000 , \quad (3)$$

in order to construct a meaningful 99% confidence interval. This presupposes that the sample mean  $\bar{x}$  and the standard error  $\sigma_{\bar{x}}$  have to be calculated first ([11]). By using the quantile  $z_{0.005}$  of the standard normal distribution with

$$z_{0.005} = 2.58 , \quad (4)$$

a 99% confidence interval is calculated by

$$[\bar{x} - z_{0.005} \cdot \sigma_{\bar{x}}, \bar{x} + z_{0.005} \cdot \sigma_{\bar{x}}] . \quad (5)$$

This confidence interval is applied to the transport time of different implementations, which are described in Section IV.

## IV. EXPERIMENTAL SETUP

To determine the transport time  $t_t$  of the LCS-UPP for a more realistic worst case scenario, this section deals with the setup of a 5G network that based on hardware with limited processing power. In this context, Figure 3 shows the hardware components of this setup, their role within a 5G network and the user plane establishment between the UE and the LMF. More precisely, the LCS-UPP data exchange, which is crucial for the transport time calculation, is performed over the user plane between the Raspberry Pi Zero 2 W on the top and the Raspberry Pi 5 on the bottom of Figure 3.

The Raspberry Pi Zero 2 W and the Raspberry Pi 5 are single-board computers that are manufactured by the Raspberry Pi Foundation. While the former is equipped with 512 MB of *Random Access Memory (RAM)* and a 1.0 GHz quad-core ARM Cortex-A53 CPU, the used Raspberry Pi 5 has 8 GB RAM and a 2.4 GHz quad-core ARM Cortex-A76 CPU.

With regard to the 5G network, Figure 3 shows that the Raspberry Pi Zero 2 W and the Quectel RM520N-GL 5G modem, which are connected over a USB adapter cable, form the UE. The modem is also equipped with a freely programmable SIM card, which is offered by the Open Cells Project. However, the SIM card is not shown in Figure 3.

In contrast to this, a Raspberry Pi 5 and the Ettus USRP B210 SDR represent a base station within the radio access network. To transport data over the physical air interface between the Quectel 5G modem and the SDR, the 5G parameters, such as the frequency band, the frequency and the bandwidth, have been set in accordance with Figure 3.

Two additional Raspberry Pi 5 are used to set up a 5G core network on hardware layer. The reason for using two of these devices is explained later in this section. To enable the interaction with the radio access network, all three Raspberry Pi 5 are connected to each other via Ethernet cables and a switch, where the latter is not shown in Figure 3.

As a requirement for the transport of LCS-UPP data, a user plane connection between the UE and the core network has to be established first. To realize this, different software systems are used on top of the hardware components, which are shown in Figure 3, to enable the investigation of the LCS-UPP transport time metric. In this context, Table II summarizes the software components of each device, which are crucial for the transport time investigation, and highlights the contributed software of the paper at hand that realizes the transport time measurement according to the approach shown in Figure 2.

Table II shows that the srsRAN software system is deployed on the radio access network side, while the UHD library is used by the Raspberry Pi 5 for the configuration of the SDR to enable the data exchange over the physical air interface. On the core network side, the Open5GS software has been installed. However, Figure 3 and Table II indicate that the core network is divided into a main part and the LMF. This is due to a conflict between the investigated data path implementation types, which are introduced in Section III, and the Open5GS core software. More precisely, the Open5GS implementation

uses a virtual network driver to forward data from the user plane to the target *data network* and vice versa. This network driver is called *tunnel (TUN) device* and causes additional context switches between user and kernel space. Therefore, the LCS-UPP server implementation of the paper at hand has been moved to a separate Raspberry Pi 5. This is particularly important for Type 3, where no context switches may occur.

In contrast to the existing srsRAN and Open5GS solutions, this paper contributes a distributed system, which has been written in the C programming language and consists of a LCS-UPP client and server. While the client realizes the calculation of the transport time  $t_t$ , the server is responsible for processing and responding to incoming UL LCS-UP messages from a UE. More precisely, depending on the data path implementation type, there can be several implementations of client and server, whose source code is publicly available in [12]. For instance, three different TLS user space libraries of Type 1 are investigated, where a corresponding LCS-UPP client and server implementation have been developed for each library. These libraries are summarized in Table III.

According to Table III, the TLS implementation of the wolfSSL and Mbed TLS libraries, which are optimized for resource-constrained devices, based on TCP, while the MsQuic library provides a software solution of the QUIC protocol. Therefore, a Type 1 comparison between TLS user space libraries using different protocol stacks is possible.

TABLE II  
OVERVIEW OF USED SOFTWARE COMPONENTS

Device	Software	Version	Note
RPi Zero 2 W	LCS-UPP client (Linux v6.6.42)	—	contribution of the paper at hand
RM520N-GL	Firmware	AAR03A02M4GA	—
USRP B210 SDR	UHD	4.7.0.0	library to interact with the SDR
RPi 5 (SDR)	srsRAN (Linux v6.8.0)	24.10	5G radio access network software
RPi 5 (main)	Open5GS (Linux v6.8.0)	2.7.2	5G core network software
RPi 5 (LMF)	LCS-UPP server (Linux v6.6.42)	—	contribution of the paper at hand

TABLE III  
OVERVIEW OF USED TLS USER SPACE LIBRARIES (TYPE 1)

Name	TLS base	Language	Version	Reference
wolfSSL	TCP	C	5.7.6	[13]
Mbed TLS	TCP	C	3.6.2	[14]
MsQuic	QUIC	C	2.4.5	[15]

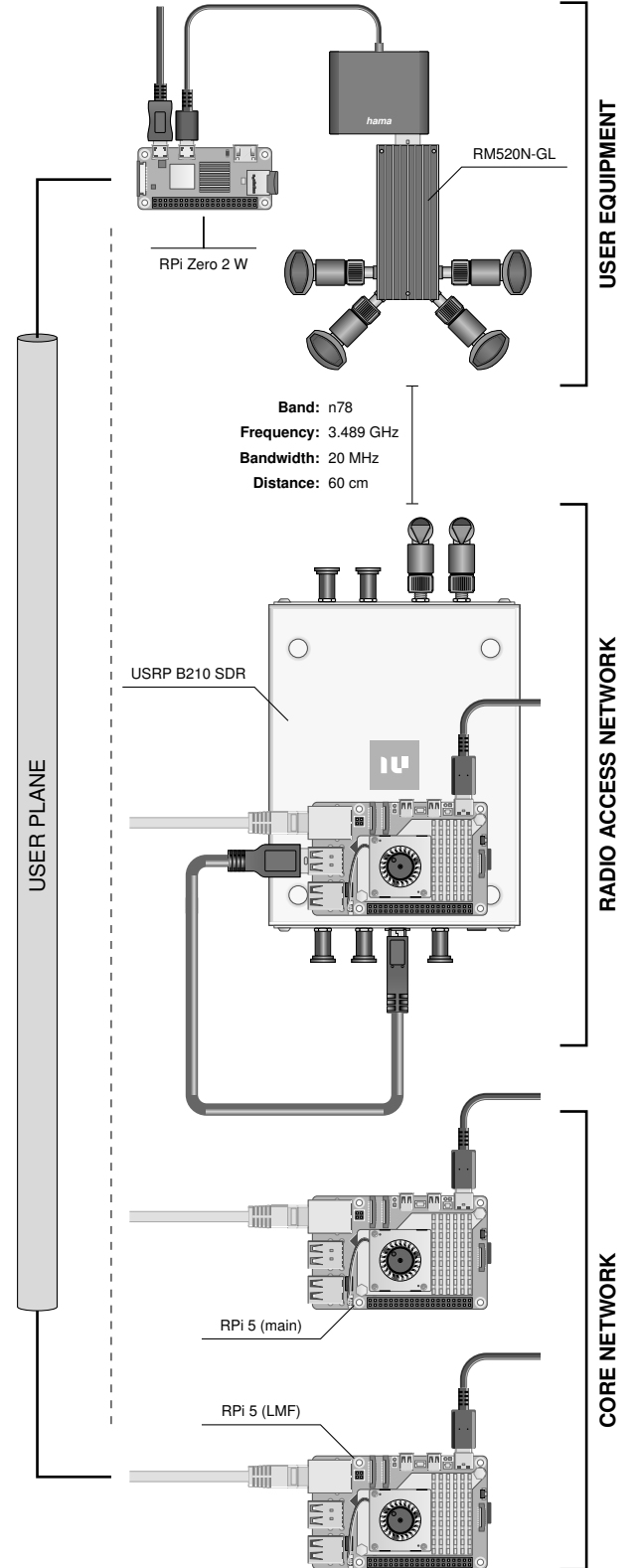


Fig. 3. Top view of the 5G setup for the LCS-UPP transport time analysis: The UE (top) that consists of a Raspberry Pi Zero 2 W and the Quectel RM520N-GL 5G modem sends LCS-UPP data over the user plane to the LMF (bottom), which is represented by a Raspberry Pi 5. The user plane is established by the radio access and the main part of the core network, which based on the Ettus USRP B210 SDR and two Raspberry Pi 5.

To investigate the impact of Type 2 and Type 3 implementations on the transport time  $t_t$ , the TCP-based TLS implementation of the Linux operating system is considered. Due to the fact that the in-kernel TLS feature of Linux is disabled by default, the kernel version 6.6.42 has been compiled from source. As shown in Table II, this modified operating system kernel is installed on the Raspberry Pi Zero 2 W as well as on the Raspberry Pi 5, which are primarily responsible for the LCS-UPP transport time investigation.

In addition to the TLS feature, the real-time capabilities of the Linux operating system are also enabled by patching the kernel 6.6.42 with `RT_PREEMPT`. To reduce the deviation of the transport time  $t_t$  statistically, the real-time behavior of the contributed LCS-UPP implementations of this paper is improved by assigning the following properties:

- ① **Scheduling policy**, which is set to `SCHED_FIFO` to enable a first come, first served scheduling strategy.
- ② **Real-time priority**, which is set to the lowest real-time priority of one to preempt non-real-time tasks.
- ③ **Memory locks**, which are used to prevent the *swapping* of memory segments by the Linux operating system.

The Raspberry Pi Zero 2 W is more resource-constrained than the Raspberry Pi 5 in terms of memory and processing power. In order to address this disadvantage, the CPU governor feature, which is also provided by the Linux operating system, is set to *performance mode* on the Raspberry Pi Zero 2 W to increase its overall processing power.

## V. EVALUATION OF RESULTS

Based on the theoretical approach and the used 5G setup, this section deals with the achieved results to evaluate the transport time  $t_t$  of the LCS-UPP. In accordance with the data path implementation types, the following investigations in relation to the transport time  $t_t$  are addressed by the paper at hand:

- ① general impact of the LPP payload size  $n$ ,
- ② Type 1 comparison between the TLS user space libraries wolfSSL, Mbed TLS and MsQuic,
- ③ comparison between the wolfSSL library (Type 1) and the in-kernel TLS implementation of Linux (Type 2),
- ④ impact of context switches between user and kernel space by comparing Type 3 with Type 1 and Type 2.

In this regard, Figure 4 summarizes the overall results of the transport time analysis. To ensure transparency and accessibility, all collected raw data and their corresponding statistical parameters have been made publicly available in [12].

More precisely, Figure 4 shows two diagrams, which illustrate three 99% confidence intervals of the transport time  $t_t$  for each investigated LPP payload size  $n$ . While the upper diagram compares the transport time between the investigated Type 1 libraries wolfSSL, Mbed TLS and MsQuic, the lower diagram of Figure 4 compares the transport time between all three data path implementation types. Figure 4 also highlights

the percentage change of the upper bound  $\bar{x} + z_{0.005} \cdot \sigma_{\bar{x}}$  of most confidence intervals in each diagram. This change refers to the upper bound of the confidence interval, which based on the same LPP payload size  $n$  and the wolfSSL library.

Both diagrams indicate that the LCS-UPP transport time  $t_t$  largely increases, if the LPP payload size  $n$  also becomes larger. In particular, when the LPP payload size reaches a value of  $2^{14}$  Byte, the transport time increases more rapidly. This behavior could be due to the size of a single *TLS record*, which is also set to  $2^{14}$  Byte by default.

The comparison between different Type 1 implementations in the upper diagram of Figure 4 shows that the TCP-based TLS library Mbed TLS performs slightly worse than wolfSSL. On average, the usage of the Mbed TLS library increases the transport time  $t_t$  by 3.38%. In contrast to this, the investigated QUIC-based TLS library MsQuic can reduce the transport time of LCS-UPP in comparison to wolfSSL by approximately 42.01%. However, if the LPP payload size  $n$  exceeds the TLS record size of  $2^{14}$  Byte, the wolfSSL library performs better again, as shown in the upper diagram of Figure 4.

In the diagram at the bottom of Figure 4, the confidence intervals of the transport time  $t_t$  of each investigated data path implementation type are compared using the wolfSSL library as well as the in-kernel TLS implementation of Linux. In order to investigate the LCS-UPP transport time between the same protocol stacks, which consist of TLS on top of TCP, wolfSSL is used for the Type 1 comparison despite its worse performance compared to the MsQuic library. The results show that the movement of the TLS data path to the underlying Linux operating system increase the LCS-UPP transport time by an average of 13.35%. Therefore, the TCP-based TLS implementation of the Linux operating system performs worse than wolfSSL with respect to the transport time  $t_t$ .

On the other hand, the complete movement of the LCS-UPP data path to the Linux kernel, which prevents the occurrence of context switches between user and kernel space, generally improves the transport time  $t_t$ . As shown in the lower diagram of Figure 4, the used Type 3 implementation can reduce the LCS-UPP transport time by 5.84% on average compared to the usage of the wolfSSL TLS user space library. Additionally, the comparison of the in-kernel TLS implementation of Linux between Type 2 and Type 3 shows that missing context switches affect the transport time  $t_t$  positively. On average, the transport time is improved by 16.95%, if the LCS-UPP data path is completely moved to the Linux kernel. However, this percentage change can not be determined using Figure 4.

## VI. CONCLUSION

This paper presents novel insights into the LCS-UPP and the investigation of its transport time performance. For this purpose, a 5G network has been set up, which based on the Raspberry Pi models 5 and Zero 2 W, the Ettus USRP B210 SDR and the Quectel RM520N-GL 5G modem. Depending on the LPP payload size and different data path implementation types, which primarily address the impact of different TLS 1.3 implementations on the LCS-UPP transport time, the results

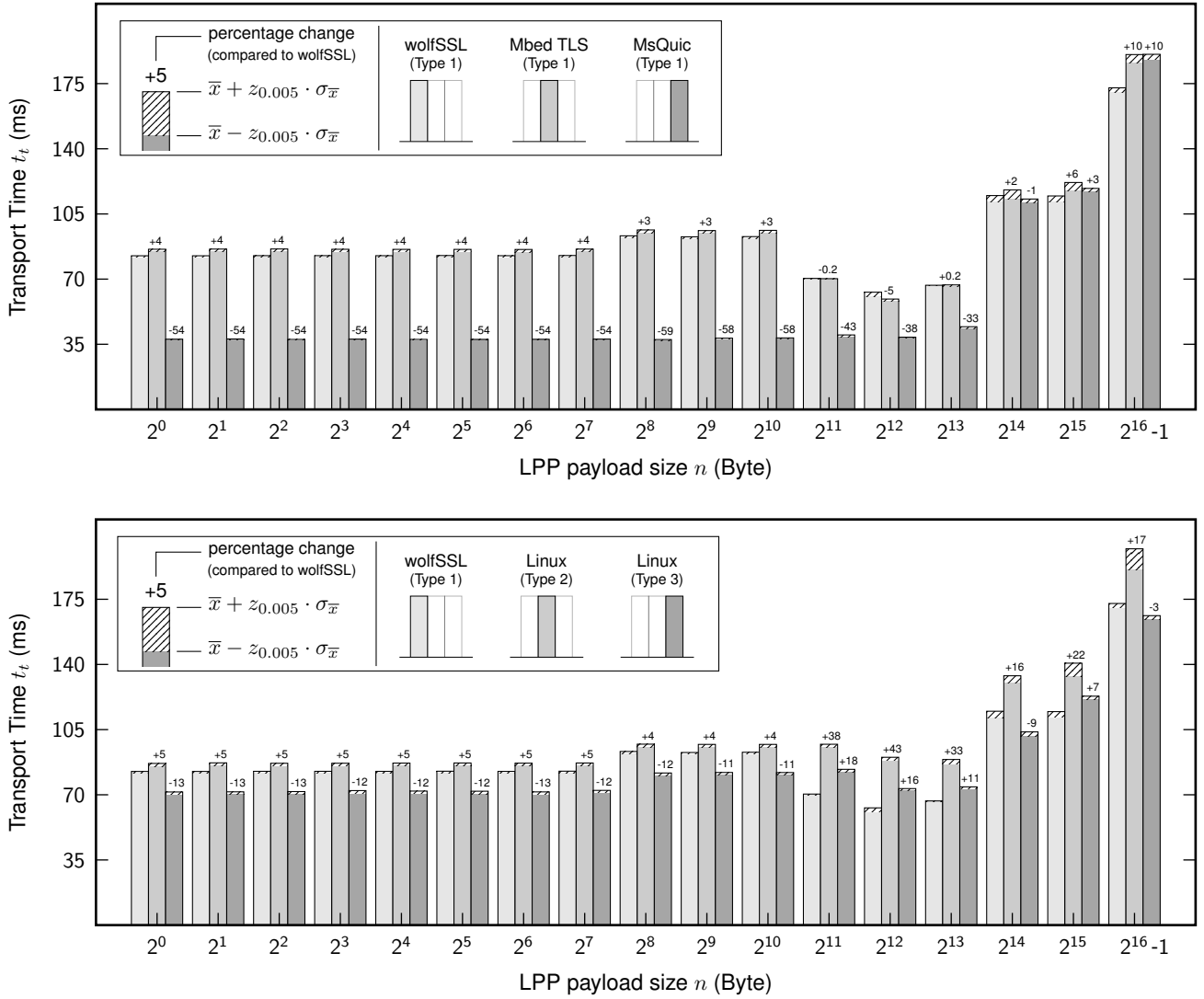


Fig. 4. Impact of the transported LPP payload size  $n$  on a 99% confidence interval of the LCS-UPP transport time  $t_t$  by using different TLS user space library implementations of Type 1 (above) as well as different data path implementation types (below)

show that the transport time increases, if the LPP payload size also becomes larger. In contrast to this, the transport time can be reduced by 42.01% on average, if the QUIC-based MsQuic library is used instead of the TCP-based wolfSSL library. Also the complete movement of the LCS-UPP data path to the used Linux operating system to prevent the occurrence of context switches between user and kernel space decreases the transport time by approximately 5.84% compared to the investigated wolfSSL user space library.

In particular, the usage of more powerful computer hardware on radio access and core network side could generally improve the LCS-UPP transport time. Also further implementation designs could be examined, for instance, using an in-kernel implementation of the QUIC protocol, the DTLS protocol to realize TLS protection based on UDP, or utilizing XDP to bypass the network stack of the Linux operating system to improve LCS-UPP data exchange. In addition, the time

behavior of selected LPP procedures on top of the LCS-UPP could also be part of future work. On the other hand, the usage of a more resource-constrained device than the used Raspberry Pi Zero 2 W on UE side could also be investigated. In the latter case, the potential benefits of employing the approach of the paper at hand warrant further investigation.

#### ACKNOWLEDGMENT

This work has been achieved in the KliNet5G project that is funded by the German Federal Ministry for Economic Affairs and Climate Action under the reference number 01MC22003G. The authors would also like to thank Mr. Patrick Bomball from the University of Wismar, Germany, for his expertise with regard to TLS and its mechanisms.

## REFERENCES

- [1] B. Rother, N. Kalis, C. Haubelt, and F. Golatowski, "Localization in 6G: A Journey along existing Wireless Communication Technologies," in *2024 IEEE 20th International Conference on Factory Communication Systems (WFCS)*, 2024, pp. 1–7. DOI: 10.1109/WFCS60972.2024.10541024.
- [2] 3GPP, "5G System (5GS); User plane Location Services (LCS) protocols and procedures; Stage 3," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 24.572, 2024.
- [3] 3GPP, "LTE Positioning Protocol (LPP)," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 37.355, Dec. 2024.
- [4] 3GPP, "5G System (5GS) Location Services (LCS); Stage 2," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.273, Sep. 2024, pp. 122–126.
- [5] 3GPP, "Security architecture and procedures for 5G System," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 33.501, Jan. 2025, p. 291.
- [6] M. Barbosa, M. Silva, E. Cavalcanti, and K. Dias, "Open-Source 5G Core Platforms: A Low-Cost Solution and Performance Evaluation," Dec. 30, 2024. arXiv: 2412.21162v1 [cs.NI].
- [7] J. E. Håkegård, H. Lundkvist, A. Rauniyar, and P. Morris, "Performance Evaluation of an Open Source Implementation of a 5G Standalone Platform," *IEEE Access*, vol. 12, pp. 25 809–25 819, 2024. DOI: 10.1109/ACCESS.2024.3367120.
- [8] O. Zeidler, J. Sturm, D. Fraunholz, and W. Kellerer, "Performance Evaluation of Transport Layer Security in the 5G Core Control Plane," in *Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '24, Seoul, Republic of Korea: Association for Computing Machinery, 2024, pp. 78–88. DOI: 10.1145/3643833.3656140.
- [9] P. Wang, C. Bianco, J. Riihijärvi, and M. Petrova, "Implementation and Performance Evaluation of the QUIC Protocol in Linux Kernel," in *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWIM '18, Montreal, QC, Canada: Association for Computing Machinery, 2018, pp. 227–234. DOI: 10.1145/3242102.3242106.
- [10] S. Zanini, L. Petrucci, I. Palamà, G. Bianchi, and S. Bartoletti, "Towards End-to-end Implementation of 5G Positioning with Off-the-shelf Devices," in *2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall)*, 2024, pp. 1–6. DOI: 10.1109/VTC2024-Fall63153.2024.10757829.
- [11] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*, 6th ed. John Wiley & Sons, Inc., 2014, pp. 200–201, 241–245, 250, 252, 279, ISBN: 978-1-11-853971-2.
- [12] N. Kalis, *Performance Analysis of 5G User Plane Positioning via LCS-UPP*, version 1.0.0, Zenodo, Mar. 2025. DOI: 10.5281/zenodo.14443056. [Online]. Available: <https://doi.org/10.5281/zenodo.14443056>.
- [13] wolfSSL. "wolfSSL – Embedded SSL/TLS Library." (2025), [Online]. Available: <https://www.wolfssl.com> (visited on 02/16/2025).
- [14] Mbed TLS. "Mbed-TLS/mbedtls." (2025), [Online]. Available: <https://github.com/Mbed-TLS/mbedtls> (visited on 02/16/2025).
- [15] Microsoft. "microsoft/msquic." (2025), [Online]. Available: <https://github.com/microsoft/msquic> (visited on 02/16/2025).