Implementation and evaluation of the Multi-connection Tactile Internet Protocol and API

Delia Rico
ITIS Software
Universidad de Málaga
Málaga, Spain
deliarico@uma.es

Karl-Johan Grinemmo
Computer Science
Karlstad University
Karlstad, Sweden
karl-johan.grinnemo@kau.se

Anna Brunstrom
Computer Science
Karlstad University
Karlstad, Sweden
anna.brunstrom@kau.se

Pedro Merino
ITIS Software
Universidad de Málaga
Málaga, Spain
pmerino@uma.es

NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium | 978-1-6654-0601-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/NOMS54207.2022.9789842

***Abstract*—Tactile Internet defines applications for remotely controlling and manipulating critical devices that require perceived real-time operation with additional demanding requirements like reliability. These use cases with stringent requirements demand adequate transport protocols to take advantage of the underlying possibilities. Traditional transport-layer solutions like TCP and UDP are no longer sufficient, hence novel protocols are being developed to support these applications. In this paper, we present an implementation and evaluation of the Multi-connection Tactile Internet Protocol (MTIP), a transport layer proposal to support these communications. MTIP uses application and network status information to perform an intelligent selection of paths in order to improve reliability and latency. In our evaluations, we study how the different configurations of the MTIP algorithm affect this selection and we see a direct trade-off where, with more restrictive thresholds, MTIP can increase the packets received correctly at the cost of sending extra duplicate packets.**

***Index Terms*—Multi-connectivity, Transport Protocols, Context Awareness, Tactile Internet, API.**

I. Introduction

Tactile Internet use cases, such as the remote operation of industrial machinery or drones, require low levels of latency combined with high levels of reliability [1]. In many cases, the nature of these use cases entails additional limitations, such as the use of a wireless network or the need to use packet-switched networks due to distance constraints. This creates the need for not only a high-quality network to support their operation, but also the use of adequate transport protocols to take advantage of the underlying possibilities.

Traditional protocols like UDP [2] and TCP [3] are not feasible in these use cases, since they can induce high levels of latency or loss, due to their strict data delivery operation or lack of reliability, respectively. Therefore, real-time transport protocols have traditionally been adapted and used for remote control operation. This is the case for protocols such as IRTP [4], ETP [5], HMTP [6], STRON [7] and the Smoothed SCTP [8]. Nevertheless, numerous end-to-end solutions have been developed lately for wireless and cellular networks, such as network awareness, the use of multi-homed devices and private network slices, among other advancements [9], but those real-time transport protocols cannot take advantage of the full potential of these current networks. That is why there has been a growing interest in the research of novel protocols or extensions more aligned with current trends [10].

Examples of these new protocols include MPTCP [11], QUIC [12], MPQUIC [13] and MPRTP [14]. However, even if these protocols are aligned with current trends, they are not directly aimed at complying with the requirements of a tactile Internet. Not least since they lack the flexibility to adapt to the specific requirements of each use case, and come with extra features that are not needed for a tactile Internet and could even be counterproductive due to the added complexity and unnecessary header overhead. The remote operation in a tactile Internet requires a lightweight protocol flexible enough to adapt not only to specific preferences but also to unexpected network conditions. The protocol should aim to maximize the trade-off between latency and reliability with respect to the application requirements in each use case.

With this in mind, we propose the Multi-connection Tactile Internet Protocol (MTIP), a multipath protocol specifically designed for the remote control of a tactile Internet in current and next generation wireless networks, such as 5G and B5G. MTIP aims to offer programmable quality to applications that need a dynamic compromise between latency and reliability, among other parameters. MTIP manages the use of the different paths available according to application preferences to maximize its operation and work on a deadline-aware basis. In this paper, we further elaborate on the initially proposed design of MTIP [15] and present an implementation. Moreover, we evaluate the ability of MTIP to intelligently select network paths in three types of scenarios that we believe cover a broad range of real use cases, and show how configurations with more restrictive thresholds can decrease lost and late packets at the cost of sending extra duplicate packets.

The paper is organized as follows: Section II discusses the design of MTIP and Section III its implementation. Next, Section IV evaluates the ability of MTIP to select network paths to meet the requirements of different tactile applications. The paper concludes in Section V.

II. Description of the Protocol

The Multi-connection Tactile Internet Protocol (MTIP) is a transport protocol for the remote control of tactile Internet applications. MTIP exploits the benefits of having several

different available paths by sending redundant packets in an intelligent manner over multiple paths or sublinks, which together act as a reliable link (see Figure 1).

Fig. 1. Use of multiple sublinks in MTIP.

In order to provide adequate transport services to tactile applications with the least amount of redundant network resources, MTIP uses context awareness to manage and schedule multi-connectivity in a flexible way. The protocol takes into consideration both application preferences through its application programming interface (API), and information about the network status through the collection of network information, in order to select the best sublinks over which to send data. In the remainder of this section, we provide an overview of the MTIP protocol, with an emphasis on its sublink management and the way it uses knowledge about application requirements as well as network conditions to decide on which sublinks to transfer data packets.

A. Message types

The protocol message structure of MTIP is defined in Figure 2 and expands a UDP-like header with some additional fields to keep it small and beneficial for tactile Internet. The timestamp is used to control the deadline of the packets; the sequence number controls the order and identifies duplicates, while the flags differentiate the two types of messages used on MTIP: data and control messages.

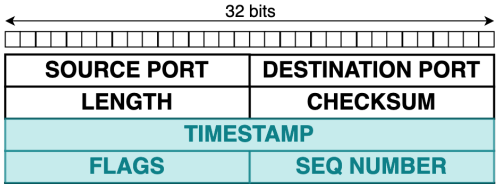

Fig. 2. MTIP header.

Data messages carry application data in the payload. These messages can be sent on one or several sublinks depending on the application preferences, and are processed on the receiver side to filter out late or duplicate packets. In contrast, control messages manage link establishment, as well as additional operations. Messages of the types *Link* and *Preference* are sent to establish the link and share information about the available interfaces on each endpoint, and to synchronize application preferences. Other types of messages include *Keep Alive* messages that are employed to perform periodic measurements on idle sublinks, and messages of the type *Characterization* that are used when more exhaustive measurements are demanded by the application. MTIP uses *Finish* messages to close a communication session. All control and data messages are confirmed by *ACK* messages.

B. Data transmission

MTIP can send data in several ways. It can use all available paths to send duplicated data, it can leave the decision to the application, or it can use an intelligent selection of sublinks, namely the MTIP sending algorithm. Using all paths at the same time could be the most beneficial in terms of maximising some Key Performance Indicators (KPI) such as reliability; however, it would in most cases also result in a waste of network resources. The MTIP sending algorithm is able to adapt to conditions maximising the KPI values and minimising the waste of resources. As shown in Figure 3, it basically makes two decisions i) the amount of sublinks to use and ii) which sublinks to use.

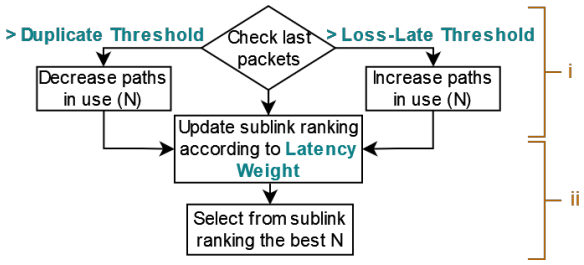

Fig. 3. MTIP sending algorithm.

Initially, MTIP starts sending data packets on all sublinks. If duplicates of too many packets are being received, MTIP reduces the number of sublinks it uses. In contrast, if reducing the number of sublinks results in too many packets arriving late or lost, MTIP increments the number of sublinks in use. This decision is performed on the basis of application preferences, such as the percentage of duplicate, late, and lost packets that are considered acceptable. MTIP decides which sublinks to use for transmission by ranking available sublinks in terms of latency and reliability. The ranking is carried out with respect to the application preference Latency Weight, which weighs in the relative importance of latency and reliability for the particular application in question.

At the receiving side, control packets are directly processed and acknowledged, while data packets go through the MTIP reception algorithm to determine if they should be sent to the application layer or discarded before being acknowledged. In particular, MTIP keeps a window that controls the arrival of packets: Packets that are not deemed obsolete enter the reception window. To control time, MTIP uses a timestamp which is included in the packets' headers. MTIP assumes that both endpoints are synchronized, e.g., by using the Network Time Protocol [16]. MTIP features a mechanism that prevents duplicates from being delivered to upper layers: Duplicate packets share the same sequence number, and when a packet arrives at the receiving side, its sequence number is checked against a list that holds the sequence numbers of already received packets. MTIP also prevents the delivery of out-of-order packets. If a packet arrives out of order, MTIP stores it until the missing packet is received or until it is considered lost. A packet is considered lost when its deadline

has passed. The deadline of a packet is calculated using the timestamps and the application preferences on the maximum latency supported.

C. Context Awareness

MTIP uses information from outside the scope of the protocol layer to offer an enhanced service. This information can be divided in network awareness and application awareness.

Through Keep Alive and Characterization messages, MTIP collects network state information. This information is stored as network KPIs for each sublink (both downlink and uplink). The most relevant KPIs are latency and reliability. Latency is calculated using the timestamp on the packets and current time, while reliability is calculated measuring packet loss through sequence numbers. This information is synchronized on both endpoints through ACK messages.

TABLE I
EXTENSION OF THE SOCKET API FOR MTIP.

Function	Description
socket(), bind(), link()	Create a socket and establish an MTIP link.
send(), receive()	Send and receive data on a selected link.
set_preferences()	Modify preferences and protocol configurations.
get_feedback()	Retrieve information about network status.

MTIP exposes an API similar to the Socket API. However, in contrast to the Socket API, it includes functions to manage network and application awareness. These functions are listed in Table I. Application preferences can be separated into communication preferences and protocol configurations. Communication preferences inform MTIP about application transport-service requirements, e.g., the maximum acceptable latency, the maximum rate of duplicate packets, the maximum permitted lost and late packets, and the factor that tells MTIP how an application weighs the importance of reducing the latency against the importance of keeping duplicate packets and packet loss to a minimum. Protocol configurations have default values and define how MTIP works internally, like the interval between Keep Alive messages, the importance of new measurements over older measurements, the number of relevant packets for measurements, etc. However, they can also be configured by the application.

III. IMPLEMENTATION

In order to test MTIP, we have implemented the MTIP protocol and API using C/C++ and the QT library [17]. The implementation follows the state diagram illustrated in Figure 4 which shows the different states of an MTIP link during its lifetime. An MTIP link starts its life in the CLOSED state, and once it has been established, it enters the LINKED state and the exchange of data may start. At the end of a transport session, the MTIP link is closed, which is done in a similar way to TCP, i.e., via a FIN_WAIT state.

The implementation of MTIP is depicted in Figure 5. When a link has been established and both endpoints have synchronized their information about the sublinks, the data plane uses the different data sublinks (DS) to send the application data

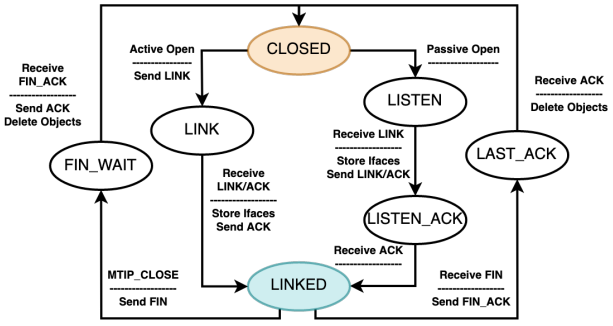

Fig. 4. MTIP state machine.

(following the algorithm presented in Subsection II-B with the application preferences presented in Subsection II-C). The control plane or monitoring entity (MTIMP) uses different ports to monitor the status of the sublinks and sends Keep Alive messages through control sublinks (CS) when the data sublinks are idle, to keep network KPIs up to date.

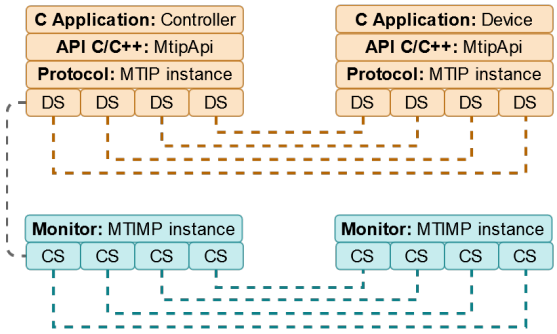

Fig. 5. MTIP implementation structure.

IV. EVALUATION

We study the impact of different MTIP algorithm configurations in different scenarios selected to evaluate to what extent MTIP is able to deliver a requested service when it is challenged with high loss rates, long delays or a mix of both. The topology under test comprises two endpoints connected by four independent sublinks, as shown in Figure 1, where the remote controller is sending packets at the typical tactile Internet rate of 1kHz [1].

The evaluation platform consists of a Ubuntu computer with 16 CPUs and 64GB RAM. We use the Mininet tool [18] to create a virtual topology with differentiated paths, and we apply Netem rules with the tc tool [19] to emulate different network impairments with random packet losses and random delays from a normal distribution. We execute each test 10 times in order to have accurate measurements. In our results, we show the mean values of the executions. Considering that the 95% confidence intervals between executions rarely exceed $\pm 5\%$, and in any case, never affect the tendency of the effect produced by the thresholds under study, we omit them from the figures to avoid cluttering the presentation.

A. Scenarios

We have created three types of scenarios, depending on the KPI impaired:

- **Delay scenarios** emulate delays in the lower layers of the protocol stack. These delays could be caused by different configurations, such as the acknowledged mode of RLC [20], that provides a higher reliability at the cost of delays. Delay scenarios explore the extreme case of delay variation with no losses.
- **Loss scenarios** emulate what could be seen as the opposite scenario, having lower-layer configurations, such as the unacknowledged mode of RLC or aggressive scheduling on guaranteed service slices that reduce the delays at the cost of having some losses. Loss scenarios explore the extreme case of loss variation with no additional delays.
- **Mixed scenarios** are the most realistic of the three types of scenarios. Mixed scenarios emulate when lower layers are configured so as to try to reduce delays and losses and, hence, resulting in a scenario with both delays and losses.

All scenarios change their conditions every second. However, we have created three variants of each scenario: one with good conditions on all the paths, one with dynamic conditions on all the paths (paths that change from good to poor conditions, and vice versa) and one with poor conditions on all paths, in order to showcase an extremely bad situation. To have some comparative values on the worst and best-case scenarios, we have characterized the scenarios using MTIP over just one of the paths and using MTIP redundantly on all paths at the same time. We show the results of this characterization in Table II. It is important to note that the percentage shown in the table is the amount of late or lost packets that do not reach the application layer. In Delay scenarios, this is caused by packets received later than a pre-set typical tactile Internet deadline of 10 ms; in Loss scenarios, the percentage shows actual packet losses; while in Mixed scenarios the percentage is the combination of packets that are late and lost, resulting in a high number of packets that do not reach the application in cases where only one of the paths is used.

TABLE II
CHARACTERIZATION OF THE TESTING SCENARIOS.

	Delay Scenarios			Loss Scenarios			Mixed Scenarios		
	Good	Dyn.	Poor	Good	Dyn.	Poor	Good	Dyn.	Poor
One Path	0.12%	9.72%	16.30%	0.86%	3.89%	10.52%	1.92%	11.58%	25.02%
All Paths	0.01%	0.02%	0.02%	0.00%	0.01%	0.02%	0.07%	0.06%	0.89%

B. Measurements

To evaluate the operation of the MTIP algorithm, we study the impact of its main parameters: the Loss-late Threshold, the Duplicate Threshold and the Latency Weight (as previously presented in Figure 3 of Subsection II-B).

We measure the impact on the main trade-off of the algorithm, the reliability achieved versus the amount of resources wasted in the process. The reliability is measured by counting the amount of lost or late packets, while the waste of resources is measured in terms of the amount of duplicate packets. In order to showcase this trade-off, we have created the metric *trade-off*. This metric makes a direct comparison between duplicates and losses, and is introduced with the sole purpose of making it easier to compare the outcome from different scenarios. Since the rate of duplicate packets is usually higher and less critical than that of losses, the weight is set in a way that lessens its importance.

$$Trade\text{-}off = \frac{Losses + (Duplicates * Weight)}{1 + Weight}$$

In our figures, we show the values of the *trade-off* metric with $Weight = 0.05$. This value helps us compare the lost, late and duplicate packets and see where the metric is lower and the case is more beneficial. However, the decision on the concrete value of the metric would depend on the specific target application and the importance of reliability versus network overload that the developer assigns to that case.

C. Results and discussion

First, we analyze the effect of the modification of the parameters loss-late threshold and duplicate threshold. In Figure 6, the loss-late threshold is modified with a fixed duplicate threshold in the top part, and then the duplicate threshold is modified with a fixed loss-late threshold in the bottom part. The fixed thresholds have a value of 5% to adapt to changes in the network without being too restrictive, in order to be able to see the effect of the other threshold in the different scenarios. It is important to note the different scales on the left side of the graphs to be able to see clearly the variations on both lost-late and duplicate packets.

The graphs suggest that lower values of the loss-late threshold or higher values of the duplicate threshold cause a higher number of duplicates and fewer losses. In these cases, the algorithm thresholds are more restrictive to reduce the number of sublinks and more prone to increase them, causing less losses at the cost of more redundancy. In cases like Delay (poor) with 0% loss-late threshold, the duplicate packets do not increase that much because even if all paths are used, not all packets arrive on time to the other endpoint. In contrast, if the loss-late threshold is set to higher values or the duplicate threshold to lower ones, the results are the opposite, since the thresholds are less restrictive and try to use fewer resources at the cost of some losses. This is seen more clearly in worse scenarios, like the Loss (poor) one, since in fairly good scenarios, the variation on the lost-late packets is minimum.

Thanks to the trade-off metric, we can analyse the trade-off between lost-late and duplicate packets. For instance, in Mixed (poor) both 0% and 5% loss-late threshold configurations are better than the 10% one, because with a 10% loss-late threshold, lost-late packets increase more than the importance we gave to the reduction of duplicates.

The last parameter we study is the latency weight used to rank and select the sublinks. This parameter can be evaluated

Fig. 6. Effect of the Loss-late and Duplicate threshold on the different scenarios.

in Mixed scenarios where the ranking can be established through only using delay measurements, only using reliability measurements or using a weighted value taking into consideration both kinds of measurements. In Figure 7, we set the values of the loss-late and duplicate thresholds to a value of 10% and 90%, respectively, to have a scenario prone to losses, and evaluate the effect of the latency weight parameter on lost-late and duplicate packets. The results show that a selection of paths only on the basis of latency (100%) or a weighted value are generally more beneficial than selection in terms of only reliability (0%). This is due to the fact that reliability is already improved by using multiple paths so, in most cases, a selection based on latency measurements would also include this KPI into the equation.

In fact, we can see that the variation of lost-late packets is minimum. Duplicate packets are more affected since an inappropriate ranking would force the algorithm to use more paths, causing more duplicates but not affecting lost-late packets that much. In Figure 8, we can see why this effect has a larger impact in good scenarios. We show the typical sublinks selected in examples of the different scenarios. In a good scenario, a proper selection would most of the time use only one sublink; in a dynamic scenario the selection would include more sublinks; and in a poor scenario, almost all sublinks will be used at all times. Therefore, in this latter case, the decision on the latency weight parameter to select the concrete paths is not that critical.

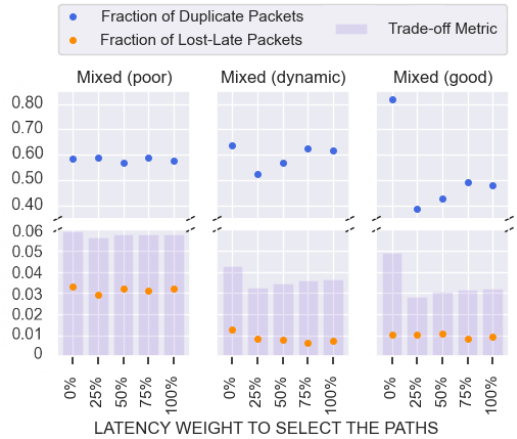

Fig. 7. Effect of the latency weight parameter on lost-late and duplicate packets.

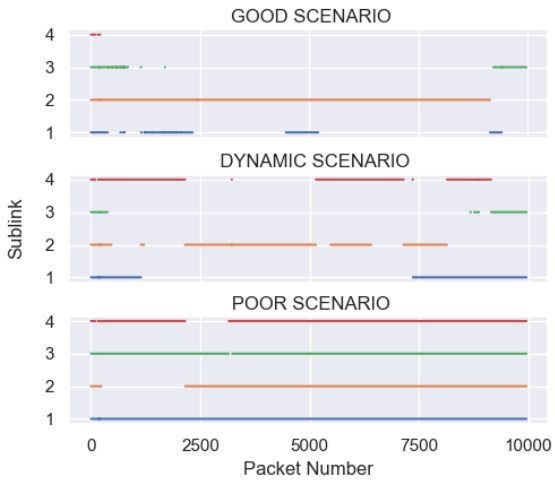

Fig. 8. Sublinks selected in the different scenarios.

V. Conclusion

In this paper we have presented the first implementation and evaluation of the Multi-connection Tactile Internet Protocol and API: MTIP. A transport protocol for tactile Internet applications over wireless networks, with the objective to increase reliability and reduce latency through the selection of the most reliable and fastest paths. We have analyzed the behaviour of the intelligent selection of sublinks under different configurations of the MTIP algorithm parameters and have seen how more restrictive thresholds reduce the amount of lost and late packets and increase the number of duplicates, while less restrictive thresholds do the opposite. Moreover, regarding the concrete selection of sublinks, we have seen that a proper selection could reduce significantly the number of duplicates, especially in fairly good scenarios.

Future work will consist of exploring new more expressive API alternatives that are currently being developed, such as TAPS [21], as well as further evaluation testing the protocol in real scenarios over 5G in the Testbed Morse Lab [22].

Acknowledgment

This work was supported by the EVOLVED5G Project (European Union Horizon 2020) under grant agreement No.101016608 and by the Ministry of Education of Spain through grant FPU17/04292 and RTI2018-099777-B-I00 (the RFOG Project). This work was part of the research internship of Delia Rico in Karlstad University, Sweden, supported by the Ministry of Education of Spain through grants EST19/00930 and EST21/00446.

References

[1] O. Holland et al., "The ieee 1918.1 "tactile internet" standards working group and its standards," *Proceedings of the IEEE*, vol. 107, no. 2, pp. 256–279, 2019.

[2] J. Postel, "User datagram protocol," Internet Requests for Comments, RFC 768, August 1980. [Online]. Available: http://www.rfc-editor.org/rfc/rfc768.txt

[3] J. Postel, "Transmission control protocol," Internet Requests for Comments, RFC 793, September 1981. [Online]. Available: http://www.rfc-editor.org/rfc/rfc793.txt

[4] L. Ping, L. Wenjuan, and S. Zengqi, "Transport layer protocol reconfiguration for network-based robot control system," in *Proceedings. 2005 IEEE Networking, Sensing and Control, 2005.* IEEE, 2005, pp. 1049–1053.

[5] R. Wirz et al., "Efficient transport protocol for networked haptics applications," in *Haptics: Perception, Devices and Scenarios*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 3–12.

[6] A. F. M. Boukerche, H. Maamar, and A. Hossain, "An efficient hybrid multicast transport protocol for collaborative virtual environment with networked haptic," *Multimedia Systems*, vol. 13, pp. 283–296, 2006.

[7] A. Cen et al., "Supermedia transport for teleoperations over overlay networks," in *NETWORKING 2005: Networking Technologies, Services, and Protocols*, ser. Lecture Notes in Computer Science, vol. 3462. Springer, 2005, pp. 1409–1412. [Online]. Available: https://doi.org/10.1007/11422778_126

[8] S. Dodeller, "Transport layer protocols for haptic virtual environments," Ph.D. dissertation, University of Ottawa (Canada), 2004.

[9] D. Rico and P. Merino, "A survey of end-to-end solutions for reliable low-latency communications in 5g networks," *IEEE Access*, vol. 8, pp. 192 808–192 834, 2020.

[10] Networld2020, "The strategic research and innovation agenda: Smart networks in the context of ngi," European Technology Platform, Tech. Rep., September 2020. [Online]. Available: https://www.networld2020.eu/3487-2/

[11] A. Ford et al., "Tcp extensions for multipath operation with multiple addresses," Internet Requests for Comments, RFC 8684, March 2020, http://www.rfc-editor.org/rfc/rfc8684.txt.

[12] J. Iyengar and I. Swett, "Quic: A udp-based secure and reliable transport for http/2," Working Draft, IETF Secretariat, Internet-Draft draft-tsvwg-quic-protocol-00, June 2015. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-tsvwg-quic-protocol-00

[13] T. Viernickel et al., "Multipath quic: A deployable multipath transport protocol," in *2018 IEEE International Conference on Communications*, 2018, pp. 1–7.

[14] V. Singh et al., "Multipath rtp (mprtp)," Working Draft, IETF Secretariat, Internet-Draft draft-singh-avtcore-mprtp-10, November 2014. [Online]. Available: http://www.ietf.org/internet-drafts/draft-singh-avtcore-mprtp-10.txt

[15] D. Rico, M. Gallardo, and P. Merino, "Modeling and verification of the multi-connection tactile internet protocol," in *Proceedings of the 17th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, 2021, pp. 105–114.

[16] D. Mills et al, "Network time protocol version 4," Internet Requests for Comments, RFC 5905, June 2010. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5905.txt

[17] The QT Company, "Cross-platform software development for embedded & desktop," 2021. [Online]. Available: https://www.qt.io/

[18] Mininet Project, "Mininet," 2021. [Online]. Available: http://mininet.org/

[19] The Linux Fundation, "Netem," 2021. [Online]. Available: https://wiki.linuxfoundation.org/networking/netem

[20] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Radio Access Network; NR; Radio Link Control (RLC) protocol specification (Release 16)," 3rd Generation Partnership Project, Tech. Rep. 38.322, 12 2020, version 16.2.0.

[21] T. Pauly et al., "An architecture for transport services," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-taps-arch-10, April 2021. [Online]. Available: https://www.ietf.org/archive/id/draft-ietf-taps-arch-10.txt

[22] Morse Research Group, "Morse lab," http://www.morse.uma.es, 2022, online; accessed 28 Jan 2022.