An Emulation Framework for Haptic Data Transmission Using Real-Time Transport Protocol

https://doi.org/10.3991/ijoe.v19i07.39187

Israa Abdullah(✉), Wrya Monnet
University of Kurdistan Hewlêr, Erbil, Kurdistan, Iraq
israa.nazhat@ukh.edu.krd

Abstract—The Tactile Internet (TI) can be regarded as the next evolution in the world of communication. With its envisioned purpose and potential in shaping up the economy, industry and society, this paradigm aims to bring a new dimension to life by enabling humans to interact with machines remotely and in real-time with haptic and kinesthetic feedback. However, to translate this into reality, Tactile Internet will need to meet the stringent requirements of extremely low latency in conjunction with ultra-high reliability, availability, and security. This poses a challenge on the available communication systems to achieve a round-trip delay within 1 to 10 milliseconds time bound that enables the timely delivery of critical tactile and haptic sensations. This paper aims to evaluate the Real-Time Transport Protocol (RTP) through an emulation framework. It integrates containerization using Linux-based Docker Containers with NS-3 Network Simulator to conceptualize a haptic teleoperation system. The framework is then used to test the protocol's feasibility for delivering texture haptic data between master and slave domains in accordance with the end-to-end delay requirements specified by IEEE 1918.1 standards. The results have shown that the timely provision of haptic data is achievable by obtaining an average round-trip delay of 17.8493 ms from the emulation experiment. As such, the results satisfy the expected IEEE 1918.1 standards constraints for medium-dynamic environment use cases.

Keywords—Tactile Internet, haptic data, RTP, Docker Containers, NS-3

1 Introduction

The wireless communication technology is continuously evolving at an extremely rapid pace. With the emergence of various Internet generations, and the massive efforts to improve and shift the communication network away from traditional methods, doors are opening to a new realm of endless possibilities in communication paradigms and network capabilities in terms of reduced latencies, increased reliability, along with the automation of human tasks. Hence, the need to envision a future where humans convey not only audio and video streams, but also haptic and kinesthetic information (i.e., touch and muscular movement) via the internet to remotely control machines, systems, or robots is becoming of the essence in this era. In light of the unprecedented

circumstances the world has gone through, from the endless lockdowns to travel restrictions and climate disasters and the pressing demands on healthcare infrastructure posed by the coronavirus outbreak [1], an efficient, and reliable remote communication and control to be deployed in such occurrences is essential to mitigate potential risks and challenges.

Generally, human beings are known to be in constant interaction with their environment. Aided by their sensory system, humans are capable of perceiving, manipulating, and adapting to their surrounding environment in a varied manner depending on factors such as the perception level of event, sensory stimulus, and the nature of the event, be it sudden or foreseen. In the case of unforeseen event, the muscles react within 1 millisecond, with an auditory reaction of 100 milliseconds and a visual reaction of 10 milliseconds [2]. This enforces the technological applications with humans in the circle to operate within the aforementioned reaction time ranges. However, if the human is expecting to manually interact with the machine and receive a timely response in real-time, a rapid response in the range of 1 millisecond should be achieved. Yet, it is a challenge ever so critical and cannot be fully met with the existing network infrastructure, as well as the kinematic and haptic sensors.

To accelerate this vision, a new dimension is being ushered into life. A next-generation innovative technology that is standardized in IEEE 1918.1 was suggested by a working group [3], and has been called Tactile Internet [4]. Tactile Internet broadly refers to the real-time transmission, manipulation, control, and delivery of haptic information remotely via the internet, with a delay of [1–10] milliseconds. It is expected to transform the communication paradigm from the sole purpose of delivery of content to skills delivery [5]. With a plethora of critical and non-critical applications and services in which Tactile Internet can be deployed in, from healthcare, Robotics [6], transportation, industry, to education, and entertainment [7], comes the need to address the enabling network technologies. The 5G communication systems can potentially mitigate the challenges posed by the ultra-reliability and low latency requirements of meeting a transmission delay of 1 millisecond and is expected to facilitate a stable human-human, machine-machine and/or human-machine interactions [8]. Yet, the significant strains on current networks persist to be a major hurdle in realizing the Tactile Internet vision and the Quality of Service measures of respective applications.

A successful haptic data transmission and delivery form an integral part of the stability and reliability requirements of haptic communication. Similar to every multimedia stream transmitted in real-time, haptic data are susceptible to network disorders such as delay, jitter, out of order delivery, along with packet loss, thus leading to instable data transmission. In order to optimize the transmission of haptic data, several haptic and non-haptic communication protocols have been proposed for this purpose. These protocols are categorized as either transport or application layer protocols and have the potential of providing an efficient haptic data delivery. Yet, these protocols should possess certain features to be able to meet the stringent Tactile Internet requirements. Such features range from minimum overhead, error correction, synchronized data delivery, prioritization of transmission and provision of key update messages.

For this, the needs arise for researchers to evaluate, and/or to re-evaluate communication protocols or create novel ones in order to facilitate a successful transmission of haptic data. Yet, the availability of extensive analysis and evaluation of available

network protocols' capabilities is insufficient in literature. This is due to the large costs of network equipment and haptic interfaces used to create evaluation testbeds, along with the fact that several haptic protocols are proprietary of their creators, leading to attempts limited to modelling the behavior of those protocols rather than evaluating their real implementations.

In this research work, an IEEE 1918.1 compliant emulation framework based on the use of Docker Containers and NS-3 is proposed to implement a real-time emulation of all upper layers of the communication system. The framework's master and slave domains are represented by two containers on which two C++ based applications utilizing JRTPLIB [9], an open-source, object-oriented RTP library, reside and run. The network domain is simulated in the host computer through NS-3 to help evaluate RTP capabilities in transmitting haptic data through the use of the LENA LTE network module, which is one of the highly supported NS-3 communication technology modules.

The remainder of the paper is organized as follows: Section 2 presents a summary of several relevant works to examine the previous implementations, simulation and emulation frameworks and TI protocols. Section 3 presents the concepts and technologies used for the emulation setup and implementation. Section 4 shows the obtained results, analysis and discussion on the emulation framework shortcomings. Finally, the conclusion and future work are presented in Section 5.

2 Related works

Several networking testbeds, simulation, and emulation tools have been used for various scientific and research purposes, allowing for the timely and cost-effective validation and performance evaluation of algorithms and protocols. Network simulation tools, such as NS-3 [10], OMNeT++ [11], OPNet [12], and NetSim [13] to name a few, provide a flexible, cost-effective, and accurate software solutions that aid in testing various conceptual models and network topologies. Network emulation, on the other hand, enables the execution of real protocols and applications on real hosts that can interact through simulated network environment with focus on network impairments, thus offering results closer to reality with applicable observations to real-world scenarios.

The promising research area of network emulation has driven many researchers to exert their efforts into developing emulation frameworks. This can be seen in the work presented by Ahrenholz, where CORE (Common Open Research Emulator) [14] was introduced as a real-time network emulator tool capable of providing rapid instantiation of hybrid network topologies that consist of both real hardware and virtual network nodes. A lightweight network emulator that is worth mentioning is Mininet [15], which utilizes OS-level container mechanism offered by GNU/Linux kernel to build realistic virtual network applications and functions on one physical machine. This emulator, however, lacks the support to containerized applications such as Docker, and has a limited level of realism in terms of large-scale network modeling.

Authors in [16] leveraged on Docker containers in the design of their emulator framework Kathará, which is presented as a lightweight alternative to virtual machines to aid in the creation of arbitrarily complex network topologies on a single host. In a similar fashion, the authors in [17] presented Megalos, a scalable, and distributed

framework, that allows the emulation of complex network scenarios with several network interfaces by relying on Docker and Kubernetes container-based technologies. Nonetheless, the platform exhibits some limitations inherited from Kubernetes such as timeout and latency issues.

The integration between NS-3 and Docker Containers was successfully accomplished by multiple researchers. This is evident in the works presented by [18] [19] where emulation support of various network scenarios is granted through the use of NS-3 and Docker Containers, with network bridges attached to the virtualized nodes to allow packet transmission. The authors claimed that their Dockemu framework is only limited by the hardware it runs on, yet the work was limited to a single type of node and only two types of scenarios involving CSMA and WiFi channels. To address these limitations, the Dockemu framework was extended by [20], whereby the original version was adapted to cover multiple nodes and more complex IoT scenarios involving the use of LTE and 6lowpan.

With regards to Tactile Internet testbeds, few researchers have contributed to this area. Authors in [21], presented the Tactile Internet eXtensible Testbed (TIXT), which is a generic, modular, and extensible testing environment that complies to the IEEE 1918.1 architecture and standards. The human to machine communication in both virtual and physical environments was highlighted in the work. Yet, the low latency requirements and network impairments were not thoroughly addressed.

Authors in [22] proposed a testbed for Tactile Cyber Physical Systems named as TCPSbed whereby haptic sensory feedback of different modalities are exchanged between the master and the slave, with emphasis on TCPS networking and non-networking latency experiments. Moreover, the support of both NS-3 emulated network along with realistic network topologies and objects are available.

Along similar lines, authors in [23] presented IoTactileSim as a virtual testbed that utilizes Mininet network emulator to create the network topology between the master and slave domains, as well as the tactile support engine. In addition, the CoppeliaSim robotic simulator was utilized to create a virtual teleoperator. The testbed is aimed at investigating the performance of real-time haptic teleoperation in both physical and virtual industrial settings, taking the network impairments such as delay, and packet drop into account to ensure a strict QoS and QoE requirement provisioning.

In the case of haptic communication protocols, numerous researchers developed novel protocols or enhanced existing ones to address the transmission of haptic data. Transport and application layer protocols such as IRTP [24], HoIP [25], Smoothed SCTP [26], ALPHAN [27], PAHCP [28], QUIC [29], UDP [30], RTP [31], and MTIP [32] among others have been adopted for haptic data transmission. Regardless, and to our knowledge, very few efforts were directed to test the protocols' performance in any of the available simulation or emulation platforms except for the work presented by [33] where the performance of HoIP was tested in a conceptual teleoperation case study using NS-3 and the 5G mmWave module. Another instance is seen in the work presented in [34] where an emulation platform based on Linux containers and NS-3 was created in aims of analyzing the performance and feasibility of Multipath TCP protocol in heterogenous network environment of WiFi and 4G LTE.

Driven by aforementioned factors observed in literature, which reflect in the insufficient haptic protocol performance testing within emulated network environments, the

use of simulated behavior rather than real implementations to evaluate the protocols in question, along with the lack of realism associated with the sole use of simulated behavior, the need to create an open-source, scalable, as well as flexible emulation platform that is tailored for evaluating the performance of protocols for haptic data transmission at an application level, with real protocol implementations to allow for the network impairments to be examined, is essential to drive the research wheel for Tactile Internet into further realms.

3 Proposed work

This section offers a general overview on the technologies used to implement the proposed framework, along with the implementation details, and framework setup.

3.1 A technical overview

The proposed framework, which is based on the IEEE 1918.1 reference architecture, aims to mimic a realistic behavior of haptic data transmission in a bi-directional teleoperation system scenario. The emulation framework, as depicted in Figure 1, consists of two Docker Containers, one of which acts as the master domain (operator) and the other as the slave domain (teleoperator). Whereas the network domain in the Linux host is represented by the NS-3 simulation of a Lena LTE 4G network topology which acts as a link between the master and the slave domains. Real RTP packets, generated by a C++ application are transmitted over the simulated network back and forth between the two containers. The packets carry a multimodal texture data obtained from [35] [36], with the 3-dimensional accelerometer sensor recordings represented as X, Y, and Z directions along with the sound recordings represented by the letter S.

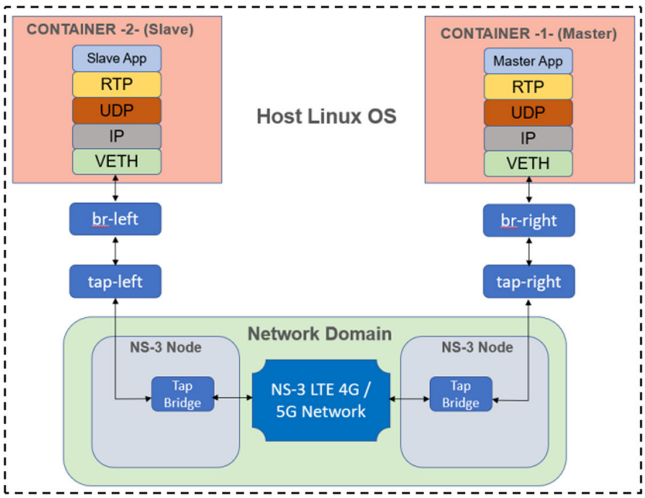


Fig. 1. Architecture of the proposed emulation framework

The following subsections are intended to serve as a background for the technologies used in the emulation testbed.

Network Simulator Version 3 (NS-3). The Network Simulator version 3 (NS-3) is an open source, discrete-event simulator available for research and educational purposes to help facilitate the design, validation, and investigation of various network communication technologies. It is considered as a replacement to its predecessor NS-2 and is built using C++ language with optional Python bindings. As such, it enables users to write their scripts both in C++ and Python to create virtual nodes aided by the various topology Helper classes, that enable the installation of internet stacks, applications and, devices to all created nodes.

NS-3 provides all the needed libraries and protocol stacks for implementing both the 4G and 5G communication networks, as opposed to GNS3 and Mininet whereby only a configured behavior of the communication link exists with no dedicated modules or packages. Additionally, more attention is given to the physical layer in NS-3 as compared with the little or no attention with regards to all other options.

NS-3 is utilized in our proposed work to create the network domain to relay traffic between the master and slave domains.

Containerization. Linux containers (LXC) provide operating system and application-level virtualization environment leveraging on namespaces and control groups to allow running multiple isolated Linux systems on a single Linux host machine. A logical boundary is formed within the same operating system in which an independent root file system, process tree and network subsystem are provided [37]. Unlike the heavy-weight full virtualization of Virtual Machines (VM) where the physical computer is entirely emulated, many advantages are offered by Linux containers. These are summed by: the faster and light-weight nature that enables the creation of multiple container instances, the lower consumption of resources, in addition to the faster boot time, portability, and consistency.

Docker Containers [38] are based on the same concept of Linux containers at its core and are considered as a unified API to manage those kernel-level technologies. Docker enables building containers through the use of portable images across multiple platforms, along with starting, stopping, destroying and deploying applications in containers. Similar to LXC, it leverages on Linux kernel resource allocation and isolation mechanisms such as cgroups, and namespaces. Docker initially used LXC when it was first deployed, until version 0.9 when Docker created a specific driver, known as libcontainer to help access kernel resources. Docker engine uses Dockerfile for creating containers, where a base image is pulled from the Docker hub registry and used to create different image templates. Additionally, it offers cross-platform support and is considered to be easier to configure and scale, with better performance due to layering of Docker container images. However, some of its downsides would be the lack of low-level control similar to the one offered by LXC and risks from malware due to running as root.

Docker containers' strengths, flexibility, and portability in comparison with traditional Linux Containers make it the ideal technology for the proposed emulation platform.

Tap Bridges. Bridging allows two or more network interfaces to forward traffic through the use of MAC addresses tables which are built upon the knowledge it

acquires from learning which host is connected to which network. The interfaces can either be real or virtual, with the virtual ones, such as tap interfaces, appearing like real interfaces to the host computer. They act as the entry and exit points of traffic which passes through the network bridges and can be constructed using the Brctl Linux Package [39]. The tap interfaces on the other hand can be constructed using the Tunctl Linux Package [40].

The Tap Bridge module of NS-3 is used to establish a virtual channel between the tap interface and a node created inside NS-3 for the purpose of exchanging traffic between the NS-3 simulation and the Linux host. Once the traffic flows from the host to the tap interface, the TapBridge transforms the arrived packets into simulated packets. The reverse is true when simulated packets are transformed into real packets through a write operation on the tap interface. This process appears as if a packet has arrived on one of the Linux host's real interfaces.

Real-time Transport Protocol (RTP). RTP [41] is considered as one of the most prominent transport protocols for real-time data transmission, such as the transmission of audio, video, and simulation data. The transport of data is augmented with the Real-Time Control Protocol (RTCP) which works hand in hand with RTP to help monitor data delivery on large multicast networks in aims of providing a reporting mechanism. Statistical and control data are carried within RTCP, including the number of bytes sent, sent and lost packets as well as round-trip delay as feedback on the Quality of Service which RTP provides. RTP is carried and built upon a lightweight version of the User Datagram Protocol (UDP) and as such, it is regarded as a connectionless, best-effort delivery protocol. Furthermore, RTP offers a sequencing system that allows the detection of missing or out-of-order packets and time-stamping information to help determine the interarrival packet time (jitter). This offers many useful features that can aid in the transport of haptic data and calls for additional studies on the protocol properties.

3.2 The emulation setup

The emulation environment setup, depicted in the extended architecture in Figure 3, needs to go through several phases to realize the full framework and as follows:

Docker Container and NS-3 integration. The initial step involves building an Ubuntu image from which the Docker Containers can be created. The definition of the base image along with the utility packages that are needed within the container are collectively specified in the Dockerfile and as shown in Figure 2. Afterwards, the two containers representing the master and slave domains are created. Upon creation, the process IDs needed to identify the containers are stored in variables to be used in subsequent steps. Later on, the left and right bridges, along with the left and right taps are created and connected to each other by adding the tap devices to their respective network bridges. The network namespaces of the two containers are created with the internal virtual ethernet interfaces of each container attached to respective bridges. They are then peered with the containers' external ethernet interface which are assigned with IP addresses that are in the same subnet as its associated nodes of NS-3. Routing information are also specified for each container to allow data transfer between the two containers.

```
1  FROM ubuntu
2
3  RUN apt-get -y update
4  RUN apt-get -y install net-tools
5  RUN apt-get -y install iputils-ping
6  RUN apt-get -y install uml-utilities
7  RUN apt-get -y install bridge-utils
8  RUN apt-get -y install netcat
9  RUN apt-get -y update
10 RUN apt-get -y install gcc-9 g++-9
11 RUN apt-get -y install cmake
12 RUN apt-get -y install vim
```

Fig. 2. Dockerfile utility packages

LTE Network. The NS-3 Lena-LTE-EPC package is deployed for the purpose of simulating the communication network. A typical LTE model is used for the core network, wherein a single base station node B (eNodeB) with a point-to-point connection to the Evolved Packet Core (EPC), a single Packet Data Network Gateway (PGW) node and a single Serving Gateway, along with a single user equipment (UE) node are incorporated. The master container and the UE node combined forming the master domain, whereas the combination of the remote host node and the slave container form the slave domain. However, one observation was made according to our trials and other multiple researchers listed in [34] [20], which confirmed that the Tap Bridge module cannot be connected directly to LTE devices, due to the fact that the only supporting technologies for bridging in NS-3 are CSMA and WiFi. Owing to these facts, the UE and remote host nodes are connected to an NS-3 node using a CSMA channel on which the Tap Bridge is installed.

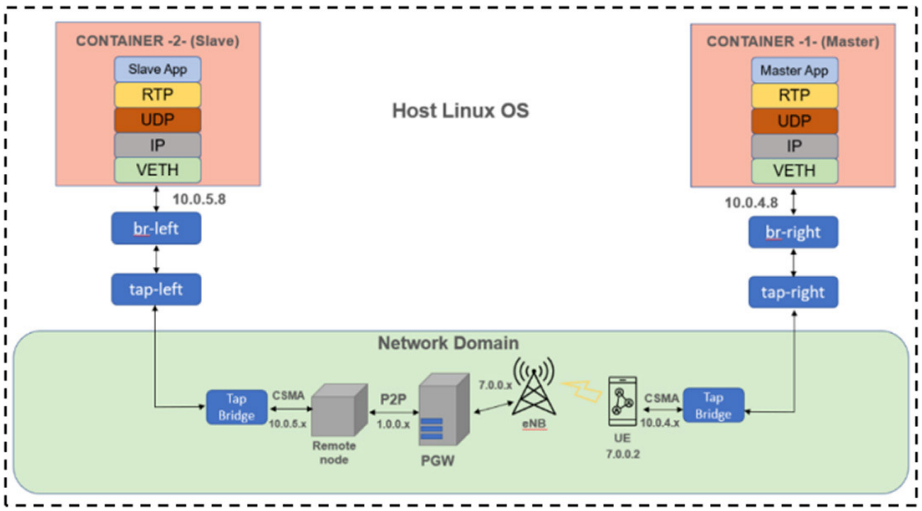


Fig. 3. Extended architecture of the emulation platform

JRTPLIB for RTP-based applications. Haptic data transmission is dependent on a communication protocol residing in both the master and slave containers. After many trials and errors with ccRTP library which was initially chosen for this purpose, the decision to use JRTPLIB [36] came due to its compatibility with Ubuntu 20.04 operating system on which the emulation framework runs on, along with its flexibility, availability of documentation, as well as ease of programming.

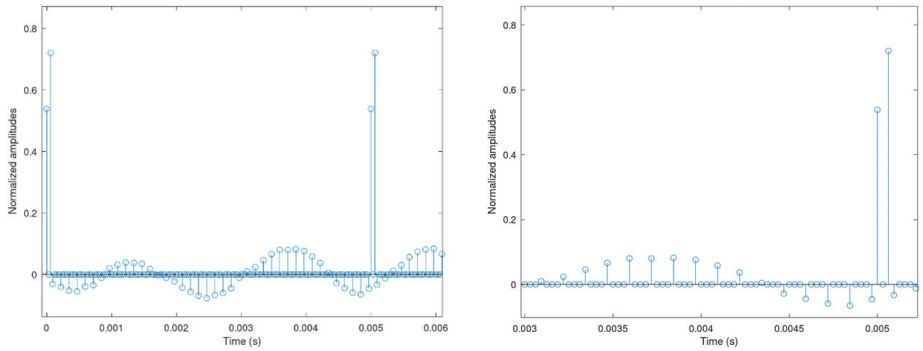
The haptic data were collected from the multimodal texture dataset containing 12 different texture classes [36]. The 3-dimensional accelerometer sensor recordings denoted by X, Y, Z and sound recordings S are collected in four CSV files corresponding to each direction of collected sensory readings and recorded sound. Each line in the X, Y, and Z CSV files corresponds to the readings of one texture class with 4000 samples per line (20 seconds of recording x 200 Hz accelerometer sampling rate), with 16000 samples (20 seconds x 8 kHz) for the sound recordings, all of which are of type float. As such, we have chosen only one of the texture classes. The data were read from the CSV files through a code specific for that purpose and all samples were combined and saved in a vector to be accessed and transmitted by each application.

3.3 The emulation setup

The emulation experiment is conducted on a machine with Intel Core i7-8550U CPU, running at 1.80 GHz with 4 cores, and 8 GB of RAM, with Linux Ubuntu 20.04 LTS OS. The Network Simulator Version 3.34 (NS-3) and JRTPLIB library version 3.11.2.

Two Docker Containers, representing the master and slave domains of the teleoperation case study, are attached and linked to NS-3 via two virtual interfaces and Linux Bridges. NS-3 TapBridge Module is used to allow the exchange of packets from both containers through the simulated network domain, see Figure 2.

Using the JRTPLIB classes, the two RTP based applications on both containers were created to perform the haptic data encapsulation and transmission. First, an instance of the RTPSession class is generated and initialized by means of the Create() function with the appropriate timestamp unit specified using the SetOwnTimestampUnit(). The aim is to send packets containing the multimodal haptic data (3-dimensional displacement and sound data) obtained from our data set [36]. The sampling rate of the data set is 8 kHz for sound and 200 Hz for motion data on X, Y, and Z axis. The data should be transmitted at a rate of 8 kHz, which gives a ratio of 40 audio samples for 1 motion sample in each direction (i.e. 3 haptic samples). For synchronization reason, zero values for motion in X, Y, Z directions are sent with the other 39 audio samples, as depicted in graphs in Figure 4 (a) and (b). The multimodal data are then combined, packetized and sent by specifying the destination IP address and port for data transmission by invoking the AddDestination() function and then the SendPacket() function which takes the data to be sent as its first parameter along with the packet length.



(a) Packetized multimodal data **(b) Zoomed graph for the packetized data**

Fig. 4. A portion of the packetized multimodal data of (X, Y, Z, and S)

As for data reception, data sources are continuously traversed and polled using the PollData() and GotoFirstSourceWithData() of RTPSession class to detect whether data has been received in the RTP session. Once detected, the GetNextPacket() function is utilized to extract received data along with other parameters such as the time of reception, the stream synchronization source identifier (SSRC), length of data received, and so on. For the bilateral communication to be achieved, the code is slightly modified on the slave end, whereby the received packets are echoed back to the master side but with a 1-byte acknowledgement code signal attached to the same packet. Once perceived in the master side, the other packets in sequence are sent in a similar fashion.

The network domain is situated in the middle of the master and slave domains and acts as a bilateral communication medium that interfaces the two domains, whereby texture haptic signals are exchanged, thus closing a global control loop. This exchange is supported by the network domain comprised of 4G LTE network in which the UE and remote nodes are used to relay haptic data traffic back and forth. However, for this two-way communication to be realized, several routing techniques were needed to be used due to the restricting specifications of the EPC's end-to-end IP connectivity that disallows packet transmission with destination IPs different to the one specified for the UE node. Hence, without further routing, the PGW would end up dropping the packet once arrived. For overcoming this issue, the Ipv4ListRouting class in NS-3 was edited, and IP values were modified to allow the UE and the PGW to route/reroute the haptic data packets to their destination containers.

4 Implementation results and discussion

The emulation platform was tested with the previously mentioned teleoperation scenario, where 4000 multimodal samples (equivalent to 0.5 s of real data) that add up to 16000 individual readings. The data were exchanged between the two master/slave containers and the IPv4 network addresses of 10.0.4.8 and 10.0.5.8 were given to the master and slave domains, respectively. Once the emulation experiment is started, the received haptic data and the recorded timestamps were extracted and saved on both

containers within less than 78 timestamp units. As a result, a synchronized delivery in the same packet order was observed on both sides of the master and the slave domains.

The recorded timestamp values obtained from the CurrentTime() function were curated to produce the average results representing the haptic data transmission latency between the master to slave domain, slave to master domain, as well as the round-trip delays. The calculations resulted in an average of 13.2233 ms for master to slave latency, with the highest recorded latency being 21.5 ms and the lowest being 7.8339 ms, as seen in Figure 5 that plots the recorded latency values on the y-axis against the time-stamp unit on the x-axis. It is worth noting that the timestamp unit starts with a random value and is not of seconds rather a sampling frequency or time of the first byte and is used to place the incoming data packets in their correct timing order.

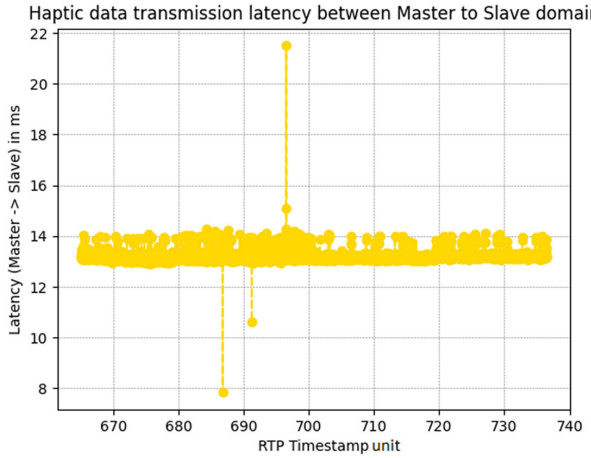


Fig. 5. Haptic data transmission latency between master to slave domain

The histogram depicted in Figure 6 is used to illustrate the frequency distribution of latencies.

Fig. 6. Frequency distribution histogram of master to slave latencies

As for the recorded latencies for the slave to master transmission, an average of 4.5905 ms was recorded. It can be seen that the average latency, in this case, is well within the order of 1–10 ms, with 3.6399 as the lowest latency and 13.6399 as the highest. Figure 7 below indicates the plotted latency values against their timestamps.

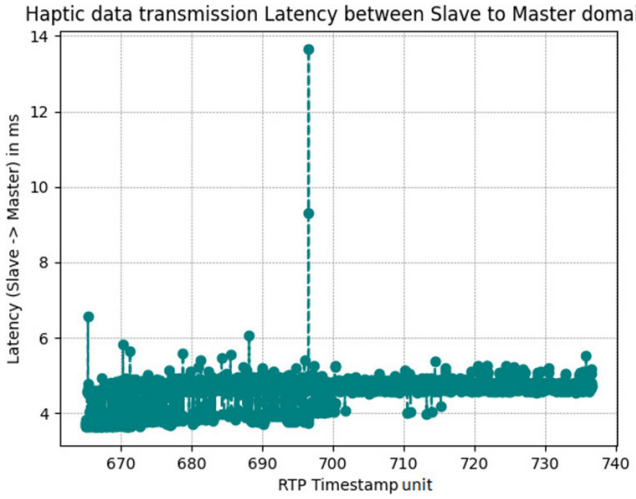


Fig. 7. Haptic data transmission latency between slave to master domain

Similarly, the histogram shown in Figure 8 is used to illustrate the frequency distribution of latencies recorded between slave to master domain.

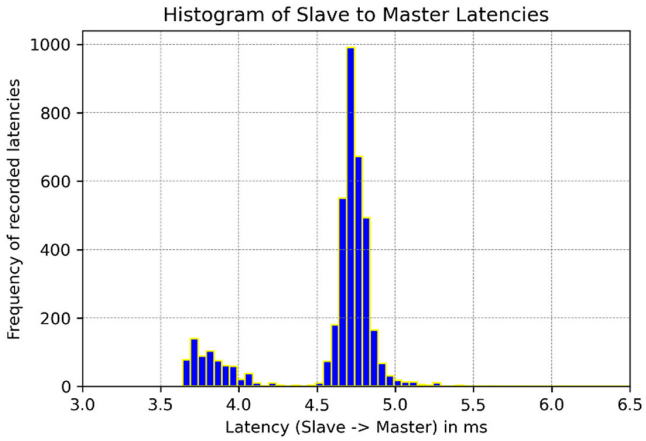


Fig. 8. Frequency distribution histogram of slave to master latencies

As for the round-trip delay, an average of 17.8493 was recorded, with one peak latency standing at 35.18 ms and 12.3798 ms as the lowest, as shown in the plot in Figure 9.

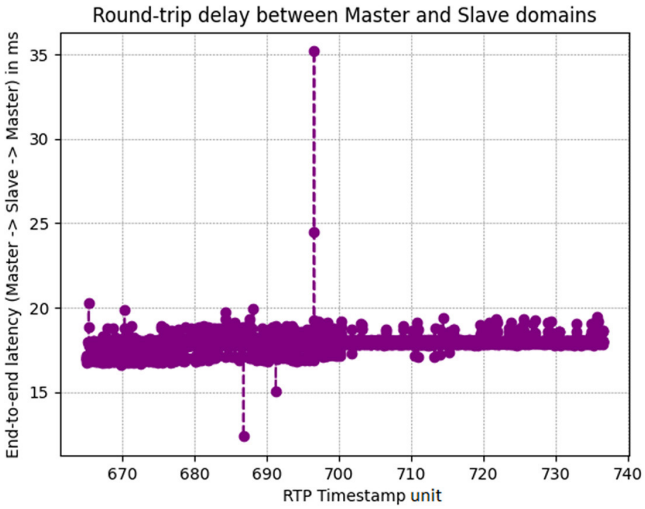


Fig. 9. Round-trip delay between master and slave domains

In a similar fashion, Figure 10 shows the frequency distribution of round-trip latencies between master and slave domains.

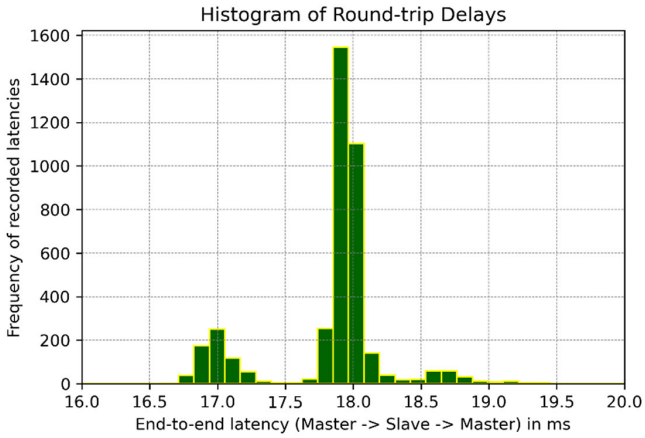


Fig. 10. Frequency distribution histogram of round-trip delay between master and slave domains

As such, the results of the experiments have shown that the RTP-enabling applications have successfully transmitted all the haptic data packets within the context of the Docker container-emulated teleoperation case study. The synchronized arrival of transmitted haptic data on both ends is a noteworthy result, as it validates the RTP protocol capabilities in successfully transmitting haptic data. In addition, the RTP processing time taken between each reception of haptic data and feedback sending is very minimal and is as low as an average of 0.04 ms.

Although the average round-trip latency values were above the highly dynamic and extremely time-critical latency requirement of [1–10] ms specified by the IEEE 1918.1 standards working group, they, nevertheless, fall into the category of medium-dynamic environment teleoperation, which this study targets, where the latency requirement is extended to a range between [10–100] ms for master to slave traffic direction by IEEE 1918.1 standard, in which the emulation framework recorded an average latency of 13.2232 ms. With regards to the average latencies of slave to master traffic direction, the emulation framework recorded an average of 4.5904 ms, which is well within the [1–10] ms threshold.

The inability to achieve a lower round-trip delay could be attributed to various factors, with the main factor being the communication network forward or uplink delay. This was concluded from the UlPdcpStats trace file generated by the NS-3 simulation code, whereby an average uplink delay of 12.14619 ms was noted, as opposed to an average of 3.8602 ms for the downlink delay. LTE's uplink delays are associated with the fact that the uplink transmission uses the single-carrier frequency division multiple access (SC-FDMA), which is a pre-coded version of Orthogonal Frequency Division Modulation (OFDM) that is adopted in the downlink transmission. Although SC-FDMA solves OFDM's undesirable high Peak to Average Power Ratio (PAPR), this comes at a trade-off of resource contiguity enforced in resource block allocation to the UE which adds a problem to eNodeB's packet scheduling and thus introduces additional delays related to sending the scheduling requests, as well as processing delays.

Besides the above, another factor could be the additional CSMA hops on both ends of the simulation network which are used to connect the containers through the TapBridge module to NS-3. These CSMA hops could result in an additional delay for each relayed packet, however it is minimal in comparison with the previously discussed factor. Along similar lines, the real-time scheduler in NS-3 could be a bottleneck due to its attempts at providing accurate time synchronization between the simulation clock and real-time clock, thus creating overhead on the CPU, where only one core out of its available four is utilized due to NS-3's single-threaded design. To mitigate these issues, scheduling algorithms and techniques can be leveraged to reduce the delays at the LTE uplink channel. Latency values can be significantly reduced using 5G as the core network. however, due to the encountered routing issues with the mmWave module during initial tests, test results could not be obtained and none of the packets from both containers managed to pass to the NS-3 core network regardless of the static routing techniques used in the NS-3 code as well as on both containers. The mmWave module, as a matter of fact, is an extension to the NS-3 simulator and support to containerized emulation platforms could still be a work in progress. Nonetheless, the feasibility of using RTP to transmit haptic data has been tested and verified with satisfactory results within the scope of medium-dynamic teleoperation environment.

5 Conclusion and future work

In this research paper, we have presented an emulation framework aimed at analyzing Real-time Transport Protocol's capabilities in transmitting haptic data. The framework leveraged on Docker Containers, JRTPLIB and the real-time support offered by NS-3

to create a teleoperation use case whereby texture haptic data are bilaterally relayed between the master and slave domains with a core network based on LENA 4G LTE.

The technologies involved in the framework, the emulation setup along with implementation scheme were thoroughly described within the paper. In addition, a general overview of the working mechanism of the C++ RTP-enabling applications was also provided.

The teleoperation case study was investigated, and results were obtained and analyzed in terms of end-to-end latency variations between the master to slave as well as the slave to master streams. The capabilities of the emulation platform were validated through the successful transmission of haptic data from master to slave domain and back in a synchronized fashion with a round-trip delay of 17.8493 ms. The obtained round-trip delay values fall in the acceptable range within the medium-dynamic environment teleoperation use cases thresholds specified by the IEEE 1918.1 Tactile Internet standards.

Lastly, the limitations encountered during the framework testing and deployment were highlighted and discussed. As a future direction to this research, additional techniques and solutions can be used to mitigate the network impairments. This can be achieved by exploring additional 5G modules and investigating their feasibility of working in emulation mode with the module developers, to alleviate the routing issues that restricted the network upgrade. The 5G LENA module is a potential candidate to be explored as it enables the operation in mmWave bands.

In addition, a future direction is to upgrade the emulation framework setup by incorporating network noise and cross-traffic conditions from multiple containerized sources. As such, multilateral teleoperation and the fusion of multiple haptic transmission sources can also be explored. Furthermore, and given the need for the provisioning of low-latency and high-reliability communication, path optimization algorithms [42] can be utilized to optimize the communication pathways between different tactile nodes within the network and help improve the performance of tactile internet applications.

Regardless of the challenges, RTP demonstrated promising results that open doors for further investigations on the protocol's performance and capabilities in transmitting additional types of haptic data in different use case scenarios and environments.

6 References

[1] M. Sengupta, A. Roy, A. Ganguly, K. Baishya, S. Chakrabarti and I. Mukhopadhyay, "Challenges encountered by healthcare providers in COVID-19 times: An exploratory study," *Journal of Health Management,* vol. 23, no. 2, pp. 339–356, 2021. https://doi.org/10.1177/09720634211011695

[2] "The Tactile Internet, ITU-T Technology Watch Report," ITU, 2014.

[3] O. Holland, E. Steinbach, R. V. Prasad, Q. Liu, Z. Dawy, A. Aijaz, N. Pappas, K. Chandra, V. S. Rao, S. Oteafy, M. Eid, M. Luden, A. Bhardwaj, X. Liu, J. Sachs and J. Araujo, "The IEEE 1918.1 "Tactile internet" standards working group and its standards," *Proceedings of the IEEE,* vol. 107, no. 2, pp. 256–279, 2019. https://doi.org/10.1109/JPROC.2018.2885541

[4] G. P. Fettweis, "The Tactile Internet: Applications and Challenges," *IEEE Vehicular Technology Magazine,* vol. 9, no. 1, pp. 64–70, 2014. https://doi.org/10.1109/MVT.2013.2295069

[5] M. Simsek, A. Aijaz, M. Dohler, J. Sachs and G. Fettweis, "5G-Enabled Tactile Internet," *IEEE Journal on Selected Areas in Communications,* vol. 34, no. 3, pp. 460–473, 2016. https://doi.org/10.1109/JSAC.2016.2525398

[6] R. Gupta, S. Tanwar, S. Tyagi and N. Kumar, "Tactile internet and its applications in 5G era: A comprehensive review," *International Journal of Communication Systems,* vol. 32, no. 14, 2019. https://doi.org/10.1002/dac.3981

[7] A. Aijaz, "Towards 5G-enabled Tactile Internet: Radio resource allocation for haptic communications," *2016 IEEE Wireless Communications and Networking Conference,* pp. 1–6, 2016. https://doi.org/10.1109/WCNC.2016.7564661

[8] J. Belani, Y. Thakore, N. Kotak, K. Borisagar and B. Sedani, "Performance analysis of various 5G Mobile Architectures," *International Journal of Interactive Mobile Technologies (iJIM),* vol. 16, no. 8, pp. 94–106, 2022. https://doi.org/10.3991/ijim.v16i08.29003

[9] "j0r1's page | CS/Jrtplib," 2022. [Online]. Available: https://research.edm.uhasselt.be/jori/page/CS/Jrtplib.html.

[10] Nsnam, "ns-3," 2022. [Online]. Available: https://www.nsnam.org/.

[11] OMNeT++, "OMNeT++ Discrete Event Simulator," 2022. [Online]. Available: https://omnetpp.org/.

[12] OPNET Network Simulator, "Opnet Projects," 2022. [Online]. Available: https://opnetprojects.com/opnet-network-simulator.

[13] Tectos, "Tetcos: NetSim – Network Simulation Software, India," 2022. [Online]. Available: https://www.tetcos.com/.

[14] J. Ahrenholz, C. Danilov, T. R. Henderson and J. H. Kim, "Core: A real-time network emulator," *MILCOM 2008 – 2008 IEEE Military Communications Conference,* pp. 1–7, 2008. https://doi.org/10.1109/MILCOM.2008.4753614

[15] Z. Xiang and P. Seeling, "Mininet: An instant virtual network on your computer," *Computing in Communication Networks,* pp. 219–230, 2020. https://doi.org/10.1016/B978-0-12-820488-7.00025-6

[16] M. Scazzariello, L. Ariemma and T. Caiazzi, "Kathará: A Lightweight Network Emulation System," *NOMS 2020 – 2020 IEEE/IFIP Network Operations and Management Symposium,* pp. 1–2, 2020. https://doi.org/10.1109/NOMS47738.2020.9110351

[17] M. Scazzariello, L. Ariemma, G. D. Battista and M. Patrignani, "Megalos: A Scalable Architecture for the Virtualization of Network Scenarios," *NOMS 2020 – 2020 IEEE/IFIP Network Operations and Management Symposium,* pp. 1–7, 2020. https://doi.org/10.1109/NOMS47738.2020.9110288

[18] M. A. To, M. Cano and P. Biba, "DOCKEMU – A Network Emulation Tool," *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops,* pp. 593–598, 2015.

[19] E. Petersen and M. Antonio To, "DockSDN: A hybrid container-based software-defined networking emulation tool," *International Journal of Network Management,* vol. 32, no. 2, 2021. https://doi.org/10.1002/nem.2166

[20] A. R. Portabales and M. L. Nores, "Dockemu: An IoT Simulation Framework Based on Linux Containers and the ns-3 Network Simulator – Application to CoAP IoT Scenarios," *Advances in Intelligent Systems and Computing,* pp. 54–82, 2019. https://doi.org/10.1007/978-3-030-35944-7_4

[21] V. Gokhale, K. Kroep, V. S. Rao, J. Verburg and R. Yechangunja, "TIXT: An Extensible Testbed for Tactile Internet Communication," *IEEE Internet of Things Magazine,* vol. 3, no. 1, pp. 32–37, 2020. https://doi.org/10.1109/IOTM.0001.1900075

[22] K. Polachan, J. Pal, C. Singh, T. V. Prabhakar and F. A. Kuipers, "TCPSbed: A Modular Testbed for Tactile Internet-Based Cyber-Physical Systems," *IEEE/ACM Transactions on Networking,* vol. 30, no. 2, pp. 796–811, 2022. https://doi.org/10.1109/TNET.2021.3124767

[23] M. Zubair Islam, Shahzad, R. Ali, A. Haider and H. Kim, "IoTactileSim: A virtual testbed for tactile industrial Internet of Things services," *Sensors,* vol. 21, no. 24, p. 8363, 2021. https://doi.org/10.3390/s21248363

[24] Ping Li, Wenjuan Lu and Zengqi Sun, "Transport layer protocol reconfiguration for network-based robot control system," *Proceedings. 2005 IEEE Networking, Sensing and Control, 2005,* pp. 1049–1053, 2005.

[25] V. Gokhale, O. Dabeer and S. Chaudhuri, "HoIP: Haptics over Internet Protocol," *2013 IEEE International Symposium on Haptic Audio Visual Environments and Games (HAVE),* pp. 45–50, 2013. https://doi.org/10.1109/HAVE.2013.6679609

[26] S. Dodeller and N. D. Georganas, "Transport layer protocols for telehaptics update message," *Proc. 22nd Biennial Symposium on Communications, Queen's Univeristy, Canada,* 2004.

[27] H. A. Osman, M. Eid and A. El Saddik, "Evaluating ALPHAN: A Communication Protocol for Haptic Interaction," *2008 Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems,* pp. 361–366, 2008. https://doi.org/10.1109/HAPTICS.2008.4479972

[28] Q. Nasir and E. Khalil, "Perception based adaptive haptic communication protocol (PAHCP)," *2012 International Conference on Computer Systems and Industrial Informatics,* pp. 1–6, 2012. https://doi.org/10.1109/ICCSII.2012.6454426

[29] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang and Z. Shi, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," *Proceedings of the Conference of the ACM Special Interest Group on Data Communication,* p. 183–196, 2017. https://doi.org/10.1145/3098822.3098842

[30] J. Postel, "RFC 768 – User Datagram Protocol (UDP) – IETF," 1980. [Online]. Available: https://www.ietf.org/rfc/rfc768.txt. https://doi.org/10.17487/rfc0768

[31] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, "IETF RFC 3550 – RTP: A Transport Protocol for Real-Time Applications," 2003. [Online]. Available: https://www.ietf.org/rfc/rfc3550.txt. https://doi.org/10.17487/rfc3550

[32] D. Rico, M.-d.-M. Gallardo and P. Merino, "Modeling and verification of the Multi-connection Tactile Internet Protocol," *Proceedings of the 17th ACM Symposium on QoS and Security for Wireless and Mobile Networks,* p. 105–114, 2021. https://doi.org/10.1145/3479242.3487328

[33] T. Ali-Yahya, W. Monnet and B. M. Amin, "HoIP over 5G for Tactile Internet Teleoperation Application," in *The Tactile Internet,* 2021, pp. 173–207. https://doi.org/10.1002/9781119881087.ch9

[34] V. S. Hapanchak and A. D. Costa, "Emulation of multipath solutions in heterogeneous wireless networks over Ns-3 platform," *Simulation Tools and Techniques,* pp. 3–18, 2022. https://doi.org/10.1007/978-3-030-97124-3_1

[35] O. Kursun and A. Patooghy, "An embedded system for collection and real-time classification of a tactile dataset," *IEEE Access,* vol. 8, pp. 97462–97473, 2020. https://doi.org/10.1109/ACCESS.2020.2996576

[36] O. Kursun and A. Patooghy, "VibTac-12: Texture dataset collected by tactile sensors," *IEEE Dataport,* 2020.

[37] S. M. Jain, Linux containers and virtualization, 2020. https://doi.org/10.1007/978-1-4842-6283-2

[38] D. Merkel, "Docker: Lightweight Linux containers for consistent development and deployment," *Linux Journal,* vol. 2014, no. 239, 2014.

[39] "Brctl(8) – linux man page," 2022. [Online]. Available: https://linux.die.net/man/8/brctl.

[40] "TUNCTL(8) – linux man page," 2022. [Online]. Available: https://linux.die.net/man/8/tunctl.

[41] "RFC 3550 – RTP: A transport protocol for real-time applications," 2022. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc3550.

[42] I. I. Hamad and M. S. Hasan, "A review: On using ACO based hybrid algorithms for path planning of Multi-Mobile Robotics," *International Journal of Interactive Mobile Technologies (iJIM),* vol. 14, no. 18, pp. 146–156, 2020. https://doi.org/10.3991/ijim.v14i18.16371

7 Authors

Israa Abdullah is an Adjunct Laboratory Assistant of the Computer Science and Engineering department at the University of Kurdistan Hewlêr. She completed her Master's degree in Computer Systems Engineering with honors from the University of Kurdistan Hewlêr in 2022, and has been involved in research and teaching activities for the past six years. Her research interests include robotics, computer networks, Internet of Things (IoT) and Tactile Internet (TI).

Wrya Monnet is an Associate Professor and faculty member of the Computer Science and Engineering department at the University of Kurdistan Hewlêr in the Kurdistan Region of Iraq, since 2012. He was the chair of the department for two years, 2014–2016, and has accumulated 23 years of experience in industry and academia. He obtained his PhD at the University of Nice-Sophia Antipolis, France, in 2001, followed by a postdoc at Telecom SudParis. Later, and for 10 years, he worked as an R&D engineer on signal processing dominated projects and as an embedded software engineer consultant for many technological companies in France, such as Airbus and Sagem. Since 2016, he has supervised tens of master's students. Currently, his research is directed towards WSN, IoT, IIoT, and the Tactile Internet.

Article submitted 2023-02-26. Resubmitted 2023-04-16. Final acceptance 2023-04-16. Final version published as submitted by the authors.