Delft University of Technology

Plug-and-Play Haptic Interaction for Tactile Internet Based on WebRTC

Iiyoshi, Ken; Gebremedhin, Ruth; Gokhale, Vineet; Eid, Mohamad

Citation (APA)
Iiyoshi, K., Gebremedhin, R., Gokhale, V., & Eid, M. (2021). Plug-and-Play Haptic Interaction for Tactile Internet Based on WebRTC. In N. Shaghaghi, F. Lamberti, B. Beams, R. Shariatmadari, & A. Amer (Eds.), *Intelligent Technologies for Interactive Entertainment - 12th EAI International Conference, INTETAIN 2020, Proceedings* (pp. 65-81). (Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST; Vol. 377). Springer. https://doi.org/10.1007/978-3-030-76426-5_5

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Plug-and-Play Haptic Interaction
for Tactile Internet Based on WebRTC

Ken Iiyoshi[1] , Ruth Gebremedhin[1] , Vineet Gokhale[2] ,
and Mohamad Eid[1(✉)]

[1] New York University Abu Dhabi, Abu Dhabi, UAE
{ki573,rgg282,mohamad.eid}@nyu.edu
[2] Delft University of Technology, Delft, Netherlands
V.Gokhale@tudelft.nl

Abstract. *Tactile Internet* promises a widespread adoption of haptic communication over the Internet. However, as haptic technologies are becoming more diversified and available than ever, the need has arisen for a plug-and-play (PnP) haptic communication over a computer network. This paper presents a system for enabling PnP communication of heterogeneous haptic interfaces. The system is based on three key features: (i) a haptic metadata to make haptic interfaces self-descriptive, (ii) a handshake protocol to automatically exchange haptic metadata between two communicating devices, and (iii) a multimodal (haptic-audio-visual) media communication protocol. Implemented using WebRTC, the PnP communication is evaluated using a Tele-Writing application with two heterogeneous haptic interfaces, namely Geomagic Touch and Novint Falcon. Our findings demonstrate the potential of the system to be employed in any Tactile Internet scenario.

Keywords: Tactile Internet (TI) · Haptic-Audio-Visual (HAV) handshake · TI Metadata (TIM) · WebRTC · Request/response

1 Introduction

Tactile Internet (TI) [16] – deemed as the *Internet of Skills* [12] – is anticipated to redefine the nature of human interactions with remote environments. TI extends the human capability to effectively control and manipulate remote environments by providing haptic (touch) experience in an ultra-responsive and ultra-reliable fashion [6]. This enables humans to experience remote environments as if they are located there. This has opened up a world of new opportunities with potential to impact every aspect of human lives [18]. Some examples include telesurgery [7], remote disaster management, online shopping [31], gaming and entertainment, and long-distance inter-personal communication.

The inclusion of haptic media as an integral element of TI presents several challenging communication requirements that are unique to TI, and hence need to be separately addressed. For example, design of schemes for robust control

and communication, inter-media (haptic, audio, and video) as well as intra-media (haptic sensors and actuators) synchronization, and so on. To address these challenges, IEEE established the IEEE P1918.1 TI standards Working Group (WG) [18] for defining a standard framework encompassing a generic TI reference model and architecture. It also aims to standardize the interconnections between multitude of interfaces featured in the framework. Further, in order to identify and standardize the TI modules specific to haptic communication, a sub-WG – P1918.1.1 – has been created. This has spawned a string of activities with specific focus on design and development of:

1. *haptic codecs* for perception-based haptic signal compression
2. *plug-and-play (PnP) communication* for interoperability under heterogeneous environment settings.

   While the former has witnessed significant progress, the latter is still in its nascent stages of development.

Design Challenges: Designing a PnP communication system for TI comes with several challenges. We list the most important ones here.

1. **Application-level heterogeneity**: TI applications manifest a diverse range of requirements. For instance, while in a haptic-based VR game a single point device with 3 Degrees of Freedom (DoF) suffices, complex interactions, such as a telesurgery, require several sensors and actuators possibly with heterogeneous Quality of Service (QoS) requirements. A PnP communication system should be capable of detecting these interfaces and start communication on the fly with zero or minimal configurations.
2. **Interoperability**: PnP communication system for TI should be cross-platform and work without requiring installation of any software/firmware.
3. **Quality of Service**: TI applications demand extremely stringent QoS requirements, such as a round trip time (RTT) 10 ms [16]. PnP communication system should be capable of strictly complying to such requirements.

   In this work, we attempt to fuel this direction of advancement in TI by proposing a WebRTC-based system for enabling PnP haptic communication for TI interactions. Our contributions in this paper are the following:

1. We propose a system for haptic interaction between multiple TI nodes in a PnP fashion. Our system is robust to the characteristics of applications and the haptic interfaces used.
2. We present the design details of our PnP communication system developed using WebRTC API. The efficacy of our design lies in the fact that we achieve haptic communication by leveraging only the built-in features of WebRTC, thereby posing no demands for modifications to the standard WebRTC structure.
3. We test the proof-of-concept of the proposed system through a tele-writing application using both homogeneous and heterogeneous haptic interfaces connected via a real-network. The latency measurements of our system demonstrate its potential to be employed in any TI scenario.

The remainder of the paper is organized as follows. In Section 2, we present a review of the related literature. In Sects. 3, 4, we present an overview of our PnP system and discuss its implementation details, respectively. We present our experimental results and discussions in Sect. 5. Finally, we state our conclusions in Sect. 6.

2   Related Literature

Standardizing haptic communication interfaces has been a pervasive challenge in the haptic research community. This primarily requires (i) comprehensive definition of haptic metadata format, (ii) exchanging and negotiation of metadata, and (iii) exchange of media payload.

Several works exist in the literature that have looked at definition and exchange of haptic metadata. In order to systematically describe various attributes of a haptic interface, researchers have proposed a structured data format. Early studies proposed an XML-based approach to represent generic haptic applications [8]. Cha et al. extended MPEG-4 Binary Format for Scenes (BIFS) [29] to support the synchronization and communication of haptic-audio-visual (HAV) media streams [9]. Others [14] have proposed HAML, a haptic applications metadata language, to describe haptic-related information, including haptic interfaces, haptic development APIs, and quality of experience requirements. A HAML-based authoring tool has also been developed [13] to facilitate the development of haptic applications for various haptic interfaces for non-programmer developers or artists. The work in [24] explored the use of Session Initiation Protocol (SIP), used commonly in VoIP sessions, for establishing haptic interactive sessions.

Only a handful of works have addressed the latter challenge of exchanging media payload using standardized protocols/tools. An example is the work in [24] that considered Real-time Transport protocol (RTP), which forms the cornerstone of VoIP applications, for haptic interactions.

A recent work in [20], presents Tactile Internet Metadata (TIM) and a haptic handshake protocol, the implementations of which were realized using WebRTC. This study is a continuation of our previous work where we evaluate the PnP system with heterogeneous haptic devices in a realistic tele-haptic application (Tele-writing). This design choice of using a browser-based API enabled the authors to develop the protocol in a cross-platform manner. In this work, we make significant enhancements to the work in [20] to come up with a PnP communication system for TI that we present in the following section.

3   Proposed PnP Communication System

In this section, we provide an overview of the proposed PnP communication system. The system consists of three tightly coupled components: Tactile Internet Metadata (TIM) scheme, haptic handshake protocol, and a real-time haptic-audio-video communication protocol to support network applications. While a

preliminary implementation of these components was done in the recent work
[20], we have enhanced it significantly in order to make the communication sys-
tem plug-and-play. Nevertheless, we will provide a holistic view of the proposed
system here.

3.1   HAV Handshake

We propose a three-way handshake protocol for the exchange of haptic-audio-
video (HAV) metadata between TI nodes, as shown in Fig. 1. The TI node
initiating the TI session (node A) advertises all of its capabilities/requirements
to the other participant (node B) through the *request* message. For example,
node A could be capable of supporting a set of haptic codec types and certain
maximum haptic refresh rate. In response, node B chooses a feasible option out of
the advertised specifications. For example, only a subset of the advertised codecs
and a lower refresh rate could be supported. Node B transmits the *response*
message carrying the chosen parameters. Upon reception of the *response* signal,
node A transmits *ACK* message indicating that the consensus on metadata is
reached. The packet structures of these messages are discussed in Sect. 3.2.

The reception of ACK marks the completion of HAV handshake phase where
the advertisement and negotiation of metadata happens. This is then followed
by the *media communication* phase in which the exchange of media and control
data corresponding to the live TI interaction is carried out. We discuss further
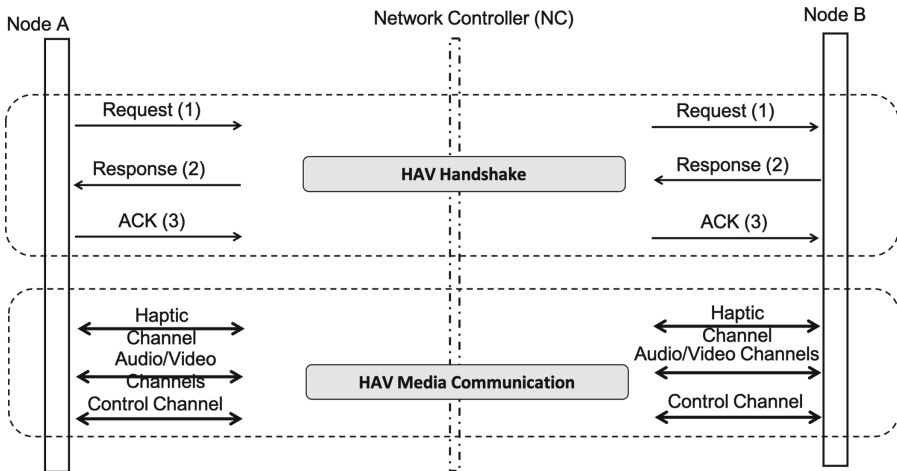details on this in Sect. 3.3.

Fig. 1. Schematic of the proposed HAV handshake. HAV synchronization involves a
simple 3-way handshake consisting of request, response, and acknowledgment.

3.2   Tactile Internet Metadata (TIM)

TIM is designed to provide a technology-neutral description of the various characteristics and requirements of TI systems in terms of session attributes. As shown in Fig. 2, the session attributes are organized into two broad categories: *Quality of Experience (QoE)* and *haptic*. QoE captures the essential parameters that are crucial for an immersive perception of the remote haptic interaction by the human operator. This category is further sub-divided into *Quality of Experience (QoS)* and *user experience*. QoS specifies the end-to-end network requirements for guaranteeing transparency between the TI nodes. For example, *latency* and *jitter* fields specify the maximum tolerable end-to-end delay and jitter, respectively. User experience specifies the perceptual attributes that will be used to describe the current quality of human perception, such as the quality of user immersion or telepresence.

On the other hand, the haptic category represents the haptic modality in terms of the properties of the *media* (data) and the haptic *interface* attributes. While the former describes the attributes for source coding and communication of the haptic data, the latter describes the capabilities of the haptic interface being employed in the TI session. These include the number of degrees of freedom, ranges of displayable force and torque, and position resolution, among others.
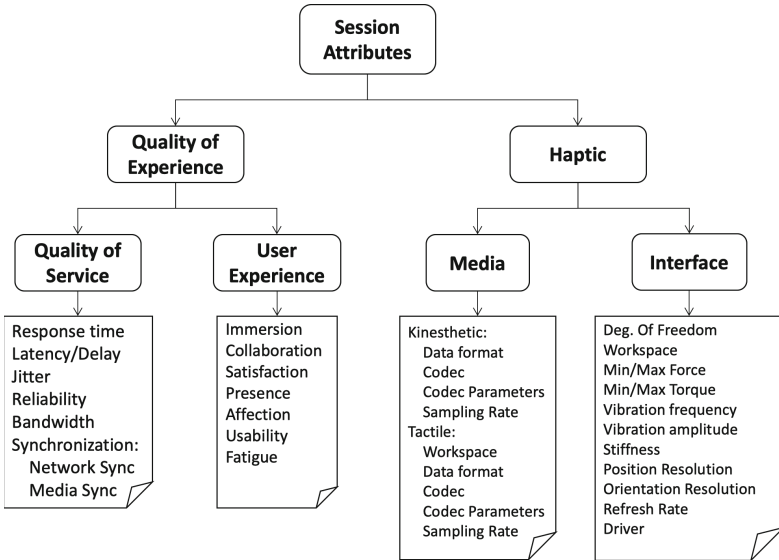

Fig. 2. Haptic session attributes as defined in TIM.

The TIM structure is adapted for various types of haptic handshake and communication messages, including the schema for the request/response messages and the data/control messages.

As for the haptic handshake, two types of messages are defined, namely the request and the response (Fig. 3 (a)). The message header consists of a *packetType* identifier for indicating the type of handshake message – request, response, or ACK. The header also include a packet sequence number for unique identification of the packet. The request/response message payload carries the metadata advertised/selected by the transmitting TI node. In order to support evolving requirements, *CustomHapticAttributes* field is provided in which several user-defined attributes may be added. The payload of the message is formatted in accordance with the TIM definition (Fig. 1).
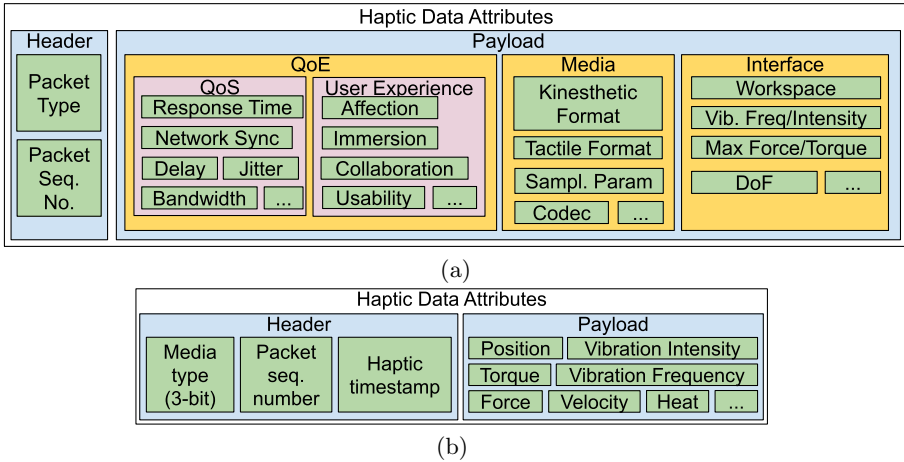

(a)

(b)

Fig. 3. Schematic representation of TIM packet format for (a) request/response, (b) data.

As seen in Fig. 3 (a), the attributes in the *interface* options of the request/response packet cater to the capabilities of the tactile device by keeping the attributes in the data packet within the lower/upper bounds of the specifications of the device. For instance, the TIM communicates the maximum force attribute during the handshake (via request/response packets) and a value for maximum force is set. If the payload of the data packet is force, it will stay within the bound of maximum force set during the handshake throughout the operation phase. Similarly, the attribute "immersion" could be set to "true" or "false" depending on the application.

3.3   HAV Media Communication

The media communication phase starts once the haptic handshake phase completes. In Fig. 1, this is shown as the operation state, consisting of haptic, audio-visual, and control channels. In the phase, two types of messages are exchanged: data message shown in Fig. 3 and (b) control message which is a combination of

fields in Fig. 3 (a). The data message simply carries a header and a payload. The header defines the media type (haptic, audio, video, or combination), the packet sequence number, and the haptic timestamp (utilized for intra- and inter-media synchronization). The payload comprises the corresponding haptic media data (such as position, force, torque or velocity for kinesthetic haptic interaction or vibration frequency, location, intensity for tactile feedback, etc.). On the other hand, the control messages are used to deal with dynamic control parameter adjustments during the data communication, such as change in network characteristics, addition of a new media type or user.

In this work, while we implement the handshake and media communication of haptic-related messages, for communicating AV-related messages, we simply invoke the de-facto standards – Real-time Transport Protocol (RTP) and Session Description Protocol (SDP) for data and control, respectively.

4 Implementation on WebRTC

In this section, we will provide the implementation details of our PnP communication system using Web Real Time Communication (WebRTC) – an open-source peer-to-peer cross-platform and cross-browser communication API [23].

4.1 TIM

A sample request packet of TIM is shown in Fig. 4. As can be seen it contains several attributes and their corresponding values. This particular example configures the deadband parameter for velocity signals ("VelocityDeadbandParameter") to 0.1, the manufacturer of the haptic device used ("manufacturerName") as "3D Systems", and the device model ("modelName") as "Geomagic Touch". Note that this excerpt of response packet is being taken from a real TI experiment of tele-writing that we discuss in detail in Sect. 4.4. Here, we used the kinesthetic codec [19] to smoothly interface haptic devices to the TI nodes. Other attributes related to QoS, media, and interface fields are specified in detail. The fields are arranged in JSON format. TIM is extensible and the exact attributes can be easily defined by the developers depending on their application needs. This can include any attributes related to handshake/data/control packets. For AV metadata, we simply leverage the SDP implementation of WebRTC.

4.2 HAV Handshake

Haptic handshake and communication requires open source protocols for accessibility, flexibility, and maintenance. For these reasons, commercial and proprietary services such as Skype and Google Hangout are challenging to adopt, although their texting features can certainly be re-purposed for haptic handshake. Due to these reasons, we resort to WebRTC for realizing our proposed system.

```
{"packetType":"request",
 "payload"    :{"QoS"        :{"CommandDelay":0},
                "Media"      :{"ControlMode":true,
                               "FlagVelocityKalmanFilter":false,
                               "ForceDeadbandParameter":0,
                               "PositionDeadbandParameter":0,
                               "RecordSignals":false,
                               "VelocityDeadbandParameter":0.1},
                "Interface":{"actuatedGripper":false,
                             "actuatedPosition":true,
                             "actuatedRotation":false,
                             "gripperMaxAngleRad":0,
                             "leftHand":true,
                             "manufacturerName":"3D Systems",
                             "maxAngularDamping":0,
                             "maxAngularStiffness:0,
                             "maxAngularTorque":0,
                             "maxGripperAngularDamping":0,
                             "maxGripperForce":0,
                             "maxGripperLinearStiffness":0,
                             "maxLinearDamping":4,
                             "maxLinearForce":3.3,
                             "maxLinearStiffness":400,
                             "model":14,"modelName":"Geomagic Touch",
                             "rightHand":true,
                             "sensedGripper":false,
                             "sensedPosition":true,
                             "sensedRotation":true,
                             "workspaceRadius":0.075}}}
```

Fig. 4. Sample TIM Request generated when TI node is connected to Geomagic Touch Device.

WebRTC handles AV data through Secure Real-Time Transport Protocol (SRTP) and non-AV data, which can be re-purposed for haptic data, through UDP-based, DTLS-encapsulated SCTP [RFC4960]. Supported by major browsers, it is being standardized through the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF) [23,25]. These features make WebRTC the most versatile solution among other open source options such as easyRTC [26], Jitsi [21], Linphone [2], Jami [1], Riot [4], and Retroshare [3].

AV Handshake: We leverage the standard AV handshake handled by WebRTC's *MediaStream* for handling AV handshake and communication. Figure 5 outlines the AV handshake. First, each TI node involved in the TI interaction creates a *PeerConnection* for the AV channel. These nodes then connect with each other through signaling messages. The request/response/ACK generations are automated, and the SDP exchange process between the nodes can be implemented using any third-party services such as emails, SMS, and external servers. After these steps are executed, the *PeerConnection* objects of the interacting nodes get attached to each other and the AV communication starts between the nodes.

Haptic Handshake: WebRTC provides a generic object *RTCDataChannel* that can be employed for any non-AV data communication. We make use of this for haptic data. Figure 6 outlines the haptic handshake. First, both TI nodes obtain TIM described earlier in Sect. 4.1. This is then encapsulated in SDP packets that are exchanged through the haptic handshake *RTCDataChannel*. Note that the haptic handshake leverages the *PeerConnection* that was established
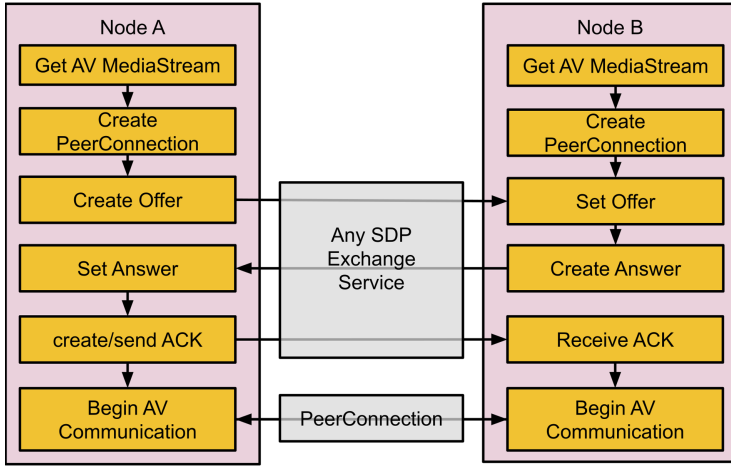
Fig. 5. Schematic of AV handshake carried out by WebRTC.

during the AV handshake. Hence, there are no more processes to be executed before the SDP exchanges can begin.
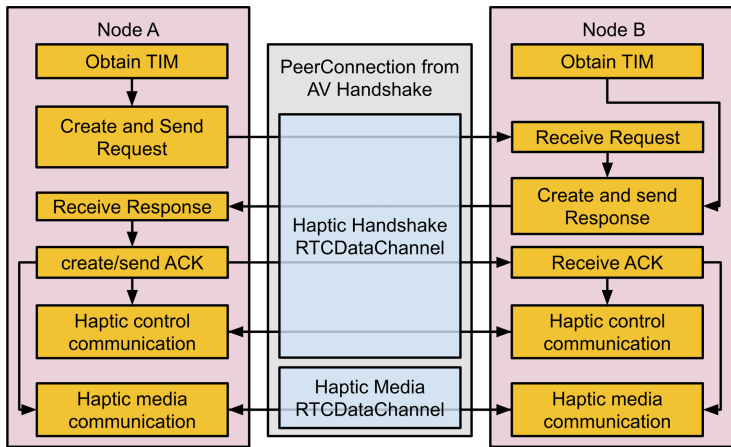
Fig. 6. Schematic of haptic handshake after completion of AV handshake.

AV and haptic media are intentionally handled in separate channels so that third party developers can use different AV codecs based on their own application needs. Note that the haptic metadata exchange happens separately from the AV metadata, however, haptic media communication may or may not happen with RTP.

4.3   HAV Media Communication

After the handshake, the *RTCDataChannel* that was used for haptic handshake is reused for control communication, and a second *RTCDataChannel* is opened to be used for haptic media communication, thus triggering the comprehensive HAV media communication.

   RTCDataChannel objects exist in WebRTC to address communication needs of various data, not only limited to haptic data. Hence, although not exactly TCP or UDP, they can be configured to protocols within those two ends of transport layer spectrum. The configurable options are listed below in Fig. 7:

```
const  mediaChannelOptions  =  {[[Ordered]] ,
                                [[MaxPacketLifeTime]] ,
                                [[MaxRetransmits]] ,
                                [[DataChannelProtocol]] ,
                                [[Negotiated]] ,
                                [[DataChannelId]]]};
```

Fig. 7. Different configuration options offered by the general *RTCDataChannel* object.

   When the parameter *Ordered* is set to true (default), it means choosing a reliable method of communication (leaning towards TCP). For UDP, it is set to false and *MaxRetransmits* is set to 0. Additionally, *RTCDataChannel* can control *MaxRetransmits* or *MaxPacketLifeTime* attributes but not both. *RTC-DataChannel* is by default negotiated in-band between two nodes. This means that the local node calls *createDataChannel()*, and the remote node connects to the *ondatachannel EventHandler*. This enables a dynamic creation of *RTC-DataChannel* where the number of channels is not predetermined. Alternatively, they can be negotiated out of-band, where both sides call *createDataChannel()* with a predetermined ID to create data channels statically. This method opens the channels with lower latency and has higher stability as the creation of the channels is symmetric.

   Based on the above descriptions, for the purpose of our experiments, we configured the *RTCDataChannel* objects used for haptic handshake, media, and control are to be UDP-like as shown below. Of course, there is flexibility to set the handshake and control channels to be more reliable (Fig. 8).

4.4   Tele-Writing Demonstration

To evaluate the performance of the proposed PnP communication system, we developed a tele-writing application where a human can write something physically on a piece of paper present in a remote location. For this demonstration, we attached a pen to the remote haptic device (Node A). The human user controls it through another device in his/her location (Node B). The two nodes are connected to each other through a star connection of category-5e RJ45 ethernet

```
const handshakeChannelOptions = {Ordered: false ,
                                  MaxRetransmits:0 ,
                                  negotiated: true ,
                                  id: 0};

const mediaChannelOptions = {Ordered: false ,
                             MaxRetransmits:0 ,
                             negotiated: true ,
                             id: 1}};
```

Fig. 8. Setting of the *RTCDataChannel* objects that are used for haptic signaling in the proposed communication system.

cables and gigabit desktop switch. The interface for the haptic device was programmed in C++ language. A schematic diagram of the complete tele-writing setup is shown in Fig. 9. The implementation of the tele-writing setup is divided into two programs at each node: a C++ program interfacing with the haptic device and a WebRTC simulation written in HTML, CSS, and JavaScript. Node A can be connected to either Geomagic Touch or Novint Falcon for controlling the Node B device, which uses a pen-mounted Novint Falcon. This allows demonstrations of the PnP system for both homogeneous and heterogeneous TI nodes. It should be noted that for the sake of simplicity we mount the pen to only one type of haptic device. However, the tele-writing application can be directly extended to any other haptic device as well.
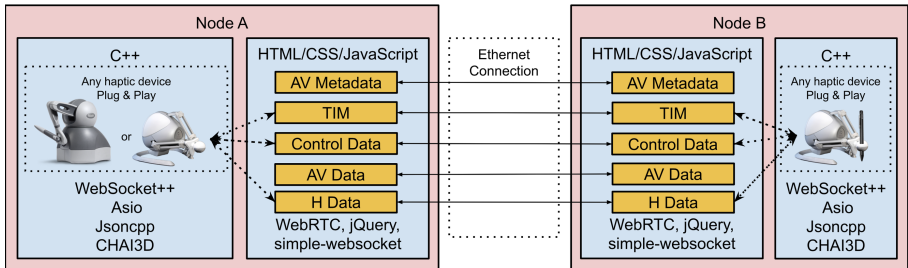

Fig. 9. Schematic diagram of PnP communication system for tele-writing application.

As noted in Fig. 9, the HAV WebRTC simulator was built by combining HTML, CSS, and JavaScript files from WebRTC's Munge SDP sample program with those from C++ WebSocket Server Demo's client project [5,28]. The client project consists of jQuery and simple-websocket [15,30]. jQuery simplifies traditionally verbose JavaScript expressions while simple-websocket is used to receive WebSocket data. The C++ haptic interfacing programs consist of C++ WebSocket Server Demo project and a simplified version of kinesthetic codec provided by the works in [17,19,28]. The WebSocket server projects consist of

WebSocket++ [27], Asio [10], and Jsoncpp [11]. The layout of the nodes are identical. The only difference is in how the signaling process is handled on each node. Node A first generates its own TIM, and based on this, Node B generates its own TIM. Node A, upon receiving this information, sends an acknowledgement to Node B and begins data communication.

Kinesthetic Codec Reference Software. We use the kinesthetic codec reference software, proposed by IEEE Haptic Codecs Work Group, to interface the C++ programs with the haptic devices, particularly in a well-controlled, stable manner [19]. It uses Chai3D engine to sense and actuate the haptic interface, queue-based haptic packet management for smooth communication, and Winsock's UDP mode to communicate between localhost-simulated TI nodes. To reduce congestion, it uses the perceptual deadband haptic data reduction.

Since UDP is not supported on browsers due to security issues, this was replaced with the WebSocket functionalities. Thus, the Chai3D-obtained haptic data are converted into stringified JSON via Jsoncpp and WebSocket++, and are then sent to the WebRTC simulator.

It is worth mentioning that the communication involved Node A sending timestamped velocity data to Node B. Based on this data, Node B device's force data is calculated through Algorithm 1 and sent back to Node A along with its own timestamp.

Algorithm 1: Slave force calculation algorithm.

Data: Newly read position p. Received velocity v. Previously stored error e_p and input force f_p. Coefficients A, B, and C.

$e \leftarrow 0.001 \cdot v - p$;
$f \leftarrow A \cdot e - B \cdot e_p - C \cdot f_p$;
$e_p \leftarrow e$;
$f_p \leftarrow f$;

Constants A, B, C are based on z-domain PD control and low pass filter applied to this system. They were calculated using Tustin's approximation prior to the media communication phase. The s-domain parameters used were $K = 1000$, $K_e = 5$, $T = 0.001\,\text{s}$, and $\tau = 0.0016\,\text{s}$. The s-domain transfer function was:

$$F = \frac{Ke + K_e\dot{e}}{\tau s + 1}$$

where, K, K_e, and τ respectively were Proportional Gain, Derivative Gain, and Low pass filter parameter. The resulting z-domain functions were:

$$A = \frac{2K_e + KT}{2\tau + T}, \quad B = -\frac{-2K_e + T}{2\tau + T}, \quad C = \frac{-2\tau + T}{2\tau + T}$$

The coefficients above can be modified to support applications with different control needs. These parameters can be included in the TIM packets if needed.

Figure 10 shows both homogeneous and heterogeneous modes of the tele-writing application along with C++ console for displaying the contents of request and response packets.

Figure 10 (a), Novint Falcon was used for controlling Node B, whereas in Fig. 10 (b), was used Geomagic Touch. In Fig. 10 (c), the C++ console log and web GUI from the Node A is shown, which, content-wise, was identical to Node B. Figure 10 (C). The C++ console log displaying packet communication rates, and the web GUI from Node A displaying SDP request and response from Node A and B, as well as their video data. The setup allowed the user to choose the audio, video, and haptic source devices before setting up the session. In this case, the AV data was provided from webcams connected to the nodes. Once the user began the WebRTC signaling process, the HAV communication commenced automatically.
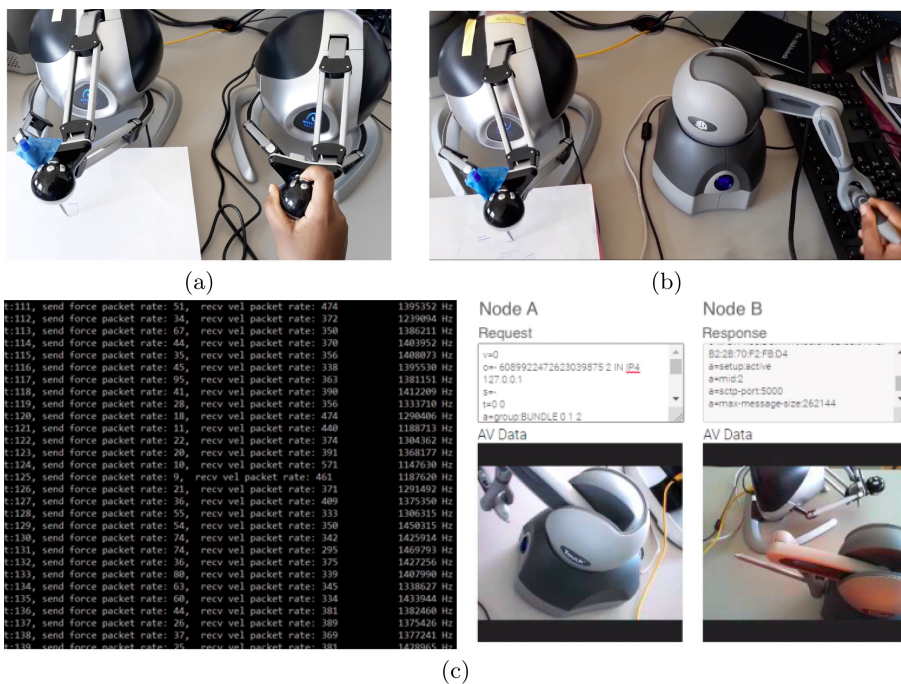

(a)

(b)

(c)

Fig. 10. Two heterogeneous nodes experimental setup. (a) Novint Falcon to Novint Falcon. (b) Geomagic Touch to Novint Falcon. (c) C++ console log displaying packet communication rates, and web GUI from Node A displaying SDP request and response from Node A and B, as well as their video data. Its content is identical to the other node.

5   Experimental Results

In this section, we present the latency findings of the tele-writing experiment. The mean and the standard deviation of the handshake latency is measured to be 47.25 ms and 23.38 ms, respectively. The performance of the bidirectional HAV communication was assessed through its mean roundtrip time (RTT) measured over the tele-writing experiment with a duration of 10 s. The standard packet rate of 1000 (consequence of 1 kHz sampling rate), leads to measuring RTT of 10000 packets. It was observed that the mean and standard deviation of RTT were 3.57 ms and 1.81 ms, respectively. The breakdown of the mean RTT is shown in Fig. 11. We observed a high level of consistency in these RTT measurements over several runs of the application.
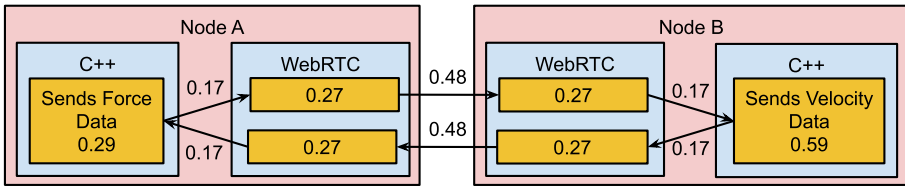

Fig. 11. Breakdown of 3.57 ms RTT 10000 packets were sampled in 10 s, at 1000 packets per second. Note that the C++ segments are only sample applications complimenting the handshake implementation and do not represent the performance of the handshake protocol.

Discussion: The average and standard deviation of RTT measured in our experiments indicate that the proposed system can provide stable bidirectional communication under both heterogeneous or homogeneous haptic interfaces. Around 1 ms of the total RTT, is the propagation delay between the TI nodes. Another 1 ms came from delays within the TI node browsers. This is attributed to WebRTC-related protocol handling. This is expected since the browsers are in high-level JavaScript-based implementation. Around 0.6 ms additional latency came from interfacing the C++ programs with the browsers through WebSocket-based communication. The remaining 1 ms latency came from the C++ programs, which is discussed later in detail.

Latency for some of these segments could potentially be eliminated. For example, switching from JavaScript Web API to native C++ implementation would address synchronization of AV data with haptic data. It will also allow the haptic interface setup to be integrated with WebRTC. This will eliminate any unnecessary cross-language latency. HAV handshake and communication latency between TI nodes will further reduce once WebRTC extends *RTCDataChannel* transport configuration option to pure UDP.

It should be noted that the latency for each of these segments are heavily dependent on the size of the haptic information being communicated. The

C++ programs are designed to send a set of timestamp and velocity/force data upon significant haptic device movement, with processing delays as low as around $1\,\mu$s. When they are wrapped in JSON formatting and converted to strings for communication, their sizes are typically around 120 bytes in length. If other types of data such as rotation, gripper movement, and button press also need to be communicated, the data size would further increase. The processing delay may then increase, thereby congesting the system. The communicated data should therefore be minimized by using compression mechanisms. Xu et al. effectively addresses such issues by demonstrating a combination of the perceptual-deadband haptic data reduction approach and the time domain passivity approach (TDPA) [32]. The size of the programs also affect the processing delay. As the C++ programs run multiple threads within themselves and the browser programs are executed asynchronously, adding more features could increase RTT proportionally more than when they are run on a single thread or run through a synchronous language. In addition to this, using browsers other than Chrome is known to increase the processing delays [22].

Lastly, the hardware-level performance of TI nodes might have contributed to the RTT as well. To our surprise, the latency in Node B's C++ program was twice as much as that of Node A, as seen in Fig. 11. These programs were developed to exchange different form of data, namely force or velocity, and while Node A requires calculation of force data based on the received velocity data, Node B has to simply actuate the received force data. Hence, the latency difference was most likely due to the fact that the Node A and B's haptic loop frequency were around 2 and $0.5\,$MHz respectively, as measured by Chai3D's frequency counter. This made Node A four times more responsive than Node B at handling possible congestions. To address this limitation, faster running programs or hardware should be used.

Taking these into account, the RTT is specific to the software and hardware environment it was implemented in. It is therefore recommended that for more complex TI applications, more efficient applications and hardware should be used. In addition, more types of haptic devices and applications can be tested to ensure that the PnP system is agnostic to multitude of scenarios. The above discussion suggests that as long as these recommendations are met, the system can be used to construct practical TI applications. These application will meet the sub-10 ms requirement for safe haptic control, and will be near the $1\,$ms average human haptic reaction time [16]. To our knowledge, this implementation is the only openly available WebRTC-based HAV communication system, and will therefore serve as an integral part of future TI application development.

6 Conclusions

In this paper, we presented the design of a WebRTC-based PnP communication system for TI interactions encompassing haptic feedback. We described in detail the TI Metadata (TIM) devised for conveying the haptic metadata of various TI nodes, the handshake protocol, and communication protocol for HAV interaction. Through implementation of the proposed PnP system on a WebRTC-based

platform and heterogeneous haptic interfaces, we provided a proof of concept of its operation. Further, the average and standard deviation of RTT were 3.57 ms and 1.81 ms, respectively. This paves way for sub-10 ms RTT crucial for TI interactions. As the system substantiates its usefulness for TI applications, it has been made open-source for further TI application development. The system will thus serve to form the foundation of TI application development. Such progress will drive innovation in global products and services, changing societies for the better. As for future work, we would like to evaluate the PnP system with a wider range of haptic interfaces, such as interfaces with multi-points of haptic interaction.

The proposed PnP communication system is aimed to be used by a broad set of audience. Hence, in future, we plan to make the project resources publicly available to fuel the explosive growth of TI.

References

1. Jami. https://jami.net/
2. Linphone. https://www.linphone.org/
3. Retroshare. https://retroshare.readthedocs.io
4. Riot. https://about.riot.im/
5. Webrtc samples munge SDP. https://webrtc.github.io/samples/src/content/peerconnection/munge-sdp/
6. Aijaz, A., Dohler, M., Aghvami, A.H., Friderikos, V., Frodigh, M.: Realizing the tactile internet: haptic communications over next generation 5G cellular networks. IEEE Wirel. Commun. **24**(2), 82–89 (2016)
7. Anderson, R.J., Spong, M.W.: Bilateral control of teleoperators with time delay. IEEE Trans. Autom. Control **34**(5), 494–501 (1989)
8. Carter, J., et al.: The gothi model of tactile and haptic interaction. In: Proceedings of GOTHI-05 Guidelines on Tactile and Haptic Interactions, pp. 93–95 (2005)
9. Cha, J., Ho, Y.S., Kim, Y., Ryu, J., Oakley, I.: A framework for haptic broadcasting. IEEE Multim. **16**(3), 16–27 (2009)
10. Kohlhoff, C.: Asio C++ library (2020). https://think-async.com/Asio/
11. Dunn, C.: open-source-parsers/jsoncpp (2020). https://github.com/open-source-parsers/jsoncpp
12. Dohler, M., et al.: Internet of skills, where robotics meets AI, 5G and the tactile internet. In: 2017 European Conference on Networks and Communications (EuCNC), pp. 1–5. IEEE (2017)
13. Eid, M., Andrews, S., Alamri, A., El Saddik, A.: HAMLAT: a HAML-based authoring tool for haptic application development. In: Ferre, M. (ed.) EuroHaptics 2008. LNCS, vol. 5024, pp. 857–866. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69057-3_108
14. Fayez, R., Eid, M., Orozco, M., El Saddik, A.: Haptic applications meta-language. In: 2006 Tenth IEEE International Symposium on Distributed Simulation and Real-Time Applications, pp. 261–264. IEEE (2006)
15. Aboukhadijeh, F.: feross/simple-websocket (2020). https://github.com/feross/simple-websocket
16. Fettweis, G.P.: The tactile internet: applications and challenges. IEEE Veh. Technol. Mag. **9**(1), 64–70 (2014). https://doi.org/10.1109/MVT.2013.2295069

17. Conti, F.: Chai3d (2020). https://www.chai3d.org/
18. Holland, O., et al.: The IEEE 1918.1 "tactile internet" standards working group and its standards. Proc. IEEE **107**(2), 256–279 (2019). https://doi.org/10.1109/JPROC.2018.2885541
19. IEEE P1918.1.1 Haptic Codecs for the Tactile Internet Task Group: Kinesthetic reference setup (2018). https://cloud.lmt.ei.tum.de/s/8ol5mX6TCDBS8t4
20. Iiyoshi, K., Tauseef, M., Gebremedhin, R., Gokhale, V., Eid, M.: Towards standardization of haptic handshake for tactile internet: a WebRTC-based implementation. In: 2019 IEEE International Symposium on Haptic, Audio and Visual Environments and Games (HAVE), pp. 1–6. IEEE (2019)
21. Ivov, E.: Jitsi. The architecture of open source applications, pp. 121–132 (2011)
22. Jansen, B., Goodwin, T., Gupta, V., Kuipers, F., Zussman, G.: Performance evaluation of WebRTC-based video conferencing. SIGMETRICS Perform. Eval. Rev. **45**(3), 56–68 (2018). https://doi.org/10.1145/3199524.3199534
23. Johnston, A.B., Burnett, D.C.: WebRTC: APIs and RTCWEB protocols of the HTML5 real-time web. Digital Codex LLC (2012)
24. King, H.H., Hannaford, B., Kammerly, J., Steinbachy, E.: Establishing multimodal telepresence sessions using the session initiation protocol (SIP) and advanced haptic codecs. In: 2010 IEEE Haptics Symposium, pp. 321–325. IEEE (2010)
25. Loreto, S., Romano, S.P.: Real-time communications in the web: issues, achievements, and ongoing standardization efforts. IEEE Internet Comput. **16**(5), 68–73 (2012). https://doi.org/10.1109/MIC.2012.115
26. Pelton, D.: Easyrtc framework tutorial (2013)
27. Thorson, P.: zaphoyd/websocketpp (2020). https://github.com/zaphoyd/websocketpp
28. Rehn, A.: Websocket server demo (2020). https://github.com/adamrehn/websocket-server-demo
29. Signes, J., Fisher, Y., Eleftheriadis, A.: Mpeg-4's binary format for scene description. Sig. Process.: Image Commun. **15**(4–5), 321–345 (2000)
30. The jQuery Foundation: jquery (2020). https://jquery.com/
31. de Vries, R., Jager, G., Tijssen, I., Zandstra, E.H.: Shopping for products in a virtual world: Why haptics and visuals are equally important in shaping consumer perceptions and attitudes. Food Qual. Prefer. **66**, 64–75 (2018)
32. Xu, X., Panzirsch, M., Liu, Q., Steinbach, E.: Integrating haptic data reduction with energy reflection-based passivity control for time-delayed teleoperation. In: 2020 IEEE Haptics Symposium (HAPTICS), pp. 109–114 (2020)