Setting up CLUE telepresence sessions via the WebRTC data channel

R. Presta, S. P. Romano
University of Napoli Federico II, Napoli, Italy
{roberta.presta,spromano}@unina.it

ABSTRACT

In this paper we present the current results of our ongoing work in the field of standardization of the CLUE telepresence framework within the Internet Engineering Task Force. We first introduce the CLUE architecture and data model. Then, we move to the definition of an application-level protocol for the negotiation and setup of telepresence sessions between CLUE-enabled entities. We propose to transport CLUE protocol messages on top of the WebRTC data channel, which is a further ongoing effort involving both the Internet and the World Wide Web standards organizations. The data channel allows for low-latency, secure transmission of generic data in a peer-to-peer fashion. The paper also describes a proof-of-concept implementation of the mentioned features. Such an implementation aims to represent a living playground for the collection of real-world data for the definition of a call flows standard document illustrating protocol interactions and use cases.

Keywords

Telepresence, IETF, W3C, RtcWeb/WebRTC, Network Protocols, Data Channel, Web communication, JavaScript

1. INTRODUCTION

Telepresence refers to the set of technologies allowing for real-time communication among several remote participants who have the sensation of being in a shared environment. The main aim of such technologies is to let users feel like being present in a place other than their true location, by properly describing, transmitting and rendering an integrated set of real-time multimedia streams. Telepresence session participants have to be immersed in a properly equipped room providing them with multi-sensorial stimuli replicating in space and time what is going on in the remote location they are connected to. The equipment within the room must be able to capture and send all the information needed to allow a high-fidelity reproduction at a remote site of local participants' movements, voice, appearance, background sounds, etc.. This can be accomplished by using ad hoc designed rooms with, for example, multiple displays permitting life size image reproduction, multiple cameras, microphones and loudspeakers.

Of course, video conferencing can be seen as a basic way of doing telepresence. However, it is not conceived at the outset to provide such an immersive experience. Participants' high quality video is certainly one of the main media streams a telepresence session involves. Nonetheless, enriched telepresence sessions are conceptually made of more media streams than a typical video conference. Indeed, telepresence rooms are equipped with several capture devices wisely distributed to best capture the local scene from multiple view points. Each of these streams should be sent to the remote site in order to make it able to choose the set of streams that best fits its rendering capabilities, i.e., the displays and loud-speakers installed at the remote location. Remote sites must be informed of the contents of all received media streams and select the most suitable combination they wish to be provided with. The same holds for information about original spatial distribution of the streams, which has to be transferred to the remote site to better duplicate the source environment.

Similarly to the case of multimedia web conferencing [4], the need for standardization arises to address interoperability issues. Several proprietary telepresence systems exist to date but, although mainly based on open standards (SIP, RTP, H.264, the H.323 suite), their interoperability is achievable only by means of operator assistance and expensive additional equipment allowing for intercommunication among heterogeneous vendor platforms. The main obstacle is the lack of a common way for fully describing and negotiating the multiple media streams involved in a telepresence session. The Internet Engineering Task Force (IETF) standardization body created in 2010 a dedicated working group to address the problem, named "CLUE" (ControLling mUltiple streams for tElepresence) [1].

The authors of this paper are active participants in the mentioned working group, with special focus on the definition of the data model associated with the description of telepresence sessions, as well as of the protocol used to exchange this type of information between CLUE-enabled participants. Within the framework of such activities, we have worked on prototype implementations of the proposed ideas, so to make available to the community a living experimentation playground to be leveraged for on-the-field evaluation of the actual operation of the ongoing standard proposals. More precisely, we have recently focused on the realization of a simple architecture involving two CLUE-enabled entities willing to negotiate, through the CLUE protocol and by leveraging the CLUE data model, a simple yet complete telepresence session. Since the protocol we are defining runs at the application level, we chose to rely, for the transport of its messages, on the WebRTC [3] data channel, which is a logical abstraction for a generic, peer-to-peer data communication link between any pair of WebRTC-enabled entities (either browsers or a different kind of endpoints compliant with the WebRTC specification). The data channel is a further ongoing standardization effort being developed as part of a joint initiative between the IETF (where a working group called RtcWeb [2] has been created) and the World Wide Web Consortium (W3C).

This paper deals with the above mentioned activities and is structured in eight sections. We start by providing the reader with some related work in Sec.2 and with the needed background information about telepresence standardization activities within the IETF in Sec. 3. Then, we dig into some of the details of the CLUE protocol, in Sec. 4. A quick introduction to WebRTC and the data channel is hence provided in Sec. 5, which preludes to the definition of an integrated approach to the setup of CLUE sessions on top of such an advanced communication means. A simple explanatory example of the mentioned ideas is reported in Sec. 7, which illustrates how to negotiate and properly set up (between any pair of WebRTC-compatible browsers) a simple telepresence session by exchanging, on top of the data channel, CLUE protocol messages. Sec. 8 concludes the paper, by summarizing its main achievements, at the same time discussing open issues and identifying some directions of our future investigations in the field.

2. RELATED WORK

Telepresence, conceived in a broad sense as the sharing of a virtual environment among remote communicating users, has been widely investigated in the literature [11]. The term first appeared in a well known work on the topic from Minsky, published in 1980 and referring to a teleoperation scenario with a human agent having the perception of being and acting in the remote site thanks to the stimuli provided by the user interface of the system. The user experience, and the human factors design issues in general, are indeed among the main targets of the research efforts [6] for technology innovations. Three-dimensional technologies have become one of the most active areas of investigation. Their incorporation into dedicated conference rooms provides end users with life-sized and realistic 3-D virtual audio-visual immersive environments [8].

From a communication network perspective, commercial telepresence systems typically rely on proprietary conferencing architectures realized ad-hoc to best exploit the features of the inter-communicating conference rooms. As already mentioned in Sec. 1, though such systems usually exploit existing standard technologies, conference sites equipped by different vendors are not able to interoperate because of the lack of a common framework enabling for the comprehensive description and negotiation of the involved media streams. This is recognized as one of the main limits of the current telepresence systems[1] and that's why, besides the IETF, also the ITU standardization body activated some study groups on the topic[2].

At the time of writing, the IETF interoperability research effort on telepresence systems is near to completion. The paper discusses such an effort from an insiders' perspective and provides a prototype implementation of the developed framework. The prototyping activity has already proved to be useful for the standards development process, thanks to the insights deriving from the practical implementation of real world scenarios [5].

To the best of our knowledge there are no other works in the literature about CLUE results and experimentation. In particular, this work represents a first attempt of combining CLUE mechanisms with the upcoming web real time communication technologies.

3. THE CLUE FRAMEWORK AND DATA MODEL

Among the major interoperability requirements imposed by telepresence, there are those related to the description of the available media streams a transmitting site can send to remote sites. Such a description should help the receiver in the selection process and in the rendering phase which is critical for the provisioning of a real "being-there" experience to the telepresence user.

In SIP-based conferencing, media streams in a multimedia session are usually described (in terms of their type, as well as available encodings) by means of the Session Description Protocol (SDP). Nonetheless, SDP and its extensions do not address the aforementioned description needs. First, SDP does not envision the possibility to provide spatial information about media streams. Second, the "content" SDP extension [9], conceived to specify the information represented by the media stream in a more detailed fashion than the media description line, has been considered unsatisfactory to represent the complete semantics of the streams involved in a telepresence session, since it does not cover all of the desired facets. Moreover, its intrinsic ambiguity makes it unsuitable to be machine-interpreted. Such a limitation explains why the media streams negotiation carried out by means of SDP within SIP cannot fulfill completely the telepresence requirements.

The CLUE framework aims to define information for the set up of a telepresence session. Negotiation mechanisms are also specified on the basis of both the collected requirements [16] and the envisioned use cases [17]. The CLUE framework document [7] provides the big picture about the overall architecture.

For the sake of simplicity, the focus is on the communication between a Media Provider (transmitting endpoint, MP) and a Media Consumer (receiving endpoint, MC), though

[1]http://www.forbes.com/sites/ciocentral/2012/05/03/5-reasons-cisco-and-polycom-are-in-trouble-in-telepresence/
[2]http://www.itu.int/ITU-T/studygroups/com16/sg16-q5.html

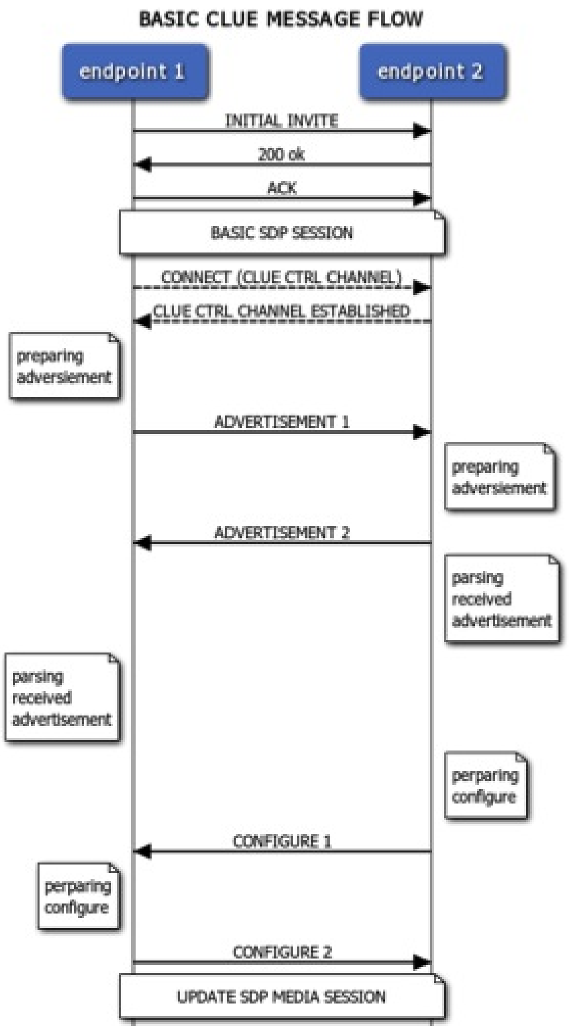

Figure 1: CLUE signaling overview

each dialogue should be considered bidirectional and multiple endpoints communication scenarios are allowed by introducing Multipoint Control Units acting as the center of a star communication topology.

From a signaling [12] perspective (Fig.1), the two endpoints first set up a signaling channel (the CLUE data channel [10]) and then start to exchange the metadata about the media streams and their desired configuration by means of a dedicated protocol (the CLUE protocol [15], which will be further detailed in the following). Such a protocol essentially envisions an `ADVERTISEMENT` message, sent from the MP to the MC, and a `CONFIGURE` message, flowing along the opposite direction as a response to the received `ADVERTISEMENT` message. By means of the `ADVERTISEMENT` message, the Media Provider informs the Media Consumer about its telepresence capabilities, i.e., a set of properly arranged information about the available media and related metadata. On the other hand, the `CONFIGURE` message allows the receiver to communicate to the sender the results of its internal selection process, by indicating the media streams of interest, as well as their detailed configuration.

Besides the big picture, the CLUE framework drills down the details about the information that needs to be exchanged by means of the protocol. Such an information is formally defined in [14] by using the XML Schema language. The telepresence capabilities of the MP are mainly described in terms of:

- *media captures*: within the CLUE framework, media captures are the fundamental representations of the streams an endpoint can transmit. They can represent the immediate output of either a physical source (e.g., camera, microphone) or a "synthetic" source (e.g., laptop, computer, DVD player), but also concepts such as "the loudest speaker stream".
- *encoding groups*: groups of encodings sharing a certain maximum bandwidth. Each media capture is indeed associated with an encoding group, meaning that it can be encoded with the encodings included in the group. Media captures sharing the same group must respect the maximum bandwidth constraints when encoded.
- *capture scenes*: aggregates of media captures semantically and/or spatially related, i.e., representing the same subjects/portion of the room and/or sharing the same coordinate space. This grouping of captures helps the Media Consumer understand which captures represent a certain view of the remote telepresence endpoint.
- *simultaneous sets*: groups of captures that can be encoded and sent simultaneously.

On the other hand, the Media Consumer needs to specify the desired streams in terms of *capture encodings*, a term used in CLUE to indicate the real media streams resulting from the instantiation of a certain media capture encoded with certain encodings. Capture encodings allow a Media Consumer to specify also some options about the selected captures.

A significant part of the efforts in CLUE has been devoted to the design of the attributes that can be used to characterize media captures. Such attributes include of course the spatial information about the captures, that can be represented with the position of the capturing device within the source environment and, when dealing with video captures, also with the coordinates of the represented area. Besides that, other features like the potential mobility of the capture point, information about the represented participants (e.g., their names and roles in the conference), the switching policy according to which the content of the capture is changed (e.g., the loudest segment representation), can be indicated.

Such an organization of the advertised information allows the Media Consumer to clearly understand (both spatially and semantically, not to say quantitatively) the telepresence options available on the Media Provider's side. As a consequence, the Media Consumer can formulate its request of media streams being aware of all the necessary information and constraints that need to be mapped with its receiving and rendering capabilities.

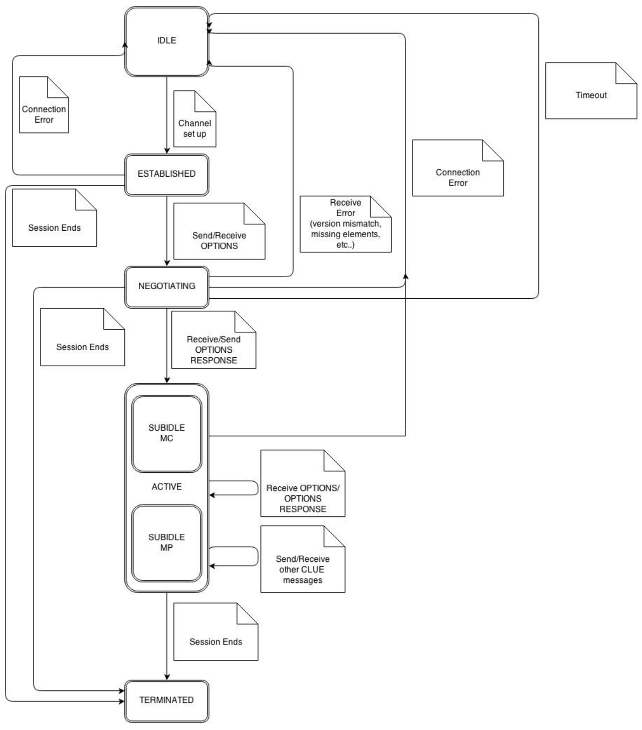

Figure 2: CLUE protocol state machine

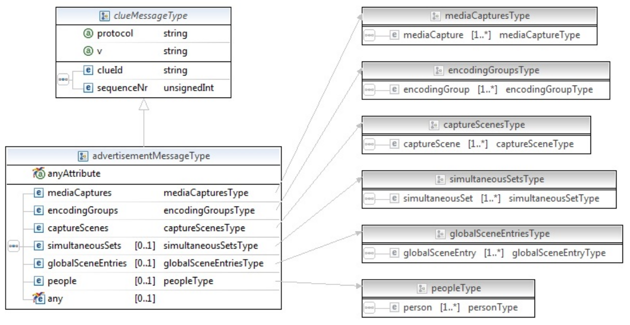
clueMessageType
protocol string
v string
clueId string
sequenceNr unsignedInt
advertisementMessageType
anyAttribute
mediaCaptures mediaCapturesType
encodingGroups encodingGroupsType
captureScenes captureScenesType
simultaneousSets [0..1] simultaneousSetsType
globalSceneEntries [0..1] globalSceneEntriesType
people [0..1] peopleType
any [0..1]
mediaCapturesType
mediaCapture [1..*] mediaCaptureType
encodingGroupsType
encodingGroup [1..*] encodingGroupType
captureScenesType
captureScene [1..*] captureSceneType
simultaneousSetsType
simultaneousSet [1..*] simultaneousSetType
globalSceneEntriesType
globalSceneEntry [1..*] globalSceneEntryType
peopleType
person [1..*] personType

Figure 3: Schema of the ADVERTISEMENT message

4. CLUE PROTOCOL

The CLUE protocol is the client-server application protocol conceived for the set up and configuration of a CLUE telepresence session. CLUE protocol messages are textual, XML-based messages flowing between the CLUE endpoints linked by the CLUE data channel. They basically carry information imported from the XML-based CLUE data model in order to let Media Providers announce their telepresence capabilities (in terms of media captures, encoding groups, etc.) and Media Consumers request the desired multimedia streams among those advertised.

Fig.2 represents the state machine of a CLUE endpoint. If the data channel set up phase is successfully completed, the CLUE endpoints move from the *IDLE* state to the *ESTABLISHED* one, where the initiation phase takes place. The CLUE protocol comes into action at that moment (*NEGOTIATING* state). By means of a special couple of CLUE messages (*OPTIONS* and *OPTIONS RESPONSE*), the CLUE endpoints agree on the version of the CLUE protocol and on the options to be used in the telepresence session. At the end of that basic negotiation, each CLUE endpoint starts its activity as a Media Provider and/or Media Consumer (*SUBIDLE-MP* and *SUBIDLE-MC* macro states, respectively).

Media Provider and Media Consumer state machines are not presented for the sake of brevity. On the MP's side, the `ADVERTISEMENT` message is sent to the other endpoint as soon as the initiation phase is completed, while on the MC's site, the endpoint is idle waiting for an `ADVERTISEMENT`, from which a reply is properly built in the form of a `CONFIGURE` message. Besides `ADVERTISEMENT` and `CONFIGURE`, other messages have been conceived in order to provide all the needed mechanisms and operations.

All CLUE messages show a common basic structure which is specialized with specific information from case to case. In Fig.3 the ADVERTISEMENT example, extrapolated from the schema file under standardization, is depicted. The common part ("clue message type") includes information like the version of the protocol and the sequence number, while all the information presented in Sec. 3 and needed to appropriately describe the telepresence offer of the MP appears in the specialized part of the message ("advertisement message type").

5. WEBRTC AND THE DATA CHANNEL

Web Real Time Communication (WebRTC) [13] is the new frontier of web-based interaction and collaboration. It is an upcoming new standard which aims to enable real time communication among web browsers in a Peer-to-Peer fashion. The IETF RTCWeb and the W3C WebRTC working groups are jointly defining both the Application Programming Interfaces and the underlying communication protocols for the setup and management of a reliable communication channel between any pair of next-generation web browsers. The objective of the mentioned standardization activities is to define a W3C Application Programming Interface (API) that enables a web application running on any device, through the secure access to the input peripherals (such as webcams and microphones), to send and receive real-time media and data in a peer-to-peer fashion between the browsers. The design of the API leaves the web developer free to implement functionality for finding and connecting participants in a communication session. It relies on existing protocols identified by the IETF community as the most appropriate to address the network-related aspects (control protocols, connection establishment and management, connection-less transport, selection of the most suitable encoders and decoders, etc.). The architectural model of real-time communication, the so called Browser RTC Trapezoid, allows the media path to flow directly between browsers without any intervening servers, while the related signaling path crosses servers that can modify, translate or manage signals as needed. The WebRTC API defines the mechanisms allowing client-side web applications (typically written in a mix of HTML and JavaScript) to interact with web browsers in both a proactive (e.g., to query browser capabilities) and a reactive (e.g., to receive browser-generated notifications) way. The mentioned application-browser API actually provides a wide set of functions, like connection management (in a peer-to-peer fashion), encoding/decoding capabilities negotiation, selection and control, media control, firewall and NAT traversal. As part of the API, the `RTCDataChannel` interface represents a bidirectional data channel for the streaming of generic data between two peers. Each data channel has an associated underlying data transport that is used to transport data to the other peer. The transport protocol between the peers is SCTP (Stream Control Transmission Protocol). An `RTCDataChannel` can be configured to operate in different reliability modes. A reliable channel ensures that data is delivered to the other peer through retransmissions. An unreliable channel is configured to either limit the number of retransmissions or set a time during which retransmissions are allowed.

Fig. 4 provides the big picture associated with a complete WebRTC call flow involving a channel Initiator, a channel Joiner, and a signaling server relaying messages between them at channel setup time.

The sequence diagram evolves through the following macro-steps:

1. The Initiator connects to the server and lets it create the signaling channel;
2. The Initiator (after getting user's consent) gets access to the user's media;
3. The Joiner connects to the server and joins the channel;
4. When the Joiner also gets access to the local user's media, a message is sent to the Initiator (through the server), triggering the negotiation procedure:
    - The Initiator creates a `PeerConnection`, adds the local stream to it, creates an SDP offer, and sends it to the Joiner via the signaling server;
    - Upon receipt of the SDP offer, the Joiner mirrors the behavior of the Initiator by creating a `PeerConnection` object, adding the local stream to it, and building an SDP answer to be sent back (via the server) to the remote party.
5. During negotiation, the two parties leverage the signaling server to exchange network reachability information (in the form of ICE protocol candidate addresses);
6. When the Initiator receives the Joiner's answer to its own offer, the negotiation procedure is over: the two parties switch to peer-to-peer communication by exploiting their respective `PeerConnection` objects, which have also been equipped with a data channel that can be used to exchange text messages directly

It is on top of the above mentioned data channel that we let two CLUE-enabled peers exchange CLUE protocol information, as it will be illustrated in the next section.

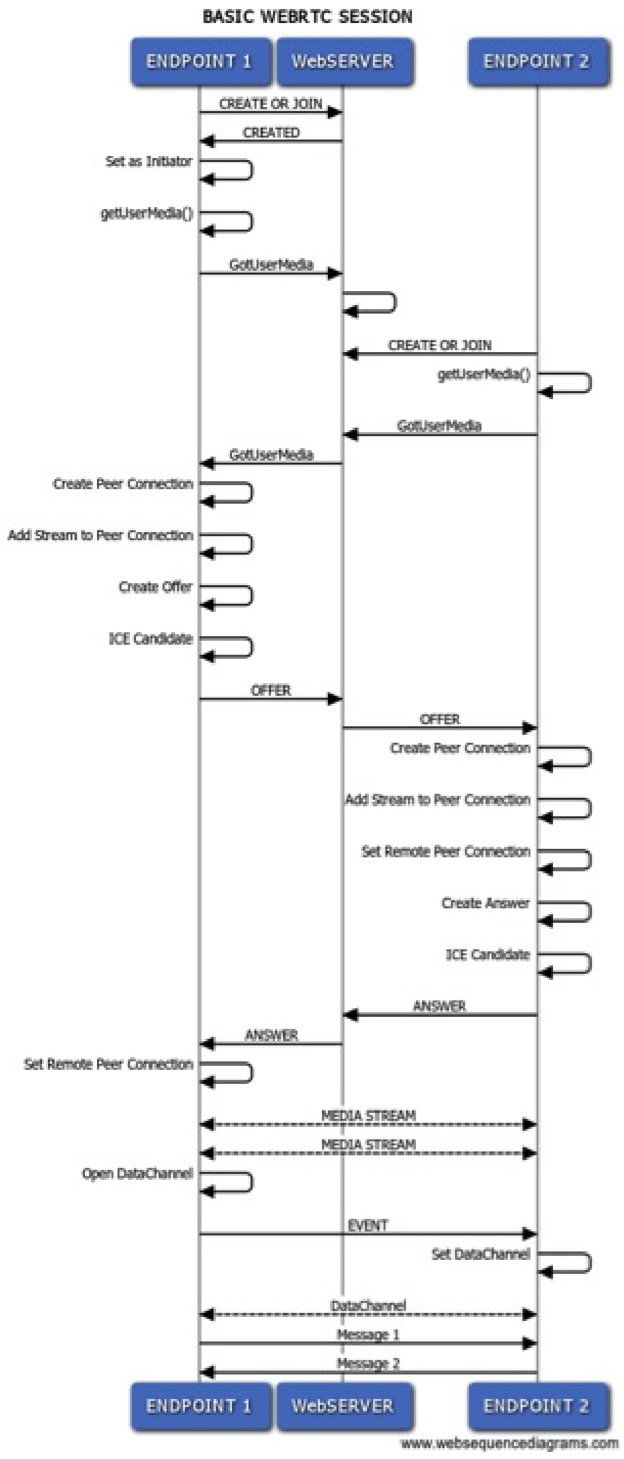

Figure 4: Basic WebRTC call flow

6. RUNNING THE CLUE PROTOCOL ON TOP OF THE DATA CHANNEL

As we briefly illustrated in Sec. 3 when discussing signaling in CLUE, the basic use case behind the CLUE protocol envisages the presence of two endpoints, each acting as both a Media Provider (MP) and a Media Consumer (MC) and exchanging with the remote party spatial information about the available media streams through the two main protocol messages, namely `ADVERTISEMENT` and `CONFIGURE`.

Before being able to do so, it is mandatory for the two parties to exchange properly formatted *Offer/Answer* messages aimed at establishing both a basic multimedia session and, more importantly, the so-called CLUE channel to be used for all subsequent CLUE-related information exchanges. Such a channel is indeed fundamental since it in the first place represents a means for an endpoint to indicate to the other party its capability to 'communicate' through the CLUE protocol.

Across the CLUE channel, each endpoint acting as a Media Provider sends to the remote party a detailed description (in terms of features and capabilities) of the streams it is capable to produce. This is achieved through the creation of an `ADVERTISEMENT` message. Upon reception of such a message, the remote party, this time playing the Media Consumer role, can properly build a `CONFIGURE` message containing information about the streams (as well as related features) the recipient is willing to get from the producer. This message exchange must be followed by a further Offer/Answer phase aimed at properly updating the features of the real-time communication channel between the two parties.

From this perspective, CLUE is a meta-protocol sitting on top of existing (even though possibly enhanced) session setup protocols like SDP and as such it calls for some form of loosely coupled synchronization between CLUE-level and session-level information (e.g., to associate CLUE stream identifiers with SDP media identifiers).

From the above discussion, it looks clear how the integration between CLUE and WebRTC can actually take place in a quite straightforward way. Indeed, the tools made available for real-time communication between browsers functionally mirror those required for CLUE-enabled information exchanging. The WebRTC `MediaStream` API makes available suitable tools for the effective management of multimedia streams. Real-time peer-to-peer communication is enabled by the `PeerConnection` API. The `DataChannel` API provides the required message exchanging functionality on top of the CLUE channel.

7. A PROOF-OF-CONCEPT PROTOTYPE

In this section we briefly describe how we implemented a simple prototype for the experimental study of the behavior of the CLUE protocol in a number of telepresence scenarios deployed in the context of WebRTC. We will first touch upon the overall design of the prototype architecture and hence dig into some relevant implementation details. Finally, we will discuss some of the scenarios we were able to reproduce.

In order to effectively emulate a telepresence scenario by using two WebRTC-enabled endpoints, it is clearly necessary to make available multiple streams per each involved partner: in such a case it is in fact useful to properly 'advertise' the availability of a set of alternative rendering options (e.g., in terms of spatial information) among which the receiver can choose its preferred 'configuration'. We have hence decided to use three different cameras and three different 'screens' per side. We have then analyzed the behavior of the CLUE protocol, given the mentioned setup representing a fictitious scenario involving two simple telepresence rooms. Each endpoint has hence been realized as a WebRTC application capable to gain access to three different multimedia flows (associated with the three available cameras) and advertise their availability to the remote party. The application starts with the exchanging of a first video flow allowing for the initialization of a basic WebRTC session. Once the basic session has been set up, the CLUE protocol comes into play: (i) `ADVERTISEMENT` messages are sent across the instantiated data channel by both sides to inform the recipient about the available sender's transmission options; (ii) `CONFIGURE` messages flow along the opposite direction in order to allow the recipients to properly set the desired reception options. The CLUE protocol is hence used to properly negotiate the addition of two more (customized) streams to the basic WebRTC session negotiated at session setup time.

7.1 Prototype architecture

As Fig. 5 shows, the prototype has been implemented as a typical WebRTC application, involving two browsers which both execute some JavaScript code embedded inside an HTML5 web page downloaded from a server. The server also acts as the needed signaling 'mediator' to be leveraged at session setup time by both endpoints (see Sec. 5). In the picture, the channel named *PEER CONNECTION* is the one used for the transport of multimedia flows; the companion *DATA CHANNEL* link is instead used for the exchanging of CLUE protocol messages.

The endpoint application used by both parties is the Google Chrome Canary browser. The signaling server is realized with *Node.js*, an extremely powerful software platform based on a single-threaded event loop management process (making use of nonblocking I/O) and allowing users to easily build scalable server-side applications with JavaScript. We also used *Adapter.js*, a handy JavaScript shim library that helps the programmer by properly abstracting browser prefixes, as well as other browser differences and changes in the way vendors are currently interpreting the specs. Finally, we have developed a further JavaScript library, named *CLUE.js*, to deal with the construction of CLUE messages starting from JSON (JavaScript Object Notation) templates stored at the Web Server and which can be flexibly manipulated through specific methods for the transformation in either XML or string format.

7.2 A simple telepresence scenario

As already anticipated, the structure of the sample telepresence application is based on the acquisition and configuration of three different video sources to be exchanged between the two CLUE-enabled parties in a customizable fashion. The web interface associated with such a scenario envisages the presence of two different three-views panels, as shown

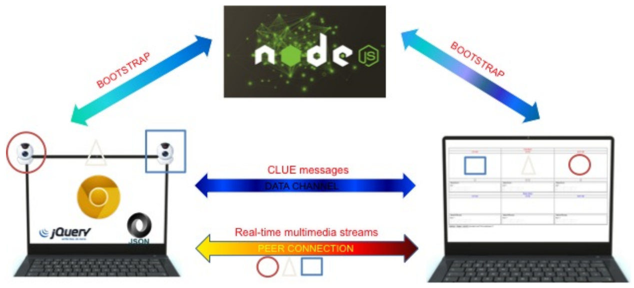

Figure 5: CLUE over WebRTC prototype structure

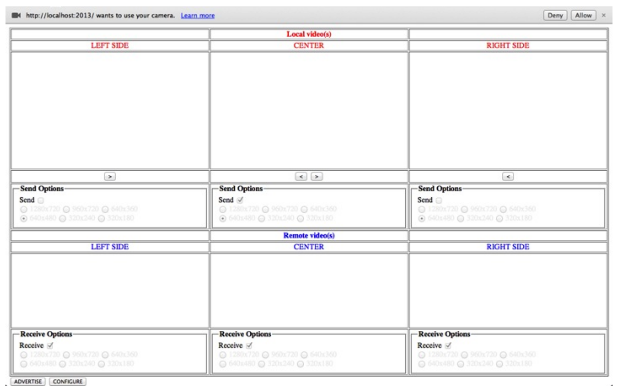

Figure 6: Prototype in action: main window structure

in Fig. 6. The upper panel refers to the local view of the acquired streams (i.e., the streams that can be advertised to the remote party). The lower panel is instead associated with the rendering of the streams received from the remote endpoint (i.e., those that the remote endpoint has advertised and which have been properly configured by the local endpoint through the transmission of a `CONFIGURE` message).

The first step of the proof-of-concept scenario we implemented envisages the acquisition, on the Media Producer side, of the three available video flows (one per each camera attached to the producer's laptop). Such a phase is immediately followed by the setup of the data-channel-enabled WebRTC session between the two parties. As soon as the session is set up, the Media Producer starts waiting for the parameters needed for the construction of an `ADVERTISEMENT` message. This is shown in Fig. 7

CLUE message parameters are initialized through the configuration panels available in the *Local Video* and *Remote Video* sections of the prototype main window. Such panels are disabled at application startup. As soon as local streams become available (i.e., right after the `getUserMedia()` calls have successfully completed), the `ADVERTISEMENT` configuration panels get activated, hence allowing the Media Producer to properly set the properties of the advertised streams. Similarly, when a remote stream is attached to the corresponding HTML5 <video> tag in the web GUI, the panel associated with the characterization of a `CONFIGURE` message is activated.

On the producer's side, once the set of resolutions to be advertised has been chosen per each available stream, a proper

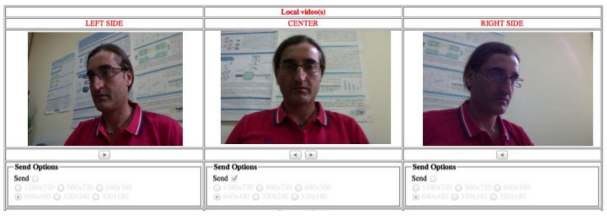

Figure 7: Sender side ready to send ADVERTISE message

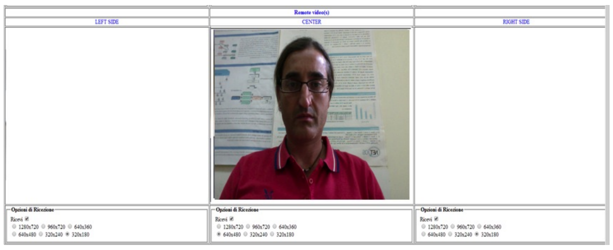

Figure 8: Receiver side before CONFIGURE message

`ADVERTISEMENT` message is built through the CLUE.js library and sent to the consumer across the data channel. The same happens on the consumer's side when it comes to the construction of the desired `CONFIGURE` message.

Coming back to the prototype, as Fig. 7 shows, on the provider's side multiple streams can be acquired and an `ADVERTISEMENT` can then be manually configured.

On the consumer's side, we initially get just the stream associated with the basic multimedia session configured at setup time, i.e., before the CLUE protocol comes to the fore. This is shown in Fig. 8.

Once the CLUE protocol has been activated and after the producer has sent its `ADVERTISEMENT` to the remote party, the desired `CONFIGURE` message can be constructed by the consumer and sent back to the other peer. Thanks to the `CONFIGURE` message, the producer becomes aware of the streams (each with specific resolution features) that must be added to the communication. In the screenshot in Fig. 9, we report an example where the consumer requests a 640x480 stream to be attached to the middle panel and two 320x180 streams to be associated with left and right panels, respectively.

8. CONCLUSIONS AND FUTURE WORK

In this paper we presented the current achievements of our ongoing work in the field of telepresence, with a special fo-

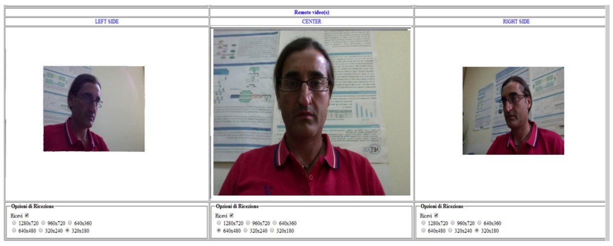

Figure 9: Receiver side after CONFIGURE message

cus on protocol standardization activities, as well as on the implementation of proof-of-concept prototypes to be used for testing and validation purposes. All of the results we presented are work in progress and hence subject to substantial changes as long as the standardization activities get further developed. For what concerns our future activities, we plan to keep on refining the specification of the CLUE protocol and data model based on the feedbacks coming from the IETF community. We also intend to improve the prototype and keep it aligned with the specs. The up-to-date prototype will be also leveraged to actively contribute to the realization of a call flows document to be used by implementors as a useful practical reference associated with the inner workings of the CLUE protocol and framework. Finally, based on the consideration that our work actually integrates two different yet related contexts, namely telepresence and web real-time communication, we also plan to further foster the seminal cross-fertilization activities between the RtcWeb and the CLUE working groups within the IETF.

Acknowledgments

This work was partially funded by the Italian Ministry of Education, University and Research (MIUR) within the framework of projects PON01 01007 “PLATform for INnOative services in future internet” (PLATINO) and PON04a2_C (“SMART HEALTH’).

9. REFERENCES

[1] CLUE, ControLling mUltiple streams for tElepresence (IETF). http://tools.ietf.org/wg/clue/charters.

[2] Real-Time Communication in WEB-browsers (IETF). http://tools.ietf.org/wg/rtcweb/charters.

[3] Web Real-Time Communications Working Group Charter (W3C). http://www.w3.org/2011/04/webrtc-charter.html.

[4] A. Amirante, T. Castaldi, L. Miniero, R. Presta, and S. P. Romano. Standard multimedia conferencing in the wild: the Meetecho architecture. *Multimedia Tools and Applications, Springer*, September 2011.

[5] M. Barnes, L. Miniero, R. Presta, S. P. Romano, and H. Schulzrinne. CCMP: a novel standard protocol for conference management in the XCON framework. In *IPTComm'10*, pages 91–100, 2010.

[6] B. Deml. Human Factors Issues on the Design of Telepresence Systems. *Presence*, 16(5):471–487, Oct 2007.

[7] M. Duckworth, A. Pepperell, and S. Wenger. “Framework for Telepresence Multi-Streams” – Work In progress (Internet Draft). draft-ietf-clue-framework-16, June 2014.

[8] J. Edwards. Telepresence: Virtual Reality in the Real World [Special Reports]. *Signal Processing Magazine, IEEE*, 28(6):9–142, Nov 2011.

[9] J. Hautakorpi and G. Camarillo. The Session Description Protocol (SDP) Content Attribute. RFC 4796, February 2007.

[10] C. Holmberg. “CLUE Protocol Data Channel” – Work In progress (Internet Draft). draft-ietf-clue-datachannel-00, March 2014.

[11] W. A. Ijsselsteijn. *History of Telepresence*, pages 5–21. John Wiley & Sons, Ltd, 2006.

[12] P. Kyzivat, C. Groves, and R. Hansen. “CLUE Signaling” – Work In progress (Internet Draft). draft-ietf-clue-signaling-02, July 2014.

[13] S. Loreto and S. P. Romano. Real-Time Communications in the Web: Issues, Achievements, and Ongoing Standardization Efforts. *Internet Computing, IEEE*, 16(5):68–73, Sept 2012.

[14] R. Presta and S. P. Romano. “An XML Schema for the CLUE data model” – Work In progress (Internet Draft). draft-ietf-clue-data-model-schema-06, July 2014.

[15] R. Presta and S. P. Romano. “CLUE protocol” – Work In progress (Internet Draft). draft-ietf-clue-protocol-00, June 2014.

[16] A. Romanow, S. Botzko, and M. Barnes. Requirements for Telepresence Multistreams. RFC 7262, June 2014.

[17] A. Romanow, S. Botzko, M. Duckworth, and R. Even. Use Cases for Telepresence Multistreams. RFC 7205, April 2014.