

Full-Body WebRTC Video Conferencing in a Web-Based Real-Time Collaboration System

Matthias Wenzel, Christoph Meinel
 Hasso Plattner Institute Potsdam
 Prof. Dr. Helmert Str. 2-3, Potsdam, Germany
 Email: {firstname.lastname}@hpi.de

Abstract—Remote collaboration systems are a necessity for geographically dispersed teams in achieving a common goal. Real-time groupware systems frequently provide a shared workspace where users interact with shared artifacts. However, a shared workspace is often not enough for maintaining the awareness of other users. Video conferencing can create a visual context simplifying the user’s communication and understanding. In addition, flexible working modes and modern communication systems allow users to work at any time at any location. It is therefore desirable that a groupware system can run on users’ everyday devices, such as smartphones and tablets, in the same way as on traditional desktop hardware.

We present a standards compliant, web browser-based real-time remote collaboration system that includes WebRTC-based video conferencing. It allows a full-body video setup where everyone can see what other participants are doing and where they are pointing in the shared workspace. In contrast to standard WebRTC’s peer-to-peer architecture, our system implements a star topology WebRTC video conferencing. In this way, our solution improves network bandwidth efficiency from a linear to a constant network upstream consumption.

I. INTRODUCTION

Working teams, especially in corporate environments, are becoming increasingly distributed in time and space. It is a development that is certain to become more commonplace in the future. Collaboration with remote partners on shared tasks has grown accordingly. As a result, remote collaboration tools are well-established in today’s working environments. However, the availability of these tools across a wide range of devices has to be ensured. Remote collaboration tools have to be accessible from traditional computer systems, such as desktop computers and laptops, as well as from mobile devices, such as smartphones and tablets. Modern web browsers facilitate the development of HTML5 technology-based cross-platform software systems as capable and powerful as desktop applications [1].

Many groupware systems provide a shared workspace [2] (e.g. virtual whiteboards), allowing users to interact with shared task artifacts simultaneously from different locations. At the same time it is difficult to maintain the awareness of other users working in the shared workspace, mainly because only a small part of the perceptual information that is available in a face-to-face workspace is provided by groupware systems’ input and output devices [3], [4]. Video conferencing can increase this awareness, since a shared visual context can create a sense of co-presence [5]. This feeling of closeness improves

the communication and understanding of participants [6], [7]. Indicating objects with pointing gestures furthermore reduces the need for cumbersome verbal descriptions [8].

In order to combine video conferencing, which goes beyond a basic face-to-face setup with a flexible web browser-based groupware system, the video conferencing has to be run in a web browser context too. Compared to native groupware applications, this implies limitations regarding user interface (UI) layout (e.g. window transparency) and available video processing application programming interfaces (APIs).

In this paper we present our redeveloped real-time remote collaboration system Tele-Board [9]. The application is web browser-based and provides a shared workspace together with a video conferencing functionality. Furthermore, it is compliant with current web standards and does not rely on proprietary plugin technology. In order to increase the awareness of remote users and allow reference be made to shared artifacts, our system supports a full-body video setup as shown in Figure 1.

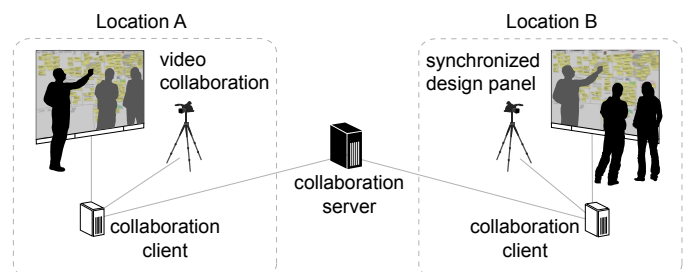


Fig. 1. Tele-Board video setup. A camera captures users standing in front of the workspace screen. Remote users see pointing gestures and facial expressions.

The translucent whiteboard can be displayed as an overlay on top of the full screen video of the other team members. This setup lets everyone see what the other participants are doing and where they are pointing at. Additionally, viewers see their gestures and facial expressions.

This way, Tele-Board provides three distinct types of spaces proposed by Buxton [10]: a person space of the remote participant’s image, a task space of involved artifacts on the whiteboard and a reference space for pointing and gesturing [9].

Though many groupware systems exist today, to our knowledge we are the first to propose a completely standard-

compliant web browser-based system that supports a full-body video setup.

II. RELATED WORK

In this section we focus on digital tools for remote collaboration. We provide an overview of relevant research ideas and prototypes as well as commercial and open source systems.

A. Research Projects

Two systems for remote collaboration at whiteboards developed in the early nineties were *VideoWhiteboard* [11], which is based on the *VideoDraw* [12] system, and *ClearBoard* [13]. In both systems, users and their drawings on a whiteboard are captured by a video camera. The video data is transferred synchronously to the remote location and projected onto the partner's screen. The VideoWhiteboard system was designed to only show the shadow and not a real video of the remote person (i.e. facial expressions were not transferred). VideoWhiteboard and ClearBoard did not support editing or erasure of the remote partner's whiteboard marks, since whiteboard content data was only projected and not really transferred to the other location.

VideoArms [14], [15] is a similar system which captures and reproduces participant's arms when working over large displays. While the system transmits a video showing hands and arms, facial expressions and full-body gestures were not transferred to the remote location.

A more recent system is *CollaBoard* [16]. CollaBoard provides full-body video in combination with a shared whiteboard workspace. The system relies on different native technologies (e.g. *Skype*¹) for audio/video conferencing. It requires a complex setup to use. As in all mentioned systems, native software (i.e. the platform specific code) has to be installed on both users' computer systems.

B. Commercial and Open Source Products

High quality telepresence systems, such as those offered by *Polycom*² or *Cisco*³ make it possible to build up a virtual meeting and convey the feeling of sitting together at the same table. However, dedicated rooms and expensive hardware entail limited access and make such systems primarily suitable for bigger companies. The drawback for creative work is the missing support of a shared workspace. A web-based tool for remote collaboration is *Adobe Connect*⁴. It can be used for document exchange and screen sharing and offers a simple whiteboard application. However, it relies on Adobe Flash plugin technology and does not offer full-body video support.

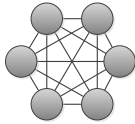
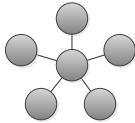
Two web browser-based systems are *Mural*⁵ and *RealtimeBoard*⁶. They offer a plugin free HTML5-based whiteboard application where users can manipulate shared artifacts. Mural

provides a text chat functionality for further interacting with remote participants but has no video support. RealtimeBoard provides a basic face-to-face video conferencing setup.

III. BROWSER-BASED VIDEO CONFERENCING

Traditionally, audio/video real-time communication within web browsers was only possible via plugins or third-party software. *Web Real-Time Communication* (WebRTC)⁷ is a collection of communication protocols and APIs that support peer-to-peer (P2P) real-time communication among web browsers. With the P2P capabilities introduced by WebRTC, browsers now break away from the classic client-server model. The advantage of this shift is that the APIs defined by WebRTC are the same regardless of the underlying browser, operating system etc. and are available on many platforms, especially mobile devices. In the context of video conferencing however, the P2P approach has a potential drawback as depicted in Table I. For n clients to interact with each other, $n(n-1) = n^2 - n \approx n^2$ transmission channels are needed (fully connected mesh). In order to improve this behavior, a central server can act in the same way as a peer. This way, the implementation of a star topology is possible. All clients connect to the central server which distributes the received media streams. This approach reduces client upstream, enabling large conferences (especially in face of asymmetric networks with considerably lower upstream than downstream bandwidth).

TABLE I
WEBRTC TOPOLOGY COMPARISON: STAR ARCHITECTURE HAS NETWORK BANDWIDTH ADVANTAGE OVER STANDARD WEBRTC P2P MESH

Topology	Client Upstream	Client Downstream
 <p>Fully Connected Mesh</p>	$n-1$ outgoing connections per client: each client sends its captured media data to all of its peers	$n-1$ incoming connections per client: each client receives media from all of its peers
 <p>Star Topology</p>	1 outgoing connection per client: each client sends captured media data to the central server	$n-1$ incoming connections: each client receives the $n-1$ media streams of all other clients from the central server (assuming no server side media mixing is performed)

A central server acting as a WebRTC remote peer to forward captured media data is often called *Selective Forwarding Unit* (SFU). Usually, it does not mix the received media data into a composite stream but only relays these streams to a set of participants.

In our implementation, we rely on an existing WebRTC framework. We had the following requirements to be fulfilled: (1) Open Source platform, so that we have full access to make necessary changes (2) star topology, since we did not want to rewrite the whole architecture (3) active development (4)

⁷<http://www.w3.org/TR/2015/WD-webrtc-20150210/>

¹<http://www.skype.com/>

²<http://www.polycom.com/hd-video-conferencing/realpresence-immersive-video-telepresence.html>

³<http://www.cisco.com/c/en/us/solutions/collaboration/telepresence/>

⁴<http://www.adobe.com/products/adobeconnect.html>

⁵<https://mural.ly/>

⁶<https://realtimeboard.com/>

code quality. Being the closest match to our requirements, we decided to use the open source SFU *Jitsi Videobridge*⁸ in our implementation.

IV. VIDEO CONFERENCING IN THE TELE-BOARD SYSTEM - APPLICATION DESIGN AND IMPLEMENTATION

With Tele-Board [17]–[19], we built a web browser-based real-time remote collaboration system. The application provides a shared workspace in terms of a virtual whiteboard surface. The content data is synchronized automatically by a central server (see Figure 2) among all connected client applications.

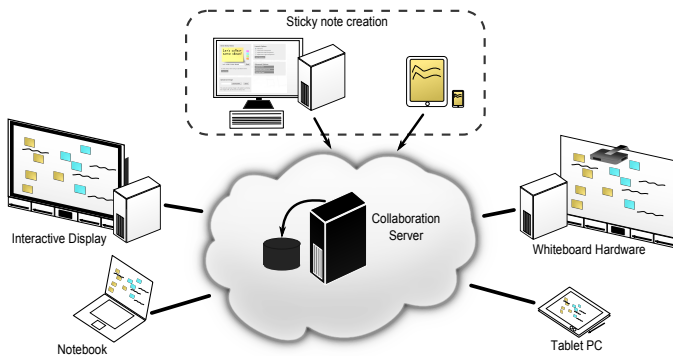


Fig. 2. The Tele-Board software system architecture. Web browser-based clients can be run on multiple devices. Virtual whiteboard data is synchronized among connected clients by the central collaboration server.

A. Tele-Board Architecture

The system mainly consists of three different parts. The *web portal* is the entry point of the Tele-Board system. It serves as an administration interface allowing users to manage projects and associated panels. In this way, users are able to organize their work and control access rights (see Figure 4). Users log into the system with their credentials. User accounts are assigned to projects which grants access to the projects’ panels. A panel represents a virtual whiteboard including its content and its course over time. This design permits the user to go back and forth in the history of the whiteboard content [20]. Started from the web portal, the *whiteboard client* allows editing of a panel. This application, written in JavaScript, requires no additional browser plugins. It facilitates whiteboard interaction (e.g. writing with different colors, erasing, and the manipulation of sticky notes and images). Whenever a user starts the whiteboard client with a particular panel, the system automatically becomes connected to all other clients operating on the same panel. All users can see remote panel actions in real-time and are equally authorized to manipulate any panel artifacts they choose. The *collaboration server* component coordinates all communication between the remote partners. Whenever a whiteboard artifact is created or changed, a serialized object representation in *JavaScript Object Notation* (JSON) is forwarded by the server to all

⁸<https://jitsi.org/Projects/JitsiVideobridge>

other connected whiteboards to keep these synchronized. In order to keep track of all whiteboard changes, our system performs synchronization in real-time (i.e. when moving a sticky note on the panel each change in its position is propagated to all connected clients). The remote participants can see the movement of the sticky note—not only its final position. We see the communication setup in Figure 3. Real-time communication causes a multitude of synchronization messages. Hence, we rely on the persistent, bi-directional *Transmission Control Protocol* (TCP)-based *WebSocket*⁹ protocol for keeping connected clients synchronized. We hereby build on the results and recommendations from Gutwin et al. [21]. The collaboration server is written in JavaScript using the server-side *Node.js*¹⁰ runtime environment. We use *Socket.IO*¹¹ JavaScript library for connection management and synchronization message relay (i.e. broadcasting to all clients participating in one panel session). A panel represents a virtual workspace in which content information is shared among a group of users who have access to this space. This concept of a chat room is realized by Socket.IO. Arbitrary channels, called “rooms”, can be defined that a user’s data connection may join and leave. In our system such a room is indicated by a panel’s identifier. This way, content data is only synchronized among the users who joined the room.

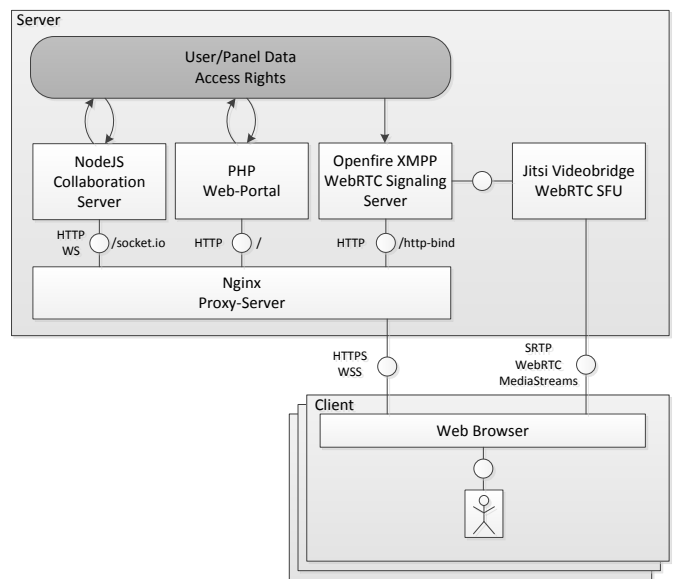


Fig. 3. The Tele-Board client-server architecture. Web portal, whiteboard synchronization and WebRTC signaling is managed by an Nginx proxy server using a single standard port 443 for encrypted communication.

With the web portal and the collaboration server we have two server-side services to be accessible by the client’s web browser. Traditionally, this would require two different ports on the server that also have to be reachable from the browser. This is sometimes difficult in client-side setups with restrictive

⁹<https://tools.ietf.org/html/rfc6455>

¹⁰<https://nodejs.org/en/>

¹¹<http://socket.io/>

firewalls. Thus, we are using the *Nginx*¹² proxy server that handles both, the web portal (HTTP) and the collaboration server (WebSocket) communication utilizing only one port. Therefore, the encrypted protocol versions *HTTP over SSL* (HTTPS) and *Secure WebSocket* (WSS) can be used with the standard port 443.

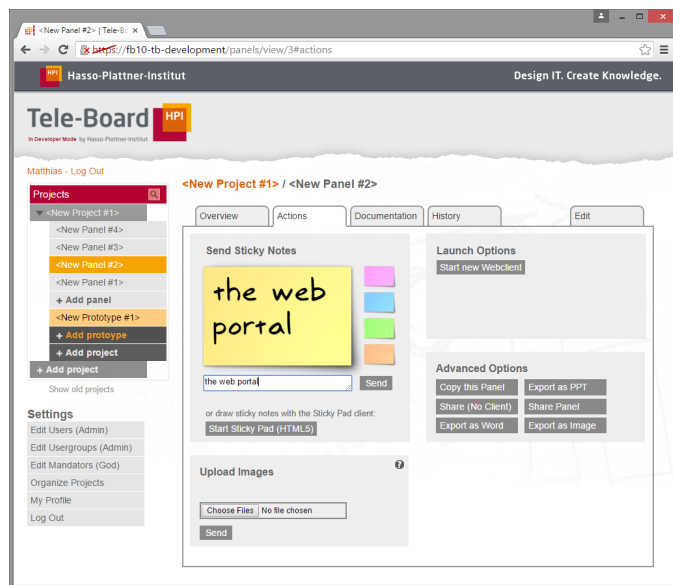


Fig. 4. The Tele-Board web portal. The left menu shows all projects and panels the user is assigned to.

In an earlier version of our system, the video part of the setup described in Figure 1 was provided by an external video conferencing system [9], such as Skype. This solution was only possible since our first whiteboard client was a Java application provided by the web portal via Java Webstart. In order to apply the full-body video setup, the whiteboard client could be started with a transparent background as an overlay for the video conferencing application. Due to the widespread usage of mobile devices such as smartphones and tablets where Java programs cannot be run, we chose to shift to a web browser-based whiteboard client implementation.

B. Video Conferencing Implementation

Using our web browser-based whiteboard client as a transparent overlay for an external video conferencing system is not possible. Relying on standards compliant JavaScript APIs, there is no access to web browser's window rendering (i.e. a transparent browser window). Consequently, a video conferencing functionality is only possible by using the web browsers' internal APIs (i.e. the above mentioned WebRTC).

Implementing WebRTC based video conferencing requires components on both server and client side. Though standard WebRTC media stream data is transferred directly between connected peers, in practice a server side part is still necessary. Establishing a connection between peers involves procedures to control communication and for metadata exchange. This

process of information exchange is called *signaling* and is not part of the WebRTC specification. Information exchange in particular encompasses messages regarding session control, network configuration, and web browsers' media capabilities. The application itself has to implement this mechanism which is typically provided by a server component.

1) *Server-Side Components*: The signaling mechanism for controlling Jitsi Videobridge SFU is based on *Extensible Markup Language* (XML) messages, that are exchanged using *Extensible Messaging and Presence Protocol*¹³ (XMPP) protocol. Jitsi Videobridge is an external XMPP component and therefore requires an XMPP server. Components can extend an XMPP environment's functionality. They receive all XMPP messages that are addressed to a particular subdomain of the XMPP server domain. For setting up a conference call (i.e. allocate channels for everyone, add or remove participants, and managing the call state) the open *XMPP extension protocol* (XEP) COLIBRI (CONferences with LIGHTweight BRIDging) is used in order to control the video bridge [22]. All these XMPP control messages are processed by the connected clients' web browsers using *Bidirectional-streams Over Synchronous HTTP*¹⁴ (BOSH) transport protocol for XMPP server communication. In our groupware system we deployed Openfire¹⁵ XMPP server. This communication scheme is shown in Figure 3. Once initial signaling process has finished, connected clients' audio/video media stream data is exchanged via Jitsi Videobridge, which relays the data to connected web browsers. This way, we realize the intended star topology in our system.

Starting the whiteboard client and connecting to all remote participants automatically, a video conference has to be established automatically without any further user interaction. At the same time it has to be ensured that only users can take part in a conference that also have access to the corresponding panel. Therefore, we had to extend the video conferencing management by an access control mechanism. We implemented an automatic, Single Sign-On (SSO) related authentication mechanism for the video conferencing signaling server on the basis of user's web portal credentials. Multiparty video conferencing with Jitsi relies on a chat room concept very similar to the panel room mechanism with Socket.IO described in section IV-A. Multiple users can join a conference defined by a shared room name. The XMPP *Multi-User Chat* (MUC) protocol extension XEP¹⁶ is used by Jitsi's signaling server for setting up multiparty video conferencing, and allowing closed, password protected rooms.

The Openfire XMPP signaling server can be used stand-alone (i.e. it has its own authentication mechanism based on its own database where before created user accounts are stored). Our intention was not to duplicate this authentication data. Instead, we use the existing database the web portal and the server component are operating on. Openfire provides Java APIs for extending its functionality. This way, we could adapt

¹²<http://nginx.org/>

¹³<https://tools.ietf.org/html/rfc6120>

¹⁴<http://xmpp.org/extensions/xep-0124.html>

¹⁵<http://www.igniterealtime.org/projects/openfire/>

¹⁶<http://xmpp.org/extensions/xep-0045.html>

the server's user authentication and room access control to our needs. In our implementation we created two extensions for the used Openfire XMPP server:

Server Authentication Provider - A library for authenticating users by their web portal user credentials. We configured the XMPP server to only allow specified login because it is an exclusively private service. We hereby use the shared web portal database for user credential verification. Our library consists of a class that implements an interface providing an `authenticate` method with a username and password signature which performs a custom authentication procedure. The compiled jar file has to be included in an Openfire folder. Afterwards, the system can be configured to use the custom authentication provider.

Conference Room Authenticator - Openfire allows creating plugins that can be uploaded to the server. These plugins can extend the server's functionality. A multitude of plugins are available on the Openfire website. We implemented a plugin that listens for MUC events such as room creation or joining and leaving a room. To check if a user is allowed to join a conference room, we provided the appropriate method that is called by the system with a given room and user name. Based on our database we can verify if the user has permission to access the room defined by a Tele-Board panel's identifier.

On the basis of these mechanisms, we establish video conference calls transparently in our system. The conference starts automatically once a user opens our web browser based whiteboard client application. Furthermore, providing a star topology conferencing increases user's client-side bandwidth resource efficiency.

2) *Client-Side Implementation*: Enabling full-body video setup was our main goal during development. However, for use cases such as desktop video conferencing in a face-to-face style or when a full-body arrangement is too complex, a more basic setup is desirable. This way, we wanted the arrangement of the video to be as flexible as possible while not interfering with the whiteboard client's panel surface. As a result, we decided on a solution that supports multiple web browser windows where video conferencing display can be detached from the whiteboard client window. The video conferencing is implemented as a single page application in our web portal in the same way as the whiteboard client. Following this approach, video conferencing can be attached to the whiteboard client window embedded via an `iframe`¹⁷ HTML markup element in the background of the panel surface or as own window detached from the whiteboard client. The video conference is initiated on-the-fly on the whiteboard client application start-up. The `iframe` element or the detached window is created by the client-side JavaScript loading the video conferencing web page from the web portal server. With the reference on this web page, video conferencing can be controlled by the whiteboard client (i.e. changing the views, volume control, switching audio/video on and off). When em-

¹⁷<http://www.w3.org/TR/html5/embedded-content-0.html#the-iframe-element>

bedded via `iframe` for applying full-body setup, the whiteboard panel surface is rendered with a transparent background on top of the video layer. The result of this technique is shown in Figure 5. The panel artifacts are rendered on top of the video enabling the remote user to point at a specific element on the panel.

The Jitsi system contains the client-side JavaScript component *Lib-Jitsi-Meet*¹⁸ that manages XMPP messaging as well as audio/video handling in the web browser. The component provides an API that can be used in web browser based applications. We used this component in our video conferencing application page. Once the page is loaded the API provides access to the local web browser's audio/video media sources such as webcam and microphone. When the user approves access to media devices, the connection to Jitsi Videobridge WebRTC server is established. Furthermore, the API defines a multitude of mandatory registration events (e.g., upon joining or leaving certain remote users conferences and remote media stream changes). This definition allows a flexible application design since conference management and media stream handling is separated from the video conference's actual visual representation. In our system, we support multiple remote conference participants in the detached window mode. Remote users' videos are rendered as small preview video elements. The local user can switch to a larger display by selecting the respective video area. In the intended full-body use case setup, the whiteboard panel can only be aligned with one remote video. Thus, in the attached full-window video mode, we currently show only one remote participant in the background.

The method of splitting whiteboard client and video conferencing application into two web pages can, we believe, best adapt to different camera positions and use cases. At the same time, it preserves our primary objective of the use of full-body video.

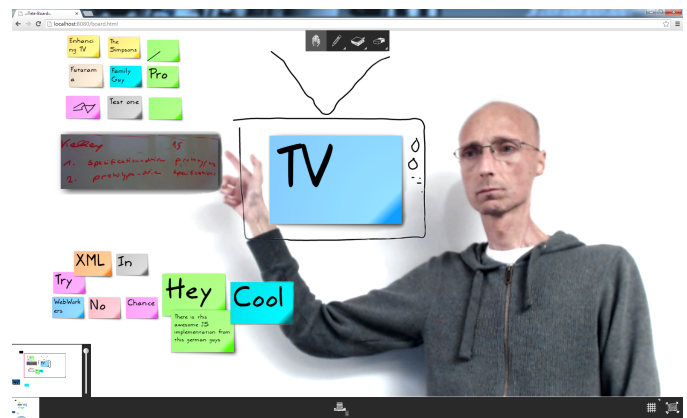


Fig. 5. Tele-Board full-body video conferencing. Users can point to shared artifacts on the workspace.

V. CONCLUSION AND OUTLOOK

It can be difficult to follow remote team members' shared workspace interactions in a real-time groupware system. Re-

¹⁸<https://github.com/jitsi/jitsi-meet/tree/lib-jitsi-meet>

remote team members may have the impression that whiteboard artifacts are moving by themselves when these are manipulated locally. This kind of remote whiteboard interaction can benefit from a video conference that shows how other team members operate the system [23]. In this paper we presented a video conferencing solution integrated in our real-time remote collaboration system—Tele-Board. System design allows a full-body video setup in order to align the remote team member with the shared workspace artifacts on a virtual whiteboard surface. This visual context lets everyone see what other participants are doing and where they are pointing. Additionally, their gestures and facial expressions can be seen. Compared to prior research and existing commercial systems, our system is web browser-based and compliant with current web standards and does not rely on plugin technology. Hence, it is a cross-platform application that can be run on desktop computers as well as on mobile devices, such as smartphones and tablets. In contrast to standard WebRTC's P2P approach, our WebRTC based video conferencing component implements a star topology. The user's network upstream bandwidth acquisition is therefore reduced to a constant value. This means it is independent of the number of participants, instead of being linearly related to the number of conferencing members.

There is another advantage to the audio/video streams being handled by a central server. It is possible to record the video sessions at a central location. Otherwise, with P2P this would be impracticable and require much greater effort. We are currently working on implementing a technique to record the video sessions and align these sessions with whiteboard content history. Tele-Board already provides a history browser interface giving the opportunity to go back and forth in the timeline of a whiteboard. This combination of content history with "audio/video" history would allow better asynchronous work comprehension [24].

ACKNOWLEDGMENT

We thank the HPI-Stanford Design Thinking Research Program for funding and supporting this project.

REFERENCES

- [1] A. Wright, "Ready for a Web OS?" *Communications of the ACM*, vol. 52, no. 12, pp. 16–17, Dec. 2009.
- [2] C. A. Gutwin, S. Greenberg, R. Blum, J. Dyck, K. Tee, and G. Mcewan, "Supporting Informal Collaboration in Shared-Workspace Groupware," *Journal of Universal Computer Science*, vol. 14, no. September, pp. 1411–1434, 2008.
- [3] C. A. Gutwin and S. Greenberg, "A Descriptive Framework of Workspace Awareness for Real-Time Groupware," *Computer Supported Cooperative Work (CSCW)*, vol. 11, no. 3-4, pp. 411–446, sep 2002.
- [4] —, "The importance of awareness for team cognition in distributed collaboration." in *Team cognition: Understanding the factors that drive process and performance.*, Washington, 2004, pp. 177–201.
- [5] J. R. Brubaker, G. Venolia, and J. C. Tang, "Focusing on Shared Experiences: Moving beyond the camera in video communication," in *Proceedings of the Designing Interactive Systems Conference*, ser. DIS '12. New York, NY, USA: ACM, 2012, pp. 96–105.
- [6] E. Koh, "Conferencing room for telepresence with remote participants," *Proceedings of the 16th ACM international conference on Supporting group work - GROUP '10*, p. 309, 2010.

- [7] E. A. Isaacs and J. C. Tang, "What video can and can't do for collaboration," in *Proceedings of the first ACM international conference on Multimedia - MULTIMEDIA '93*. New York, NY, USA: ACM Press, 1993, pp. 199–206.
- [8] S. Fussell, L. Setlock, J. Yang, J. Ou, E. Mauer, and A. Kramer, "Gestures Over Video Streams to Support Remote Collaboration on Physical Tasks," *Human-Computer Interaction*, vol. 19, no. 3, pp. 273–309, 2004.
- [9] R. Gumienny, L. Gericke, M. Quasthoff, C. Willems, and C. Meinel, "Tele-Board: Enabling Efficient Collaboration in Digital Design Spaces," in *Proceedings of the 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2011)*. IEEE Press, June 2011, pp. 47–54.
- [10] B. Buxton, "Mediaspace Meaningspace Meetingspace," in *Media Space 20 + Years of Mediated Life*, ser. Computer Supported Cooperative Work, S. Harrison, Ed. Springer London, 2009.
- [11] J. C. Tang and S. Minneman, "VideoWhiteboard: video shadows to support remote collaboration," in *Proceedings of the SIGCHI conference on Human factors in computing systems Reaching through technology - CHI '91*. New York, NY, USA: ACM Press, 1991, pp. 315–322.
- [12] J. C. Tang and S. L. Minneman, "Videodraw: a video interface for collaborative drawing," *ACM Transactions on Information Systems (TOIS)*, vol. 9, no. 2, pp. 170–184, 1991.
- [13] H. Ishii and M. Kobayashi, "ClearBoard: a seamless medium for shared drawing and conversation with eye contact," in *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '92*. New York, NY, USA: ACM Press, 1992, pp. 525–532.
- [14] A. Tang, C. Neustaedter, and S. Greenberg, "VideoArms: Supporting Remote Embodiment in Groupware," in *Video Proceedings of the ACM CSCW Conference on Computer Supported Cooperative Work*. Chicago, IL, USA: ACM Press, November 2004.
- [15] —, "VideoArms: Embodiments for Mixed Presence Groupware," in *Proceedings of the 20th British HCI Group Annual Conference (HCI 2006)*, September 2006, pp. 85–102.
- [16] A. Kunz, T. Nescher, and M. Kuchler, "CollaBoard: A Novel Interactive Electronic Whiteboard for Remote Collaboration with People on Content," *Cyberworlds (CW), 2010 International Conference on*, pp. 430–437, 2010.
- [17] M. Wenzel, L. Gericke, R. Gumienny, and C. Meinel, "Towards Cross-Platform Collaboration - Transferring Real-Time Groupware To The Browser," in *Proceedings of the 17th IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD 2013)*. IEEE, June 2013, pp. 49–54.
- [18] M. Wenzel and C. Meinel, "Parallel network data processing in client side javascript applications," in *2015 International Conference on Collaboration Technologies and Systems (CTS 2015)*. IEEE, June 2015, pp. 140–147.
- [19] M. Wenzel, L. Gericke, C. Thiele, and C. Meinel, "Globalized design thinking: Bridging the gap between analog and digital for browser-based remote collaboration," in *Design Thinking Research*, ser. Understanding Innovation, H. Plattner, C. Meinel, and L. Leifer, Eds. Springer International Publishing, 2016, pp. 15–33.
- [20] L. Gericke, R. Gumienny, and C. Meinel, "Message capturing as a paradigm for asynchronous digital whiteboard interaction," in *Proceedings of the 6th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2010)*. Chicago, IL, USA: IEEE Press, 10 2010, pp. 1 – 10.
- [21] C. A. Gutwin, M. Lippold, and T. C. N. Graham, "Real-time groupware in the browser: Testing the performance of web-based networking," in *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*, ser. CSCW '11. New York, NY, USA: ACM, 2011, pp. 167–176.
- [22] B. Grozev, L. Marinov, V. Singh, and E. Ivov, "Last N: relevance-based selectivity for forwarding video in multimedia conferences," in *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video - NOSSDAV '15*. New York, NY, USA: ACM Press, 2015, pp. 19–24.
- [23] S. R. Fussell, R. E. Kraut, and J. Siegel, "Coordination of communication: Effects of shared visual context on collaborative work," in *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, ser. CSCW '00. New York, NY, USA: ACM, 2000.
- [24] L. Gericke, M. Wenzel, and C. Meinel, "Asynchronous understanding of creative sessions using archived collaboration artifacts," in *2014 International Conference on Collaboration Technologies and Systems (CTS)*. IEEE, may 2014, pp. 41–48.