# USE Together, a WebRTC-Based Solution for Multi-user Presence Desktop

Laurent Lucas, Hervé Deleau, Benjamin Battin, Julien Lehuraux

# USE Together, a WebRTC-based Solution for Multi-User Presence Desktop

Laurent Lucas[1], Hervé Deleau[2,1], Benjamin Battin[3,1], and Julien Lehuraux[3,1]

[1] University of Reims Champagne-Ardenne, CReSTIC, France,
laurent.lucas@univ-reims.fr
[2] University of Reims Champagne-Ardenne, Image Center MaSCA, France,
[3] OPEXMedia, France

**Abstract.** Ubiquitous is one of the essential features of what should be the desktop of the future. In practice, this concept covers several issues related to multi-users collaboration, remote applications control or remote display and secure access over IP networks. With its standards and capabilities, WebRTC provides a new vision of real-time communications services that can raise these challenges. In this paper we present a WebRTC-based middleware solution for real-time multi-users remote collaboration. It allows a full desktop setup where everyone can see what other users are doing and where they position themselves in the shared workspace. In contrast to standard WebRTC's Peer-to-Peer architecture, our system supports a synchronous communication model through a star topology. It also improves network bandwidth efficiency by using hardware video compression when the GPU resource is available, though assuring a very low latency streaming. In this way, we can maintain awareness and sense of presence without changing the usual practices of the users in front of a desktop. Several use cases are provided and a comparison of advantages and drawbacks of this solution is also presented to guide users in applying this technology under real-life conditions.

**Keywords:** WebRTC, Remote Display, Multi-Users Collaborative Environment

## 1 Introduction

The requirements related to teamwork and mobility especially in corporate environments as well as in science and academic environments are becoming increasingly requested. This new way of working on spatially and temporally distributed systems has become a more commonplace practice especially with the emergence of remote collaboration tools allowing a group of people to share their resources or to create in a common effort. In this sense and for a growing range of devices, the availability of these tools has to be ensured particularly in terms of security and accessibility, for instance, from traditional computer as well as from mobile devices like smartphones and/or tablets. Web technologies through modern capabilities of browsers enable today the development of cross-platform

software systems as capable and powerful as desktop applications [1]. From this point of view, the Web has opened a new way for the development of cloud hosted Internet-based collaboration apps [2] and other means of interaction. Online collaboration tools can be classified in two categories: synchronous vs. asynchronous communication tools. Unlike asynchronous communication, synchronous communication involves an ongoing real time character and can take place face-to-face irrespective of distance. Although this distinction tends to fade, the feeling of presence has become crucial in all collaborative environments [3,4] especially with the recent development of immersive collaborative solutions [5,6]. This observation has been partly achieved by the widespread use of both HPC and graphics virtualization that has brought significant changes to corporate networks by delivering for instance an immersive, high-quality user experience for everyone [7], from designers [8] to engineers [9] and other mobile professionals or simple office workers. This technological innovation used widely in many industrial sectors is one of the most disruptive of our time.

However, if current software solutions partially and specifically address – e.g. in terms of online collaborative work, video conferencing, multi-users remote control or remote display – the issues raised by the "desktop of the future" [10], it must be noted that *i)* there is no integrated environment today around all of these elements and *ii)* data privacy is not always guaranteed which can be a serious problem of sovereignty for all strategic institutions.

Whether they are research projects as well as commercial systems, there are many collaborative online solutions used today in areas such as health [11], collaborative visualization [3,4,8] and learning [12,13] with specific software developments related to whiteboarding collaboration [14,15,16] for instance.

Regarding commercial products, software market can be segmented into three fields: *i)* online collaborative solutions first such as those offered by Cisco with Spark or Amazon with Chime, *ii)* multi-users remote control next with Screenhero and *iii)* remote visualization after all through solutions like Citrix HDX 3D Pro, HP RGS or Nice DCV.

In this paper we present our solution called USE Together. This middleware is a secure multi-user collaborative system allowing professionals to share their applications and data in real time, accessible from any device, over any network. It enhances your communications in terms of *i)* user QoE by delivering HD in real time with low latency, *ii)* simplicity of use based on standards such as WebRTC and HTML5 with zero-client deployment *iii)* security without data transmission but only pixel on a Peer-to-Peer architecture with encrypted streams and *iv)* flexibility of use by supporting both SaaS, on-premises and host-to-host deployment modes. Our contribution is based on the hybridization of solutions supporting native web access, GPU encoding and multi-cursor management, summarizing the best of both world. The reminder of this paper is organized as follows: in section 2 we propose a brief overview of the main functionalities of WebRTC before introducing, in section 3, our contribution USE Together and its architecture. Then we present and discuss in sections 4 and 5 some use cases and their performance. Finally, conclusion and future works are given in section 6.

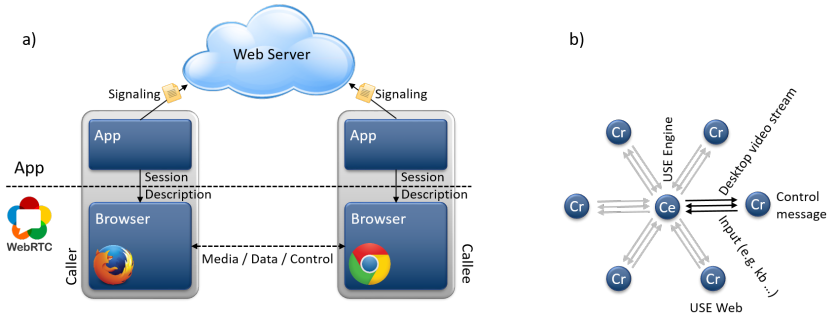


**Fig. 1.** a) WebRTC system architecture and b) peers connection topology – star – where a callee (noted Ce) sends captured media to each caller (noted Cr) which, in turn, transmit their inputs with transactions of control messages in both cases.

## 2 WebRTC

WebRTC (Web Real Time Communication) [17] is a technology that allows real-time Peer-to-Peer communication between browsers without the use of additional plugins. WebRTC is designed "to enable rich, high-quality RTC applications to be developed for the browser, mobile platforms, and IoT devices, and allow them all to communicate via a common set of protocols" [18]. WebRTC was open-sourced by Google in 2011 and after that an ongoing work started to standardize the protocols associated with it by IETF and its browser APIs by W3C. Interest and support for WebRTC has been since growing steadily. Today, the most advanced WebRTC implementation is offered by Mozilla Firefox and Google Chrome and includes three APIs:

1. MediaStream, which allows an application to stream media from the users web camera and microphone or from a screen capturing.
2. DataChannel, which allows to share arbitrary data between peers. This layer is an important feature of WebRTC allowing the development of all kind of Peer-to-Peer applications and collaborative solutions.
3. PeerConnection, which represents the glue between MediaStream and DataChannel by providing a handshake mechanism for two machines to exchange necessary information so a Peer-to-Peer connection can be set up.

The architecture of WebRTC including the signaling server is shown in schematic 1. Although WebRTC aspires to enable Peer-to-Peer communication between browsers without relaying data through any intermediary, the use of a server is still required for two reasons: the first reason is the obvious one, a web server is needed to serve the actual web application that utilizes WebRTC. The second reason is less obvious. A server is required in order to initialize sessions between the clients that need to communicate. This process is known as Signaling and is responsible for the exchange of the initial (meta) data of session descriptions (using SDP and ICE framework) which contain details on the form and nature

of the data which will be transmitted [24]. These information can include network data, such as IP addresses and ports, media metadata such as codecs and codec settings, bandwidth and media types, error messages or user and room information. PeerConnection API is used to achieve this process.

## 3 USE Together overview and implementation

Based on the native C++ APIs implementation of WebRTC by Google, USE Together is structured around two modules: USE Signaling and USE Engine. The implemented and developed solution with all its elements with respect to the architecture is illustrated in figure 1.a. In the two following subsections, to give a better understanding of the overall architecture to the reader, we will illustrate the description of each module with a typical usage scenario: a user $A$ starts a collaborative working session $\mathcal{S}$ on his desktop and a user $B$ wants to join $\mathcal{S}$.

### 3.1 USE Engine

The USE Engine module consists in two major sub-systems: the former, called 'USE Engine Core', acts as the central point of communication between the host (which initiates the collaborative session) and the remote users who join it. In terms of network topology (cf. Figure 1.b), one can see a collaborative session as a star where the host is located in the center and each remote user resides in a branch. Thus, the 'USE Engine Core' part is essentially dedicated to receiving and delivering data to each branch over WebRTC channels: video and audio streaming (resp. input and control messages) over Media Channels (resp. Data Channel). The latter sub-system, named 'USE Engine GUI', is an application responsible for the following tasks: *i)* capturing an entire desktop or a specific window, *ii)* capturing local video and/or audio data (eg. from a webcam), *iii)* encoding the resulting streams and transmitting it to 'USE Engine Core', *iv)* injecting keyboard and mouse input events from remote peers and *v)* specifying multiple settings to configure the session.
When the user $A$ wants to start a collaborative working session, he just starts USE Engine, which automatically creates a working session $\mathcal{S}$ and registers it on USE Signaling (described below). The session is now active and can be reachable by any remote user who knows the session name and the session password.
In order to provide the best possible experience to the user, USE Engine especially focuses on addressing two typical issues related to collaborative softwares: latency and multiple user inputs management. With traditional remote desktop visualization tools, the user generally has to deal with high latency which could be annoying while using real time applications remotely. USE Engine exploits the latest technologies in terms of screen capturing and video encoding respectively with the use of the NVIDIA's GRID and NVENC APIs. The first one, (GRID), provides direct access to video memory while NVENC makes use of a hardware H.264 encoding chip, integrated since the release of Kepler NVIDIA

GPUs, to produce a low latency H.264 video stream. Obviously, if the desktop is not equipped with such hardware, a fallback mode provides a desktop capture system based on OS APIs and a CPU encoding framework delivering either an H.264 (still with a low latency profile) or a VP8 video stream. The last issue lies in the input events handling of each connected user on an operating system natively thought for a single usage. To that end, USE Engine includes two interaction modes: a synchronized one, where a user can seamlessly take the control anytime he does a specific action (mouse clicks or keyboard usage) and if nobody already did, ignoring the other users input events for the duration of those actions, and a token-based one where a user has the control as long as he keeps the token (set by the session administrator).

### 3.2 USE Signaling

As mentioned in section 2, an auxiliary server, which acts both as a web server and as a signaling server, is required to set up the Peer-to-Peer communication between user $A$ and user $B$. Firstly, user $B$ has to log himself, then specify the session name and the associated password. USE Signaling is then able, from the session name, to identify the user who initiates the collaborative session (in our case, user $A$) and to relay messages between $A$ and $B$ during the signaling stage. Signaling can be defined as a classic handshaking phase during which the two users exchange network information (to find the best network route between them) and their session descriptions (a data structure containing streaming capabilities of a specific machine/browser couple) to negotiate a compatible way to exchange data. As soon as the negotiation is done, the peer connection (and the associated communication channels) can be created between $A$ and $B$. At this point, $B$ is now connected to $\mathcal{S}$ and can work collaboratively with user $A$.

## 4 Use cases description and discussion

Two kind of use cases have been realized with a common objective to stay focused on what is essential to application area by centralizing data and applications for a remote multi-peer collaborative access.
For manufacturing industries case first, USE Together has been used as a project management tool to enable its users to work remotely with different CAD applications. Project review, synchronous co-design, simulation and visualization are the main functions tested in a multi-user collaborative framework. As we can see in figure 2, four users interact synchronously on a same 3D model during a project review phase. The second use case was carried out within a biomedical environment with different softwares visualization. Mainly based on GPU-accelerated direct volume rendering algorithms, these tests confirmed the compatibility of the system with GPU-intensive resources applications without altering facility to encode the output video stream in real time. Several scenarios have been designed to work remotely with different partners on a collegial basis in order i) to jointly annotate and navigate in a set of biological data obtained through

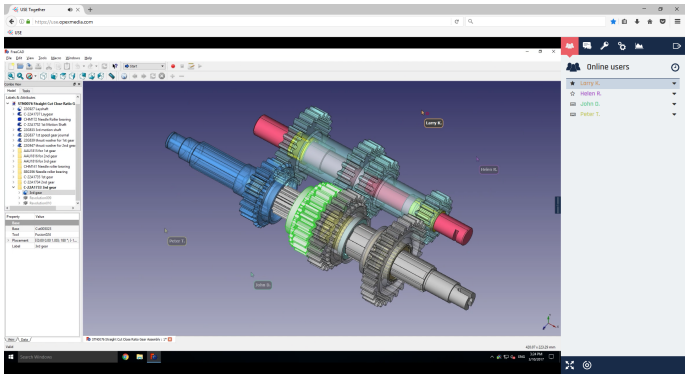


**Fig. 2.** Example CAD viewer application. The actual image shown on screen is being rendered remotely. The four users connected to their browsers can interact simultaneously on the 3D model.

a slide scanner and ii) to engage HPC resources to visualize and interact with simulations remotely.
In both cases, USE Together has received a large endorsement by:

- increasing users' productivity on load-intensive applications and complex data through remote access on centralized resources.
- enhancing performance of teams with a real time collaborative solution running on a same application instance.

## 5 Performance analysis

In order to test our solution, different experiments over several hundred kilometers between the server and three simultaneously connected clients were conducted. All these results are reported on table 1, which also includes the specifications of the various materials used. On the server side, we used a virtual machine (VM) equipped with an Intel Xeon E5-2650v2 @ 2.60GHz (8 cores), 32 GB of RAM and a NVidia GRID K2 of 4 GB of VRAM mounted in PCI Passthrough as GPU. This VM runs on Windows 7 Pro with a desktop resolution configured in HD. We used both the Unigine Valley Benchmark and FreeCAD workload to simulate real user behavior and/or monitored the following user experience and scalability metrics in fullscreen for three kinds of image quality setting (Low/Medium/High) with NVIDIA GPU based H.264 encoding (NVENC high performance low latency preset). On the client side, three device types were used with different network accesses for each of them (see Table 1). All these elements show that USE Together has achieved to bring a smooth experience on both of the use cases over any network, from 3G/4G to Wifi and Ethernet, with a mean bitrate of about 3.2 MB/s for a full HD remote display.

**Table 1.** Performance comparison on three terminal types and network connections.

| **Specs terminal** | **Network connection** | **Packets recvd(K)/lost** L/M/H | **Bytes recvd (Mo)** L/M/H | **Mean bitrate (Mb/s)** L/M/H | **FPS** L/M/H |
|---|---|---|---|---|---|
| #1 Desktop Quadro M4000 | ADSL RJ45 | 64/110/258 6/96/469 | 63/111/280 | 1.81/3.12/7.88 | 30/30/31 |
| #2 Laptop Intel HD4000 | ADSL Wifi | 75/110/265 38/551/8500 | 72/115/295 | 1.99/3.22/7.95 | 22/23/22 |
| #3 Tablet Tegra K1 | 4G | 76/150/247 11/72/394 | 73/157/270 | 1.92/4.31/7.30 | 21/18/14 |

## 6 Conclusion and future works

This paper proposed a WebRTC-based collaborative multi-user solution enhancing communications of a group by enabling them to share their applications and data in real time over any network. This solution called USE Together can be deployed on various hardware environments in a secure way and be accessed through a simple web browser without using any additional software nor plugin. Composed of two modules allowing *i)* to connect two peers and *ii)* to exchange encrypted streams between peers, USE Together is able to address many challenges in relation to pervasive computing like capabilities to offer interactive shared workspaces in a collaborative way and to maintain calculation accessibility through "invisible" resources while guaranteeing a good level of confidentiality during exchanges. Exclusively based on a web implementation today, this solution should also evolve to provide end-point devices support like specific 3D displays and VR/AR devices.

## Acknowledgment

This work is supported by the French national funds (PIA2'program) under contract No. P112331-3422142 (3DNS project).

## References

1. Alex Wright. Ready for a web os? *Commun. ACM*, 52(12):16–17, December 2009.
2. Jay F. Nunamaker, Robert O. Briggs, and Nicholas C.R. Romano. *Collaboration Systems: Concept, Value, and Use*. Taylor & Francis, 2015.
3. Petra Isenberg, Niklas Elmqvist, Jean Scholtz, Daniel Cernea, Kwan-Liu Ma, and Hans Hagen. Collaborative visualization: Definition, challenges, and research agenda. *Information Visualization Journal*, 10(4):310–326, 2011. Published online before print July 29, 2011.

4. Christophe Mouton, Kristian Sons, and Ian Grimstead. Collaborative visualization: Current systems and future trends. In *Proc. of the 16th Int. Conf. on 3D Web Technology*, Web3D '11, pages 101–110, New York, NY, USA, 2011. ACM.
5. Hank Childs, Berk Geveci, Will Schroeder, Jeremy Meredith, Kenneth Moreland, Christopher Sewell, Torsten Kuhlen, and E. Wes Bethel. Research challenges for visualization software. *Computer*, 46(5):34–42, May 2013.
6. Elena Zudilova-Seinstra, Tony Adriaansen, and Robert van Liere. *Trends in Interactive Visualization: State-of-the-Art Survey*. Springer Publishing Company, Incorporated, 1 edition, 2008.
7. Petru Nicolaescu, Kevin Jahns, Michael Derntl, and Ralf Klamma. *Yjs: A Framework for Near Real-Time P2P Shared Editing on Arbitrary Data Types*, pages 675–678. ICWE '15. Springer International Publishing, June 2015.
8. Caroline Desprat, Hervé Luga, and Jean-Pierre Jessel. Hybrid client-server and p2p network for web-based collaborative 3d design. In *Proceedings of the 23rd Int. Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, WSCG '15, pages 229–238, 2015.
9. Lirong Wang, Jiacai Wang, Lixia Sun, and Ichiro Hagiwara. A peer-to-peer based communication environment for synchronous collaborative product design. In *Proc. of the 4th International Conf. on Cooperative Design, Visualization, and Engineering*, CDVE '07, pages 9–20, Berlin, Heidelberg, 2007. Springer-Verlag.
10. Rodrigo Pizarro, Mark Hall, Pablo Bermell-Garcia, and Mar Gonzalez-Franco. Augmenting remote presence for interactive dashboard collaborations. In *Proceedings of the Int. Conference on Interactive Tabletops & Surfaces*, ITS '15, pages 235–240, New York, NY, USA, 2015. ACM.
11. Linh Van Ma, Jisue Kim, Sanghyun Park, Jinsul Kim, and Jonghyeon Jang. An efficient session_weight load balancing and scheduling methodology for high-quality telehealth care service based on webrtc. *The Journal of Supercomputing*, 72(10):3909–3926, 2016.
12. Michalis Xenos, Nikolaos Avouris, Vassilis Komis, Dimitris Stavrinoudis, and Meletis Margaritis. Synchronous collaboration in distance education: A case study on a computer science course. In *Proc. of the IEEE Int. Conf. on Advanced Learning Technologies*, ICALT '04, pages 500–504, Washington, DC, USA, 2004. IEEE Computer Society.
13. Ilya V Osipov, Alex A Volinsky, and Anna Y Prasikova. E-learning collaborative system for practicing foreign languages with native speakers. *Int. Journal of Advanced Computer Science and Applications*, 7(3):40–45, 2016.
14. Adham Zeidan, Armin Lehmann, and Ulrich Trick. Webrtc enabled multimedia conferencing and collaboration solution. In *Proceedings of the World Telecommunications Congress 2014*, WTC '14, pages 1–6, June 2014.
15. Matthias Wenzel and Christoph Meinel. Full-body webrtc video conferencing in a web-based real-time collaboration system. In *Proc. of the 20th IEEE Int. Conf. on Comp. Supported Cooperative Work in Design*, CSCWD '16, pages 334–339, 2016.
16. Nikos Pinikas, Spyros Panagiotakis, Despina Athanasaki, and Athanasios Malamos. Extension of the webrtc data channel towards remote collaboration and control. In *Proceedings of the Int. Symposium on Ambient Intelligence and Embedded Systems 2016*, AmiEs '16, 2016.
17. Ilya Grigorik. *High Performance Browser Networking: What every web developer should know about networking and browser performance*. O'Reilly Media, Inc., 2013.
18. WebRTC [online]. `https://webrtc.org/`. [Accessed 2017-03-07].