International Journal of Information and Communication Sciences
2017; 2(5): 59-67
http://www.sciencepublishinggroup.com/j/ijics
doi: 10.11648/j.ijics.20170205.12
ISSN: 2575-1700 (Print); ISSN: 2575-1719 (Online)

A Device Independent Platform for Synchronous Internet of Things Collaboration and Mobile Devices Screen Casting

Nikos Pinikas, Spyros Panagiotakis, Despina Athanasaki, Athanasios G. Malamos*

Department of Informatics Engineering, Technological Educational Institute of Crete, Heraklion, Greece

Email address:

npinikas@hotmail.com (N. Pinikas), spanag@teicrete.gr (S. Panagiotakis), despinadev@gmail.com (D. Athanasaki), amalamos@ie.teicrete.gr (A. G. Malamos)

*Corresponding author

To cite this article:

Nikos Pinikas, Spyros Panagiotakis, Despina Athanasaki, Athanasios G. Malamos. A Device Independent Platform for Synchronous Internet of Things Collaboration and Mobile Devices Screen Casting. *International Journal of Information and Communication Sciences.* Vol. 2, No. 5, 2017, pp. 59-67. doi: 10.11648/j.ijics.20170205.12

Received: April 10, 2017; **Accepted**: May 24, 2017; **Published**: October 23, 2017

Abstract: WebRTC is project that allows browser-to-browser voice, video and data communication without the use of plugins. It enables rich, high quality, Real Time Communications applications to be developed for the browser, mobile platforms, and Internet of Things (IoT) devices, and allows them all to communicate via a common set of protocols. In this paper we employ the capabilities of the WebRTC APIs to implement a platform for synchronous collaboration, screen casting and multimedia communication.

Keywords: WebRTC, Online Collaboration, Screen Casting, Collaborative Browsing

1. Introduction

Tools that aid collaboration were around some time before Computers - whiteboards, flipcharts or even a bit of paper can be utilized to bolster joint effort [1]. PCs and the Web altered the way individuals cooperate in gatherings. In the 80s the expression "groupware" was begat by C. A. Ellis who characterized it as a "PC based framework that care groups of individuals occupied with a typical assignment (or objective) and that give an interface to a common situation" [2]. Prevalent groupware programming bundles included Lotus Notes and Microsoft Exchange.

The Web opened another window for the improvement of joint effort programming. With Web 2.0 came a plenty of cloud facilitated Internet-based applications that empowered wealthier coordinated effort, development of online groups, and different methods for collaboration. Today online joint effort apparatuses can be characterized in two classifications [3]:

(1) No concurrent joint effort devices. These instruments empower members to work together at various circumstances and diverse areas. These instruments are valuable for working together after some time and giving assets and data that are available whenever. For instance, by checking the correction history members can see who has contributed, when they have contributed, and what they have contributed. Additionally, the utilization of remarks enable members to concur, wrangle about, or clarify changes required in the work.

(2) Synchronous collaboration tools. These instruments empower members to work together progressively, regardless of whether in a similar area or in better places. The key purpose of synchronous apparatuses is that the innovation gives the communicators a chance to cooperate in the meantime.

The accentuation of this paper is on synchronous online colaboration since these sorts of instruments are currently made conceivable on the Web with the presentation of Web constant advancements, for example, WebRTC. Synchronous coordinated effort can have many favorable circumstances like quick reaction and criticism, video/web conferencing considering non-verbal communication and manner of speaking, expanded inspiration and engagement with course ideas and expanded social nearness. Burdens of synchronous joint effort incorporate the absence of reflection between colleagues, the prerequisite for expansive time duty of the

associates, the trouble to accomplish one to numerous correspondence and the way that if the innovation falls flat the coordinated effort session impractical [4].

Synchronous coordinated effort incorporates whiteboards, video and sound correspondence, content talk and screen sharing. Whiteboarding specifically is an educating and coordinated effort rehearse in which members utilize a whiteboard zone to draw or compose ideas, graphs, maps, tables, outlines, conditions and so on. Smith et al. in [5] directed a writing survey on intuitive whiteboard and found in addition to other things that they are especially compelling in training and virtual classrooms enabling educators to utilize instructing time to talk about understudy created thoughts as opposed to just displaying data and outlined the advantages of intelligent whiteboards as takes after: adaptability and various features, viability in sight and sound utilize; bolster for the lesson arrange; different assets; advancement of data and correspondence innovation abilities; and more communication and understudy interest in classes. Intuitive whiteboards draw in understudies with their associates in a shared learning group and it takes into consideration "more than one educator" in a classroom by enabling understudies with whiteboards to end up instructors also [6]. This upgrades inspiration, support and collaboration. Instructive whiteboards are turned out to be a compelling learning device for individuals of any age. For instance Akbaş et al. in [7] have assessed a whiteboard-based framework that prepared more established individuals to utilize programmed teller machines.

2. Related Work

Various synchronous online joint effort stages have been proposed or actualized economically. Jara C. in [8] proposed a web learning framework which joins synchronous communitarian learning in 3D virtual research facilities. In their work, they intergraded their structure in the well-known EJS material science program, enabling clients to team up utilizing the WebGL stage. Andrioti Z. in [9] joined WebRTC and the Evie-m stage [10] to make an online cooperative instructive virtual condition for educating arithmetic. WebRTC (Web Real Time Communication) is an API that permits ongoing distributed correspondence between programs without the utilization of extra plug-ins. It is contended that online cooperation utilized as a part of instruction prompts more positive learning results (learning through interest in a gathering) and more connected with learners [11].

Whiteboarding is a standout amongst the most famous utilizations of synchronous online joint effort. Intelligent whiteboards offer an impressive potential to upgrade understudy learning and are great instructive apparatuses when utilized fittingly [12]. For instance Metz et al. in [13] composed a synergistic whiteboard which was then assessed by allocating assignments to a gathering of clients and gathering information from client collaborations and talk correspondence. They demonstrated that whiteboard can be a viable coordinated effort apparatus. Strangely, they watched that the aggregate awareness of the gathering of clients is made through off-undertaking associations thus we can derive that this ability to have "off-assignment connection" is one reason that video correspondence and content visit altogether enhance joint effort productivity and is one of the benefits of synchronous coordinated effort. Today numerous online whiteboards are industrially accessible on the web.

Community oriented Web Browsing (cobrowsing) is another type of online coordinated effort in which at least two client explore the World Wide Web together by sharing a synchronized normal perspective of a website page and additionally sharing associations, for example, co-designing, content highlighting or mouse clicks with each other [31] [32]. Non intelligent cobrowsing can be exceptionally easy to actualize and can use the screen sharing API of WebRTC. In such situation a client is asking for support from an operator. The operator then continues to share his/her screen with the client trying to demonstrate the client what to do next while the client watches the screen sharing stream and follows up on his own program in like manner.

The genuine test is executing an intuitive cobrowsing session. The deterrents that should be overcome in module less intuitive cobrowsing incorporate managing treats, page personalization, login sessions, or solicitations for validation while managing the solid safety efforts and certainty prerequisites given by both the working framework and the web program (with most essential security impediment being "A similar inception strategy". In customer server based cobrowsing framework an answer has been proposed in [14] by empowering the client to control which web application information is proliferated and to authorize security strategies upon private information inside a cobrowse session.

3. WebRTC

The mission of WebRTC is "to empower rich, top notch RTC applications to be produced for the program, portable stages, and IoT gadgets, and enable them all to convey by means of a typical arrangement of conventions" [15]. WebRTC was publicly released by Google in 2011 and after that a continuous work begun to institutionalize the conventions related with it by IETF and its program APIs by W3C. Intrigue and support for WebRTC has been since developing relentlessly. Today, the most progressive WebRTC execution is offered by Mozilla Firefox and Google Chrome. These programs are currently supporting most of the elements of WebRTC that are proposed by the relating W3C drafts [16]. Different stages that bolster WebRTC to some augment incorporate the Opera program, the Android stage and Apple's iOS stage. Microsoft in its Edge program bolsters another set conventions named ORTC which does not utilize the SDP for session depictions but rather it is wanted to be interoperable with WebRTC [17]. It is normal that by 2018 WebRTC will be bolstered by 4.7 billion cell phones [18] and 1.5 billion PCs that run WebRTC empowered programs bringing the aggregate number to more

than 6.2 billion WebRTC empowered gadgets. WebRTC executes three browser embedded APIs: MediaStream, RTCPeerConnection, RTCDataChannel.

The MediaStream API is in charge of catching floods of media. These streams can be a video taken from the client's web camera, a stream from a canvas or video component or a screen catching stream. The RTCPeerConnection API is utilized to send these streams amongst programs and the RTCDataChannel API is utilized to trade subjective information, for example, application and amusement information additionally metadata between companions.

The nearness of an information station is a standout amongst the most imperative components of WebRTC permitting the advancement of all sort of P2P applications and synergistic arrangements extending from synchronized improvement [19] to telehealth administrations [20] and dialect learning [21] to "whiteboard coordinated effort" which is the subject of this paper. The design of WebRTC including the flagging server is appeared in the accompanying schema (Figure 1).

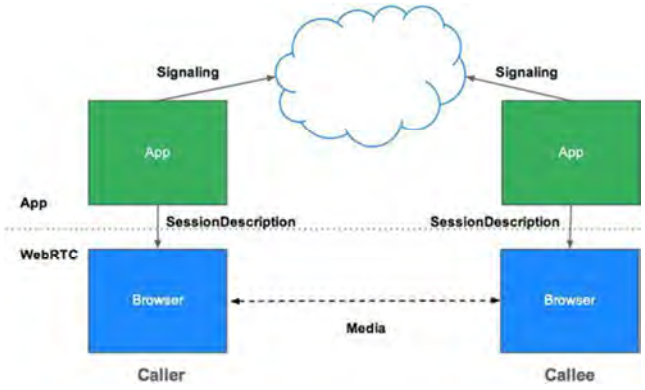

Figure 1. *WebRTC Architecture.*

In spite of the fact that WebRTC tries to empower Peer-to-Peer correspondence between programs without handing-off information through a server, an utilization of a server is as yet required for two reasons: The main reason is the conspicuous one, a web server is expected to "serve" the real JavaScript application that uses WebRTC. The second reason is more subtle. A server is required with a specific end goal to instate sessions between the customers that need to impart. This procedure is known as "Flagging" and is in charge of the trading of the underlying (meta) information of session portrayals (utilizing SDP) which contain points of interest on the shape and nature of the information which will be transmitted [22]. These data can incorporate system information, for example, IP addresses and ports, media metadata, for example, codecs and codec settings, transfer speed and media sorts, mistake messages or client and room data [23].

The codecs that are upheld by WebRTC are characterized by two IETF drafts. Sound codecs are depicted in "WebRTC Audio Codec and Processing Requirements" [24] and video codecs in "WebRTC Video Processing and Codec Requirements" [25]. As indicated by these drafts WebRTC programs must (total prerequisite) execute the VP8 video codec as portrayed in RFC6386 and furthermore H.264 Constrained Baseline as depicted in H264, and should likewise actualize the Opus sound codec depicted in RFC6716 and the G.711 PCMA and PCMU sound codec depicted in RFC3551. WebRTC likewise bolsters the iSAC and iLBC sound codecs.

4. Implementation

The application we developed is a prototype intended to demonstrate the capabilities of WebRTC and its potential use for online collaboration, whiteboarding and media streaming. Many of the APIs used in this paper are still in progress. As a result some of the features of WebRTC used in this paper are either not implemented in all major browsers or have some bugs. We decided to focus our development on the Mozilla Firefox browser which has almost all WebRTC proposed features implemented.

As explained, WebRTC requires a minimum load from a server. The server is used once to download the WebRTC application code (in our case, the whole application we developed is less than 160KB including images and code) and a second time to bring the peers together acting as a signaling server (the data exchanged is no more than a few kilobytes per connection).

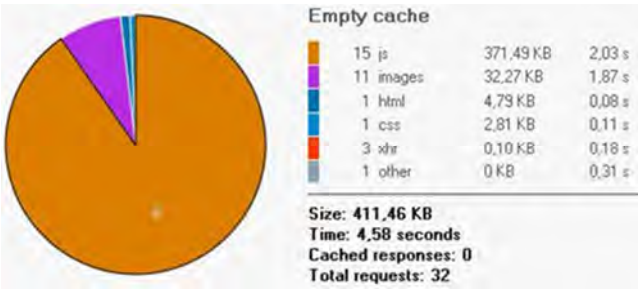

Figure 2. *Application data per type.*

In the above figure we see that the whole application downloaded from the server is under 412KB in size including images and external code (jQuery 1.12.3) amounting to a total of 32 HTTP requests. JavaScript amounts to about 90% of the bulk application data. If we take out the large size of the jQuery library (which is requested from its respective domain and not our application server) we see that the server load for each application pull is about 300KB.

For these reasons we experimented with running both the signaling server and the application host on a single-board computer. The board we selected was the BeagleBone Black which was designed by Texas Instruments. BeagleBone was launched in April 2013 and costs about $ 45 and uses up to 2W of power, making it a very economical and environmentally friendly solution. The system runs a precompiled distribution of Node.js v.0.10.41 for the BeagleBoard Black [26]. The Node.js server is then run using “forever”, a simple CLI tool for ensuring that a

given script runs continuously.

4.1. A Protocol for Synchronous Collaboration & Control

The initial move towards characterizing this "protocol" directed at synchronous online joint effort is to build up some type of deliberation layer sitting on top of the local WebRTC RTCDataChannel interface. There are two fundamental explanations behind this: First, we require uniform functions to consistently deal with messages traded amongst associates and second, on the grounds that the utilization of the inward WebRTC capacities to trade information through the information channel is regularly a muddled assignment requiring many lines of code and customizations. We propose wrapper functions exemplification the information channel usefulness into basic send and get capacities which rearrange the improvement and upkeep of the operations. At last we propose a "protocol" for consistently trading joint effort data and calling capacity on a remote associate.

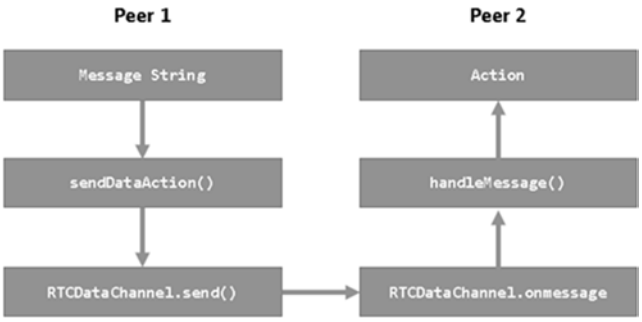

Figure 3. *Communication Model.*

The communication model described above is shown in figure 3. The foundation of the system is the native WebRTC data channel RTCDataChannel.send() function and onmessage property.

For sending data we have developed a function called sendDataAction() for sending strings and a function called sendDataFile() for sending binary data. The option to compress data using the Lempel-Ziv-Welch algorithm is also supported through an optional argument. In the following listing the definition of sendDataAction is given:

void sendDataAction(string message, bool compression)

Sends data through the WebRTC Data channel using the native WebRTC RTCDataChannel.send() method.

message The string to be sent

[compression] Optional. A Boolean representing whether the data should be compressed before sending

true: Compresses data using the LZW algorithm

false: No data compression (default)

Note that while in our implementation the send Data Action function broadcasts all messages to the other peer in the room, in a multiple-user environment this function could be adapted to send messages to specific users only (for example by adding an extra parameter defining a username)

The function handle Message evaluates incoming messages into function calls. Messages are comprised of array elements separated by a double colon (::). The first element of the array is always the name of the function to be called while the other elements correspond to the parameters of that function. For example when the system receives the string "::MSGNM::PAR1::PAR2::" the callback function handleMessage will look for a function named msgnm(p1, p2) and call it with "PAR1" and "PAR2" as its parameters as shown in the following schematic:

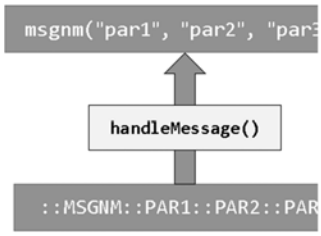

Figure 4. *Converting messages into function calls.*

Using the function handle Message we can execute functions of other JavaScript libraries, such as Cylon.js in combination with Socket.io to control IoT devices as shown in fig. 1. A remote peer can send messages to the local peer who is in direct control of the device. These messages are in turn translated into function calls of the Cylon Socket.io API. For example an incoming message to blink LED x every y seconds could be in the form of "LED::1::1000". The function led(led_no, ms) which would utilize the appropriate Cylon.js function call, would then be called by the system.

Finally the top layer of figure 2 consists of the exchanged standardized messages in the form of strings which represent the actions and function calls. This ensures that the system is well defined and can be easily expanded, but also interoperable so that any WebRTC applications that use this protocol can communicate with each other. The proposed protocol can be used for presenting metadata on video streams, which can include sketching information (Whiteboarding), or chat messaging but can be equally used for any data exchanged between peers including file data (binary), alerts etc.

As we explained, the way the system communicates actions between peers is done using a very simple language. Two colons (::, Unicode U+003A) are used to indicate that what follows is system data in the form of either strings or "stringified" JSON objects. Chat messages or any other data must be filtered and barred from containing this set of characters. Messages also contain an array of information, the elements of which are separated by a double colon (::). The first element of the array is always a 5 letter string defining the message type e.g.:

a) URMSG: A chat message

b) FILES: An incoming binary file

c) SKTCH: Sketching data etc….

We theorize here that an online collaboration platform is comprised of these two elements: video streams and users. Therefore, messages come in two distinct forms: Messages that are intended for canvases and messages that are intended for users. For example a drawing corresponds to a canvas while a chat message corresponds to a user (since a user can

have more than one canvas or video shared). We assume that in a peer-to -peer environment messages are broadcast to all peers (all users share the same streams). Of course it is possible to include the name of the recipient in the message in order to send data targeted at specific users.

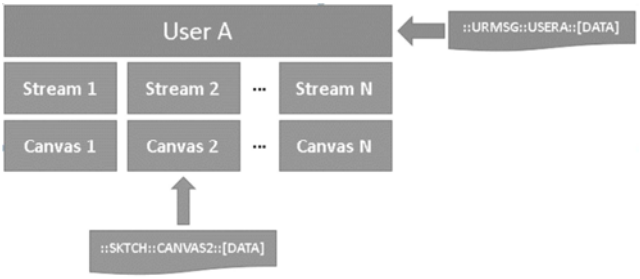

Figure 5. *Message types.*

We also have predefined and implemented a number of messages that are useful in a synchronous online collaboration environment. Some of them are shown in the following table:

Table 1. *Sample messages.*

Prefix	Data
::URMSG:: UNAME::DATA	A chat message from a user with username "UNAME"
::SKTCH:: TARGET::WIDTH::DATA	JSON Sketch data including text annotation for the stream named "TARGET"
::FILES:: DATA	Data for incoming files
::PAUSE:: TARGET::TIME	Pauses a stream at a specified time

4.2. The Interface

The interface of the developed application consists of the following areas:

1. The connection box
2. The streams list
3. The maximized stream area
4. The toolbox
5. The chat area

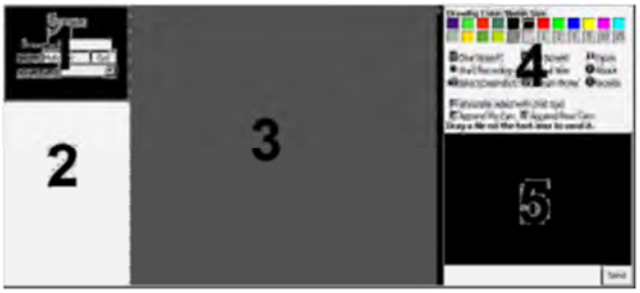

Figure 6. *User Interface Organization.*

The connection box is where the users can enter a username and a room name. Users can also chose to start a screen sharing session from using the dropdown box in the area. Upon initialization of a session the MediaStream API prompts the user for permission to use one video and/or one audio input device such as a camera, a microphone or for permission to start capturing a screen or part of it. If the user provides permission, then the returned

Clicking on the maximize button brings the selected stream on the center area of the page and enables the collaborative controls for this specific stream.

Once connection is established users have a range of tools available from the sidebar on the right side of the screen.

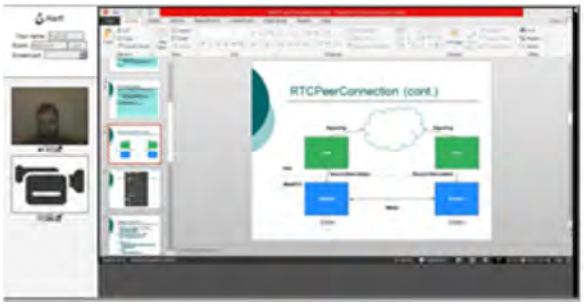

Figure 7. *A screen capturing session with the PowerPoint window maximized.*

Table 2. *Toolbar functions.*

	Starts or stops recording the currently maximized stream. The recording is available as a webm file only to the user that initiated the recording.
	Clears all sketches on the currently maximized stream
	Pauses or freezes the currently maximized stream
	Prompts the user to select a local video file which can then be streamed to the other peers
	Captures a single frame from the currently maximized stream which can then be saved as a PNG file.
	Currently maximized stream in full screen. All annotations options are disabled in this mode and users cannot annotate the stream.
	Currently maximized stream to its original size.
	Resizes the Replaces the currently maximized stream with a HTML document hosted on the same domain as the WebRTC application. Basic interactive cobrowsing is offered to the peers in this mode.

The application communicates messages to the user using the chat area below the toolbox. For example when a user clicks on the "Take screenshot" button the system sends a message to the user notifying him of the link from which he can download it. This increases system usability by eliminating the use of popups or other types of alerts. These "system messages" are only visible to the user they concern and not to the other peers of the session.

Figure 8. *Example system messages.*

The chat area can also be used for exchanging files between users. A user can drag and drop a file on the text area to send it to other users.

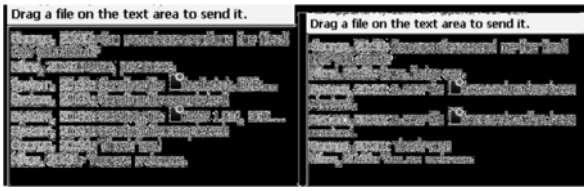

Figure 9. *Users exchanging files.*

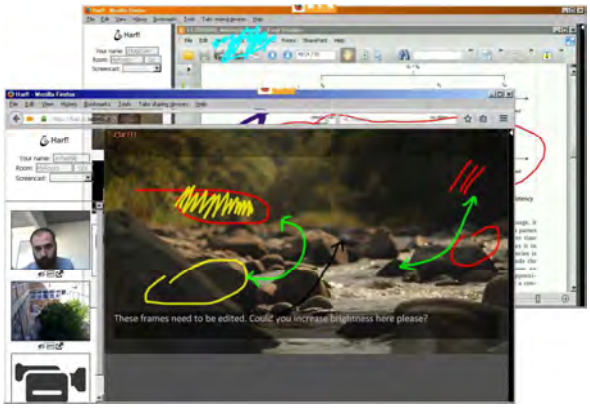

Figure 10. *Users sketching on a PDF document and on a video.*

4.3. An IoT Approach

It becomes obvious that this protocol can be extended to IoT devices by using existing IoT JavaScript libraries (figure 1). For example the SocketIO API plugin of the Cylon.js library can be used to remotely interact with an Arduino microcontroller in real-time. The user of such a library can use our proposed protocol to call functions of this library through the WebRTC data channel as seen in figure 8. In this example a Cylon.js Socket.io connection is established between one of the peers' computers. The developer has defined the message:

::USELED::USER:ACTION::PAR1::PAR2

The message states that user USER requests to perform an action on a specific LED on the Arduino board. The message would be translated in the following function call:

function USELED(user, action, par1, par2)

Which would then in turn call the Socket.io emit function:

device.emit(action, par1, par2);

Figure 11. *A remote user observing an Arduino board through the webcam can issue commands.*

In the above example a Toggle button has been added so that when clicked it sends the "USELED::USERNAME:TOGGLE" string through the WebRTC data channel. Upon receiving that string, the peer whose computer has an established connection with the Arduino device will toggle the state of LED13 on the board. Applications of this technique could include the collaborative control and observation of more advanced devices such as motors, servomechanisms, analog sensors etc.

5. Benchmarking

5.1. Compression Efficiency

We utilized the LZW compression calculation to pack information sent through the WebRTC information channel. To quantify pressure productivity we associated two PCs and measured the time it took to render an outline contingent upon the span of the JSON question that depicted the draw.

The LZW compression calculation is exceptionally productive for portraying metadata in view of the high number of catchphrases and word cycles When the information is compacted utilizing the LZW calculation the subsequent information is 6.6KB with a pressure proportion of 93%. Clearly utilizing pressure on the information channel can drastically diminish arrange overhead and with current equipment the compresion/decompression times on the nearby host framework are really miniscule as appeared in the accompanying table.

Table 3. *Compression Efficiency.*

Bytes Before Decompression	Bytes After Decompression	Compression Ratio (%)	Relative Time Difference (ms) - LAN
235	1001	77	0
486	3044	84	15
2001	21645	91	37
2668	31585	92	152
3255	40661	92	83
3991	51650	92	100
5728	76730	93	140
7535	104437	93	194

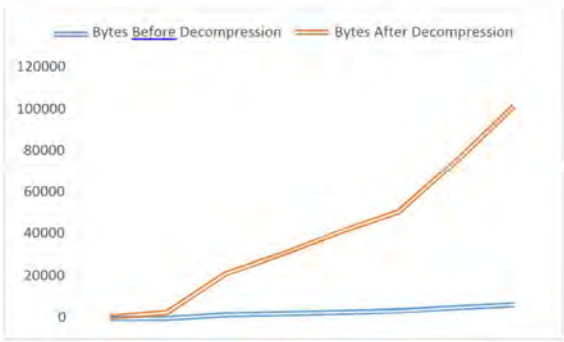

Figure 12. *Compression Efficiency.*

In the above table we measure the time required to render a graphic on the canvas assuming that the initial rendering (empty canvas) requires 0 milliseconds. Rendering time is measured from the time a user makes a sketch to the time it is rendered on the other peer's computer.

We found that compressing the data channel on LANs and

in situations where network bandwidth is not a problem may still not be an optimal choice because when combining the time required to compress a string with the time required to decompress it, the overhead is significant especially on older systems. Also the media streams (which are compressed using the WebRTC codec of choice) take up the majority of the data transferred.

As a result we concluded that although the compression ration achieved is very high (up to 93%) and that using compression on the data channel can be useful on some situations (e.g. when no media streams are utilized), compressing data on the data channel does not provide a significant advantage on most situations.

5.2. CPU Usage

On the signaling server, a typical session description message is about 2KB in size, while a candidate offer message is about 150 bytes. Assuming that each peer exchanges one session description message and 5 candidates on average, we see that for each peer connection, less that 5 kilobytes of data (10 kilobytes for 2 peers) are send and received from the signaling server.

On the client, modern hardware is powerful enough for all the video and canvas operations that are required by most applications including our own. Furthermore HTML5 Hardware Accelerated canvas is implemented on most platforms and browsers taking advantage of the capabilities of modern GPUs.

To measure the processing power requirements of our system we used the embedded developer tools in Mozilla Firefox 46. The test system was a laptop equipped with 4GB of RAM and an Intel Core i3 (U38) CPU with a clock speed of 1.33 GHz. The computer was running the Microsoft Windows 7 64bit operating system. The system can be considered outdated by today's standards.

To analyze which processes consume more time we used the Firefox Performance Tool and conducted two 20 second tests: During the first test the system was used for streaming media between two peers while during the second test the sketching feature was also used. During the first test the average framerate was measured at 42fps while during the second test average framerate was at 17 fps.

Table 2. *Function CPU usage while sketching and while streaming.*

	CPU Usage	
	Sketching &	
Function	**Streaming**	**Streaming**
Gecko (includes idle time)*	37.06%	60.83%
Sketching	19.06%	-
Graphics*	16.40%	23.57%
Garbage Collecting*	9.41%	6.52%
JIT*	4.83%	1.88%
Tools*	2.25%	3.28%
Input & Events*	1.79%	-
Compression/Decompression Algorithms	1.71%	-
Other	7.49%	3.92%

* Denotes internal browser functions

In the following figure we see the framerate in which the browser renders the page during the tests.

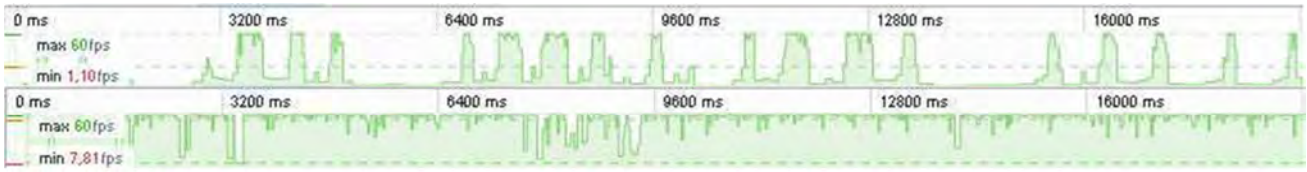

Figure 13. *Comparison of framerate during simple streaming (above) and during sketching (below). Negative spikes denote that the user is sketching on the canvas.*

6. Conclusions

WebRTC is a moderately new technology that enables program to-program correspondence. In this paper we introduce a conceivable utilization of WebRTC innovation in the fields of synchronous online joint effort and IoT control. We propose a uniform method for exchanging information through the WebRTC information channel. In this manner, can be utilized to exchange metadata for online cooperation stages and in the meantime in blend with existing IoT JavaScript libraries, for example, Cylon.js or Janus to impart or control IoT gadgets.

We have additionally built up an application as a model expected to exhibit the abilities of WebRTC and the proposed convention, and its potential use for online joint effort, whiteboarding and media gushing. The application exploits present day HTML5 APIs, for example, the Screen Capture, Media Recording and Stream Capture from Media components to offer clients the capacity to share video streams from an assortment of sources and afterward utilize the WebRTC information channel to trade cooperation metadata that incorporate portrayals, video explanations and IoT gadget activities.

References

[1] J. F. N. Jr, R. O. Briggs and N. C. Romano, Collaboration Systems: Concept, Value, and Use, New York: Routledge, 2014.

[2] C. A. Ellis, S. J. Gibbs and G. Rein, "Groupware: some issues and experiences," *Communications of the ACM,* vol. 34, no. 1, pp. 39-59, 1991.

[3] T. Walhert, "Synchronous or Asynchronous Tools," Green Hills Area Education Agency, [Online]. Available: https://sites.google.com/a/ghaea.org/aiw-iowacore-techintegration/ synchronous-vs-asynchronous. [Accessed 16 4 2016].

[4] B. Kask and S. Wood, "Synchronous and Asynchronous Communication: Tools for Collaboration," University of British Columbia, [Online]. Available: http:// etec.ctlt.ubc.ca/510wiki/ Synchronous_and_Asynchronous_Communication:Tools_for_ Collaboration. [Accessed 16 4 2016].

[5] H. J. Smith, S. Higgins, K. Wall and J. Miller, "Interactive whiteboards: boon or bandwagon? A critical review of the literature," *Journal of Computer Assisted Learning,* vol. 21, no. 2, pp. 91-101, 2005.

[6] C. J. Wenning, "Whiteboarding & Socratic dialogues: Questions & answers," *Journal of Physics Teacher Education Online,* vol. 3, no. 10, pp. 3-10, 2005.

[7] O. Akbaş, M. Baturay and a. Y. Söker, "Interactive Whiteboard-Based ATM Use Training for Older Individuals," *International Online Journal of Educational Sciences,* vol. 8, no. 1, pp. 87-97, 2016.

[8] C. A. Jara, F. A. Candelas, F. Torres, C. Salzmann, D. Gillet, F. Esquembre and S. Dormido, "Synchronous collaboration between auto-generated WebGL applications and 3D virtual laboratories created with Easy Java Simulations," in *9th IFAC Symposium Advances in Control Education*, Nizhny Novgorod, 2013.

[9] Andrioti, H., Stamoulias, A., Kapetanakis, K., Panagiotakis, S., & Malamos, A. G. (2015, June). Integrating WebRTC and X3DOM: bridging the gap between communications and graphics. In Proceedings of the 20th International Conference on 3D Web Technology (pp. 9-15). ACM.

[10] Athanasios G. Malamos, Georgios Mamakis, Paraskevi Sympa, Eleni Kotanitsi,Yannis Kaliakatsos, Dionysios Kladis, Alfredo Javier Gonel Crespo, Alvaro Zubizarreta Lopez, "Extending X3D-based educational platform for mathematics with multicast networking capabilities", in Proceedings of WBE2009, Phuket, Thailand, 16-18 March 2009.

[11] M. Hammond, "Online collaboration and cooperation: The recurring importance of evidence, rationale and viability," *Education and Information Technologies,* pp. 1-20, 2016.

[12] R. Zevenbergen and S. Lerman, "Learning Environments Using Interactive Whiteboards: New Learning Spaces or Reproduction of Old Technologies?" *Mathematics Education Research Journal,* vol. 20, no. 1, pp. 108-126, 2008.

[13] S. M.-V. Metz, P. Marin and E. Vayre, "The shared online whiteboard: An assistance tool to synchronous collaborative design," *European Review of Applied Psychology,* vol. 65, no. 5, pp. 253-269, 2014.

[14] J. Franke and B. Cheng, "Real-time privacy-preserving cobrowsing with element masking," in *2013 17th International Conference on Intelligence in Next Generation Networks (ICIN)*, Venice, 2013.

[15] "WebRTC," [Online]. Available: http://www.webrtc.org/home. [Accessed 4 7 2015].

[16] "Is WebRTC ready yet?" [Online]. Available: http://iswebrtcreadyyet.com/. [Accessed 2 3 2016].

[17] J. Wagner, "What Developers Should Know About ORTC Versus WebRTC," ProgrammableWeb, 12 10 2015. [Online]. Available: http:// www.programmableweb.com/news/what-developers-should-know-about-ortc-versus-webrtc/analysis/2015/10/12.

[18] ABI Research, "4.7 Billion Mobile WebRTC Devices by 2018 Despite Lack of Open Support from Apple and Microsoft," 25 9 2013. [Online]. Available: https:// www.abiresearch.com/press/47-billion-mobile-webrtc-devices-by-2018-despite-l/.

[19] K. Jain, A. Himmatramka, A. Bhandary, A. D'silva and D. Barge, "Synchronized Development Using WebRTC Real-Time Collaboration in WebRTC," *International Journal of Engineering Science,* vol. 6, no. 4, 2016.

[20] L. V. Ma, J. Kim, S. Park, J. Kim and J. Jang, "An efficient Session_Weight load balancing and scheduling methodology for high-quality telehealth care service based on WebRTC," *The Journal of Supercomputing,* pp. 1-18, 2016.

[21] I. V. Osipov, A. A. Volinsky and A. Y. Prasikova, "E-Learning Collaborative System for Practicing Foreign Languages with Native Speakers," *International Journal of Advanced Computer Science and Applications,* vol. 7, no. 3, 2016.

[22] C. Alexandru, "Impact of WebRTC (P2P in the Browser)," *Internet Economic VIII,* pp. 39-58, 2014.

[23] S. Dutton, "WebRTC in the real world: STUN, TURN and signaling," [Online]. Available: http://www.html5rocks.com/en/tutorials/webrtc/infrastructure/. [Accessed 20 2 2016].

[24] J. Valin and C. Bran, "WebRTC Audio Codec and Processing Requirements," 9 2 2016. [Online]. Available: https://tools.ietf.org/html/draft-ietf-rtcweb-audio-10.

[25] A. Roach, "WebRTC Video Processing and Codec Requirements," 12 6 2015. [Online]. Available: https://tools.ietf.org/html/draft-ietf-rtcweb-video-06.

[26] "Node.js for the BeagleBone Black," ARMhf, 27 4 2013. [Online]. Available: http:// www.armhf.com/node-js-for-the-beaglebone-black/.

[27] Mozilla Developer Network, "MediaDevices.getUserMedia()," [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia. [Accessed 1 3 2016].

[28] W3C, "Media Capture from DOM Elements," 10 3 2016. [Online]. Available: http:// w3c.github.io/mediacapture-fromelement/.

[29] S. Dutton, "VP9 is now available in WebRTC," Google Developers, [Online]. Available: https://developers.google.com/web/updates/2016/01/vp9-webrtc?hl=en. [Accessed 18 4 2016].

[30] C. Hart, "Does telephony matter if no one talks to each other anymore?" 18 1 2016. [Online]. Available: https://medium.com/@chadwallacehart/does-telephony-matter-if-no-one-talks-to-each-other-anymore-5edf61f27e71#.gdmm244q7.

[31] Kostas Kapetanakis, Spyros Panagiotakis, Athanasios G. Malamos, "HTML5 and WebSockets; challenges in network 3D collaboration", in Proceedings of the 17th Panhellenic Conference on Informatics (PCI 2013), 19-21 September, 2013, Thessaloniki, Greece.

[32] Steiakaki, M., Kontakis, K., Malamos, A. G. "Real-Time collaborative environment for interior design based on Semantics, Web3D and WebRTC". In Proceedings of the 15th International Symposium on Ambient Intelligence and Embedded Systems (Ami Es), 2016.

Biography

Nikos Pinikas received his B. Sc and M. Sc degrees from the Technological Educational Institute of Crete. He has worked in education and as a freelance web developer. His research interests include web programming and development, web UX design and ICT in education.

Despina Athanasaki received her B. Eng in Informatics Engineering from the Technological Educational Institute of Crete, where she was also working as a lab assistant for the courses: "Satellite Communications", "Industrial Automations & Information Systems" and "Microelectronics". Currently she is a post-graduate student at the University of Melbourne, in the faculty of MIT (Master of Information Technology), with major in Spatial Technologies. Her research interests include robotics, embedded systems, signal processing and communications.

Spyros Panagiotakis was born in Iraklio, Crete, in 1973. He received a BSc in Physics from the University of Athens (1997), an MSc in Electronic Automation in 2001 and a PhD in Communication Networks from the Department of Informatics and Telecommunications of the University of Athens in Greece in 2007. He is currently an Assistant Professor at the Department of Informatics Engineering of the Technological Educational Institution of Crete in Greece. He is author of over than 40 publications. His research interests focus on mobile multimedia technologies, communications and networking, web engineering, mobile applications, pervasive computing, and sensor networks.

Athanasios G. Malamos received Bsc in Physics from the Univeristy of Crete and Msc and Phd from the Technical University of Crete. Since 2002 is with Technological Educational Institute of Crete, Dept. of Informatics Engineering as assistant professor (2002-2006) and associate professor (2006 until present). Dr. Malamos is head of the Media Networks and Communications Lab. He is an active member of the WEB3D community. Dr. Malamos has served as Chair, program and organizational committee in several international conferences and workshops. He is regular reviewer of international journals. His research interests include multimedia services and web.