Albert Abelló Lozano

Performance analysis of WebRTC

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 20.3.2012

Thesis supervisor:

Prof. Jörg Ott

Thesis advisor:

M.Sc. (Tech.) Varun Singh

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Albert Abelló Lozano

Title: Performance analysis of WebRTC

Date: 20.3.2012 Language: English Number of pages:4+12

Department of Communication and Networking

Professorship: Networking Technology Code: S-55

Supervisor: Prof. Jörg Ott

Advisor: M.Sc. (Tech.) Varun Singh

Your abstract in English. Try to keep the abstract short, approximately 100 words should be enough. Abstract explains your research topic, the methods you have used, and the results you obtained.

Keywords: Resistor, Resistance,
Temperature

Preface

Thank you everybody.

Otaniemi, 9.3.2012

Albert Abelló Lozano

Contents

List of Tables

List of Figures

Definitions and abreviations

1 Introduction

The need of a new way to communicate between two points of the planet is a problem that many different technologies have tried to approach. Systems such Skype or VoIP are not able to cope the needs of the new generations of developers and users that everyday require a more integrated way of communication with the World Wide Web (WWW).

Besides this, the amount of data transferred during the last years and the prevision for the future allocates a new scenario where non-centralized systems such as P2P are required as data bandwidth grows and systems need to become more scalable. Nowadays, networks are still manly content-centric, meaning that data is provided from a source to a client in a triangle scheme, clients upload data to central servers and this data is transferred to the endpoint. This architecture has been provided since long time as reliable and scalable, but with the appearance of powerful applications and Video On Demand (VOD) scalability is becoming an issue.

Those circumstances lead to a whole new world of real-time browser based applications which require also a new framework to work with. Ranging from online videoconferencing to real-time data applications, for this purpose few attempts were made in the past being highly reliable on specific hardware and custom-built no-compatible systems. Those proposals were not accessible to normal users that could not afford to adapt the requirements.

All previous concepts are now possible thanks to the increase of the average performance in every computer nowadays, this situation is helping to build more complex browsers that are able to perform many different tasks that enhance web browsing to a different level. Having a browser to handle OpenGL style of applications is now possible thank to the new HyperText Markup Language version 5 (HTML5) standard. Multimedia abilities are also able to be reproduced on those browsers and webcam media shown as HTML is now a reality. Even dough, there is still an important issue that must be addressed: there is no common standardized protocol that allows developers to do this. Web Real-Time Communication (WebRTC) effort to approach this problem is to build a simple and standard solution for peer-to-peer browser communication [1] in the HTML5 environment.

Internet bandwidth capabilities helped to take the decision to start integrating peer-to-peer solutions in browsed based applications, this is due the year-by-year increase of user bandwidth connectivity during the last 10 years. Actual latency in the network is low enough to allow real-time applications to work resiliently in the browser. The amount of users being able to transfer at high speed has increased during last years (Figure 1), about 39% of users are able to download at speeds greater than 4Mbps being this a very good average speed for multimedia content [2].

Regarding the specs on the client side, recent surveys and statistics taken by the game manufacturer Steam [3] prove that more than 61% of machines are carrying 1 to 4 gigabytes of RAM and nearly 90% of computers handle 2 to 4 core CPU with a 64 bit OS, this environment can easily handle media enhanced applications that require high performance for media encoding. WebRTC concept rests over multiple layers

	Country	% Above 4 Mbps	QoQ Change	YoY Change
–	Global	39%	-2.8%	6.9%
1	South Korea	84%	-2.2%	28%
2	Switzerland	79%	2.3%	14%
3	Netherlands	79%	-5.0%	-2.0%
4	Japan	74%	2.4%	20%
5	Belgium	69%	-4.7%	-6.3%
6	Canada	69%	0.8%	8.2%
7	Hong Kong	68%	-5.2%	-4.7%
8	Latvia	67%	-1.7%	-2.8%
9	Denmark	66%	5.2%	5.0%
10	Czech Republic	64%	-6.4%	-10%
...				
13	United States	57%	-4.6%	7.9%

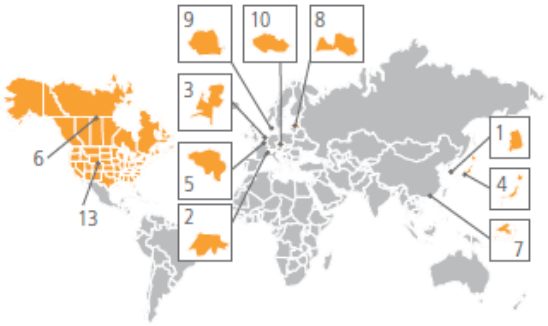

Figure 1: Broadband connectivity statistics about the speeds over 4Mbps around the globe.

having the browser as an underlying application, a traditional browser allocates a lot of resources for running being the performance of the machine a bottleneck in some cases.

Traditionally, WebRTC concept approaches rely on the usage of plug-ins or other separate software components that make the system run smoother by avoiding one layer of processing (browser) but being non-standard and not cross-compatible, one of the most import ant concepts when designing applications nowadays. This approach has a new alternative with the arrival of the new HTML5 where WebRTC is integrated as one of the new Application Programming Interfaces (APIs) available alongside other many different interesting capabilities.

1.1 Background

WebRTC API is included into the HTML5, this is the fifth version of the WWW language. This version includes different API's and JavaScript codes that help the developer to easily introduce new features into their already existing WWW applications. The initial HTML version (2.0) was published in November 1995 [4] with the only goal of delivering static content from the server to client browser. HTML became de de facto format for serving web information.

HTML is written in tag formatting to identify different elements. Those tags are then interpreted by the browser to show the different data content served by the server. During the evolution of the WWW different new features have been added to the HTML standard and new versions where published, things like JavaScript and Style Sheets increase the flexibility and features of the WWW content enhancing the final user experience.

Due to the need to extend the features of the already existing HTML4 standard, a new version was proposed in 2004 by the Mozilla Foundation and Opera Software [5]. This new proposition focused in new developing technologies that could be backwards compatible with the already existing browsers, the idea didn't make a success and was tier apart until January 2008 when the first Public Working

Draft was published by the Web HyperText Application Technology Working Group (WHATWG) in the W3C [6].

This proposal had a greater reliance in modularity in order to move forward faster, this meant that some specs that were included in the initial draft moved to different working groups in the W3C. Those technologies defined in HTML5 are now in separate specifications, one of them being WebRTC. WebRTC works as an integrated API within the browser that is accessible using JavaScript and is used in conjunction with the Document Object Model (DOM) interfaces. Some of the APIs that have been developed are not part of the HTML5 W3C specification but are included into the WHATWG HTML specification.

1.2 Contribution

Investigate how WebRTC performs in a real environment trying to evaluate the best way to set multiple peer connections able to transfer media and data in different network topologies. Measure the performance of WebRTC in a real environment identifying bottlenecks related to encoding/decoding, media establishment or connection maintenance. All this should be performed in real-time over a browser by using the already existing WebRTC API.

Using metrics related to RTT, latency, packet loss and bandwidth usage we expect to understand the way WebRTC performs when handling multiple connections.

1.3 Goals

WebRTC uses and adapts some existing technologies for real-time communication. This thesis will focus in studying how:

- WebRTC performs considering different topologies using video/audio acquired form the Webcam using the API and encoded using different codec types provided by the standard.
- Usage of WebRTC to build a real application that can be used by final users proving that the API is ready to be deployed and is a good approach for the developer needs when building real-time applications over the web. This will be done in conjunction with other new APIs and technologies introduced with HTML5.

The final conclusion will cover an overall opinion and usage experience of WebRTC, providing some valuable feedback for the needs and requirements for further modifications on the API.

1.4 Structure

Not sure about here

2 What is WebRTC?

Web Real-Time Communication is an standard that will help to build P2P applications in the developer layer relying in a defined API. The first announcement went public in a working group of the World Wide Web Consortium (W3C) in May 2011 [7] and started the official mailing list in April 2011 [8]. During the first stage of discussion, the main goal was to define a public draft for the API implementation and a route timeline with the goal to standardize the protocol by ends of 2012. The first public draft of W3C came public the 27th of October 2011 [9]. During this first W3C draft, only media (audio and video) could be sent over the network to other peers, they focused in the way browsers will be able to access the media devices without using any plugin or external software.

Alongside to the W3C working group, the WebRTC concept also joined the IETF with a Working Group (WG) in May 2011 [10] with the first public announcement charter done the 3th of May 2011 [11]. The milestones of the WG initially marked December of 2011 as deadline to provide the information and elements required to the W3C for the API design input. On the other side, the main goals of the WG covered the definition of the communication model, session management, security, NAT traversal solution, media formats, codec agreement and transport of the data [11]. All goals have evolved during the standardization process with the work done along with the W3C WG.

One of the most important steps during the process of standardization came the 1st of June 2011 when Google publicly released the source code of their API implementation [12].

During all this period both WGs have been working alongside to provide a reliable solution to enable applications to perform media and data peer-to-peer transfer in a plugin-free environment. The first final version of the WebRTC API is to be published during March 2013.

2.1 Support

The following companies have supported and are actively working in the development of WebRTC standard in the W3C: Google, Mozilla and Opera [13]. Other companies such as Microsoft have supported browser-to-browser solution but have published their own proposal which differs with the one published in the WebRTC WG, called CU-RTC-Web [14], this proposal did not get much traction by the workgroup being declined to unify with the current specs, during an W3C workgroup poll in September 2012 the chairs of the group decided to attach to the already existing WebRTC API instead of moving it to the CU-RTC-Web [15].

During the firsts attempts to build a reliable solution for WebRTC Ericsson Labs presented an initial API based on the preliminary work done in the WHATWG, this API was called ConnectionPeer API and required an special module to be installed in your browser [16]. Ericsson lately dropped from the effort to build it's own browser to focus in the standardization and codec discussion, leaving the API implementation to the Mozilla and Chrome teams. The original API evolved rapidly

during the next months thanks to the WGs and the developer community feedback that is experimenting with the unstable API.

2.2 Milestones

During the process of standardization some important moments should be remarked. In January 2012 Opera implemented the first version of WebRTC getUserMedia for accessing the camera and audio [17], during this year getUserMedia is available in the stable version of Opera.

Google Chrome integrated the first version of WebRTC in its DEV and Canary channels of the browser during January 2012 [18], in June 2012 it started moving its API to the stable channel hidden behind a flag, in November 2012 WebRTC becomes fully available in Google Chrome stable channel and is open for public usage [19].

Mozilla Firefox started working on the getUserMedia implementation early 2012 delivering the first version of media access trough API at the beginning of 2012 in the alpha version [20], in April 2012 Mozilla published a WebRTC video demo running on Firefox in the "adler" channel [21], also supporting some primitive DataChannel API. Later in October Firefox Nightly was carrying the first unstable version of the WebRTC API including DataChannel [22], Mozilla announced in September 2012 that the stable version of WebRTC will be shipped along with Firefox 18 in January 2013 [23].

Some announcements done from Microsoft point out that they are also working in some implementation into Internet Explorer by using CU-RTC-Web as the default standard, at the moment only the Media API information is publicly available [24].

Some mobile platform important moments should be pointed out. In October 2012 Ericsson announced the world's first WebRTC-enabled browser for mobile devices called "Bowser" with support for iOS and Android, this browser is able to handle WebRTC calls using RTCWeb Offer/Answer Protocol (ROAP) which is an old discontinued version of the WebRTC API that has moved to Javascript Session Establishment Protocol (JSEP). This browser also differs from the previous desktop alternatives on the codec side, it is carrying H.264 for video and G.711 for audio [25]. The API provided by Bowser is not fully W3C compliant.

2.3 WebRTC architecture

How WebRTC works, simple topology and key points.

2.4 Alternatives

Some alternatives are available to the WebRTC concept, considering the global architecture of WebRTC, Session Initiation Protocol (SIP) and Secure Real-Time Media Flow Protocol (RTMFP) are similar approaches to the same solution.

2.4.1 SIP

Both SIP and RTMFP are protocols to allow communication between two different users with audio/video support. SIP is an open standard and RTMFP is a proprietary protocol by Adobe, both systems are widely used for real-time communication. SIP final Request for Comments (RFC) was published in June 2004 [26], this document describes the methods and behaviors of SIP. From an overview perspective, SIP is an application-layer control protocol for multimedia sessions, can establish, maintain and terminate them, during the development of the standard different new functionalities were added to the drafts such as conferencing and the possibility of adding/removing media from existing sessions. SIP differentiates from RTMFP/WebRTC by locating the end user to be used for communication, this feature allows SIP to be closely related to traditional PSTN networks as it allow cross-domain communication which is not possible when using RTMFP/WebRTC. SIP is not a complete toolkit for communications, it works alongside with other existing protocols such as Real-time Transport Protocol (RTP), Real-Time Streaming Protocol (RSTP), Session Description Protocol (SDP) and Media Gateway Control Protocol (MEGACO). Using SDP for the session negotiation between the end-points and RTP/RSTP for the media transport, all those protocols usage is widely extended in the network and provides legacy for older technologies. Meanwhile SIP can locate and deliver a message to a user, SDP can provide the required information for the session establishment and RTP can transport the media specified in the SDP body.

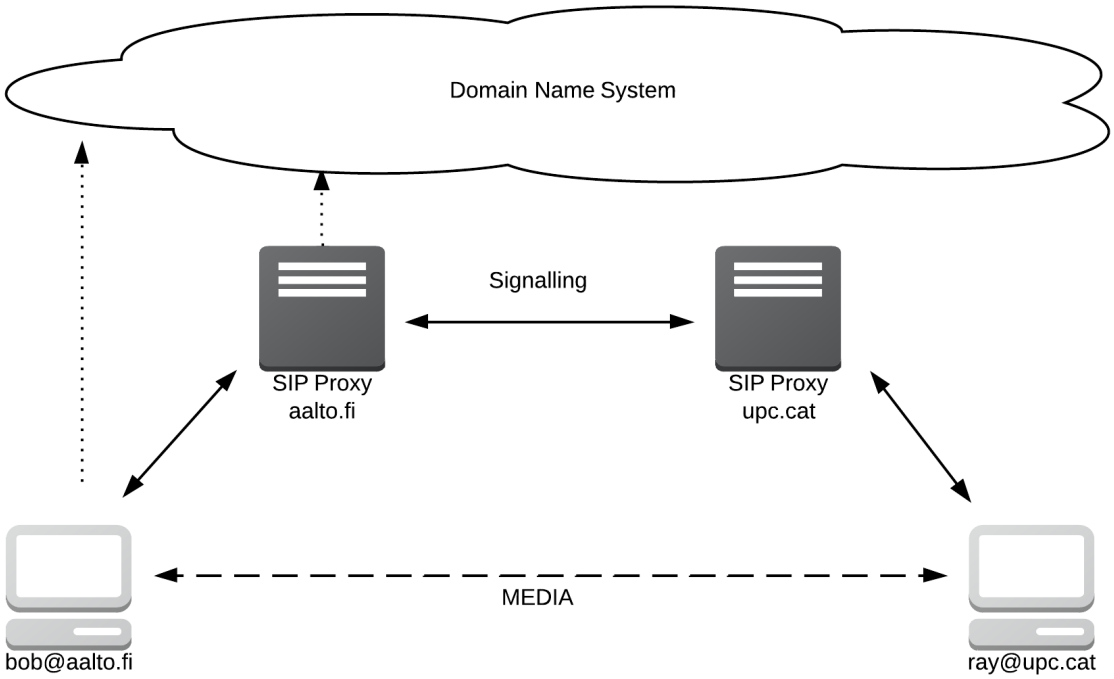

Figure 2: SIP architecture for end-to-end signaling.

SIP architecture relies in a trapezoid form where the Domain Name System (DNS) is used to locate the other peers of the system, once that peer is located and

session is negotiated, media flows peer-to-peer directly to the endpoint. In order to build this system different agents are needed, SIP Proxies, SIP Redirect and SIP Registrar. SIP Proxies transmit the SDP and SIP messages from one peer to the other to establish communication (Figure 3). SIP Registrar are the machines that collect and save all the user information from the end points.

DNS provides the IP address for both proxy servers and allow the messages to be exchanged between both peers, when SIP is used the following messages are exchanged: INVITE, RINGING and 200OK. Those messages carry the SDP data inside in an object format, when ray@upc.cat receives the INVITE message from bob@aalto.fi builds the 200OK response carrying the SDP object that providing compatibility check between both peers and which options and codecs to use. SIP provides some more messages to update the already existing session or to close them. The media transport is done using RTP and RTCP that rely over User Datagram Protocol (UDP) [26].

2.4.2 RTMFP

RTMFP is a proprietary transport protocol provided by Adobe, this protocol works in a peer-to-peer scheme to provide media and data transfer between two end points. This system usually works over UDP [27]. Differing from SIP, this protocol is a full suite for media/data transfer in a P2P environment and carries its own signaling method and codecs. It also handles congestion control on the packets and NAT transversal issues. One of the biggest differences is that, similar to WebRTC, is not a full communication infrastructure and both peers must be in the same working domain to be able to communicate, is not a PSTN style of communication but a point to point messaging system. This protocol is implemented in Flash Player, Adobe Integrated Runtime (AIR) and Adobe Media Server (AMS) [27], it is used for P2P communication between all those services.

This protocol is secured and encrypted, comparing with WebRTC, this issue has been addresses clearly in RTMFP by using proprietary algorithm and different methods. The RTMFP architecture is similar to WebRTC concept, it also allows reconnection in case of connectivity issues and works by multiplexing different media streams over the same media channel when handling conferences or multiple streams. For the signaling part Adobe uses a service called Cirrus (Figure), this service allows architectures such as: end-to-end, many-to-many and multicast [28].

Some of the most valuable features is the possibility to easy integrate P2P multicast topologies where one source sends a video to a group of receivers. This is something that none of the other protocols has implemented yet.

2.4.3 WebRTC

Talk about SIP, RTMFP and WebRTC.

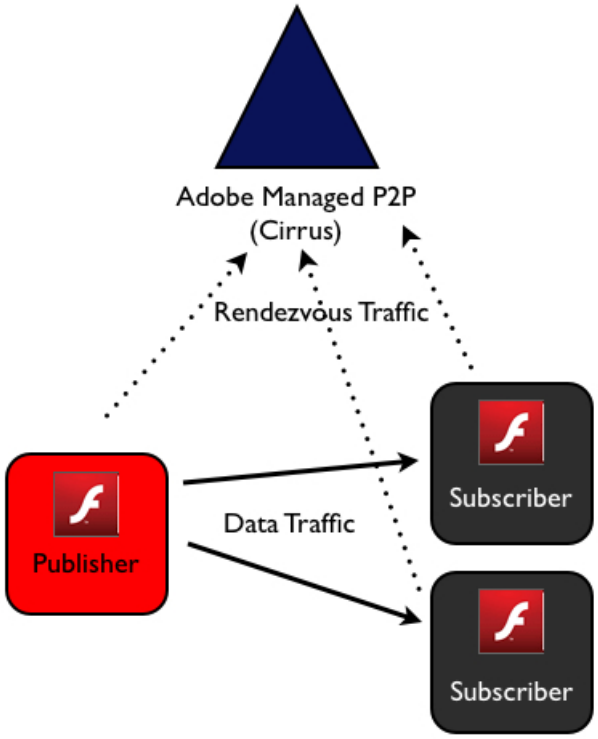

Figure 3: RTMFP architecture using Cirrus (source: Adobe).

2.5 Issues in WebRTC

WebRTC uses a mixture of different technologies to perform peer-to-peer communication between clients, those technologies range from SRTP, RTP, RTCP and multiple codecs that are being discussed. This scenario makes performance the key point for success in developing stable WebRTC applications.

Performance is manly related to computer capabilities and the ability to encode/decode at the same time as transferring and monitoring multiple peer connections. All those tasks are run over the browser and not directly on the OS, this is good for interoperability between platforms but bad in the performance aspect.

Media applications are delay sensitive and require a low packet loss for its proper function, WebRTC is working on this aspect by trying to implement congestion control over the connection stablished between peers, this work has not been completed yet and will arise as a problem in the near future. Packet loss due to system capacity and bandwidth are measurable in WebRTC using the Stats API, this API provides information about the PeerConnection performance and is accessible by JavaScript.

Constraints and bandwidth statistics will make a big difference in how media is acquired in WebRTC. Browsers and web applications have always tolerate some amount of delay and packet losses but this is not possible in media infrastructures for real time conversations, a change of scope is needed to handle Quality of Service (QoS) in WebRTC.

3 Conclusion

The end.

References

[1] H. Alvestrand. Overview: Real Time Protocols for Brower-based applications. Technical report, IETF, 2012. Also available in http://tools.ietf.org/html/draft-ietf-rtcweb-overview-04.

[2] Akamai. The State of the Internet, 2ND Quarter, 2012 Report, 2012.

[3] Steam. Steam Hardware and Software Survey, October 2012. Also available in http://store.steampowered.com/hwsurvey.

[4] T. Berners-Lee. HyperText Markup Language 2.0. Technical report, IETF, November 1995. Also available in http://tools.ietf.org/html/rfc1866.

[5] Mozilla Foundation and Opera Software. Position paper for the W3C workshop on web applications and compound documents. Technical report, W3C, 2004. Also available in http://www.w3.org/2004/04/webapps-cdf-ws/papers/opera.html.

[6] Ian Hickson and David Hyatt. HTML5. Technical report, W3C, January 2008. Also available in http://www.w3.org/TR/2008/WD-html5-20080122/.

[7] Web Real-Time Communications Working Group, May 2011. Also available in http://www.w3.org/2011/04/webrtc/.

[8] Harald Alvestrand. Welcome to the list!, April 2011. Also available in http://lists.w3.org/Archives/Public/public-webrtc/2011Apr/0001.html.

[9] Cullen Jennings Adam Bergkvist, Daniel C. Burnett and Anant Narayanan. WebRTC 1.0: Real-time Communication Between Browsers. Technical report, W3C, October 2011. Also available in http://www.w3.org/TR/2011/WD-webrtc-20111027/.

[10] Real-Time Communication in WEB-browsers, May 2011. Also available in http://tools.ietf.org/wg/rtcweb/.

[11] Cullen Jennings Magnus Westerlund and Ted Hardie. Real-Time Communication in WEB-browsers charter. Technical report, IETF, May 2011. Also available in http://tools.ietf.org/wg/rtcweb/charters?item=charter-rtcweb-2011-05-03.txt.

[12] Google release of WebRTC source code, June 2011. Also available in http://lists.w3.org/Archives/Public/public-webrtc/2011May/0022.html.

[13] Rian Liebenberg and Jan Linden. Introducing WebRTC an open real-time communications project, April 2011. Also available in http://lists.w3.org/Archives/Public/public-webrtc/2011Apr/0001.html.

[14] Bernard Adoba and Martin Thomson. Customizable, Ubiquitous Real Time Communication over the Web (cu-rtc-web). Technical report, Microsoft, August 2012. Also available in http://html5labs.interoperabilitybridges.com/cu-rtc-web/cu-rtc-web.htm.

[15] Poll for preferred API alternative, September 2012. Also available in http://lists.w3.org/Archives/Public/public-webrtc/2012Sep/0010.html.

[16] Stefan Hakansson. Beyond HTML5: Peer-to-Peer Conversational Video, January 2011. Also available in https://labs.ericsson.com/developer-community/blog/beyond-html5-peer-peer-conversational-video.

[17] Bruce Lawson. getUserMedia: accessing the camera and privacy UI, January 2012. Also available in http://dev.opera.com/articles/view/getusermedia-access-camera-privacy-ui/.

[18] Niklas Enbom. Real-time Communications in Chrome, January 2012. Also available in http://blog.chromium.org/2012/01/real-time-communications-in-chrome.html.

[19] Serge Lachapelle. See you on the web!, November 2012. Also available in https://sites.google.com/site/webrtc/blog/seeyouontheweb.

[20] Robert O'Callahan. MediaStreams Processing Demos, January 2012. Also available in http://robert.ocallahan.org/2012/01/mediastreams-processing-demos.html.

[21] Anant Narayanan. WebRTC efforts underway at Mozilla!, April 2012. Also available in https://hacks.mozilla.org/2012/04/webrtc-efforts-underway-at-mozilla/.

[22] Randell Jesup Anant Narayanan, Maire Reavy and Rob Hawkes. Progress update on WebRTC for Firefox on desktop, November 2012. Also available in https://hacks.mozilla.org/2012/11/progress-update-on-webrtc-for-firefox-on-desktop/.

[23] Robert Nyman. Full WebRTC support is soon coming to a web browser near you!, September 2012. Also available in https://hacks.mozilla.org/2012/09/full-webrtc-support-is-soon-coming-to-a-web-browser-near-you/.

[24] Media Capture API, March 2012. Also available in http://html5labs.interoperabilitybridges.com/prototypes/media-capture-api-(2nd-updated)/media-capture-api-(2nd-update)/info.

[25] Stefan Alund. Bowser - The World First WebRTC-Enabled Mobile Browser, October 2012. Also available in https://labs.ericsson.com/blog/bowser-the-world-s-first-webrtc-enabled-mobile-browser.

[26] G. Camarillo A. Johnston J. Peterson R. Sparks M. Handley J. Rosenberg, H. Schulzrinne and E. Schooler. SIP: Ssession Iinitiation Protocol, June 2004. Also available in http://www.ietf.org/rfc/rfc3261.txt.

[27] M.Thronburgh. Adobe%'s Secure Rreal-Time Media Flow Protocol, January 2013. Also available in http://tools.ietf.org/html/draft-thornburgh-adobe-rtmfp-02.

[28] Adobe. Cirrus FAQ. Also available in http://labs.adobe.com/wiki/index.php/Cirrus:FAQ.