common technique to simulate real world network impairments. Emulators are powerful tools that enable researchers to perform repeatable experiments. However, care should be taken when utilizing emulators for several reasons. For instance, their features and functionalities differ and may not behave exactly as configured. [2] presents a comparative research about network emulators and [3] shows that emulators might sometimes not exactly produce the desired network conditions.

D. Speech Quality Evaluation

Subjective speech quality evaluation is typically performed as a listening activity rather than a conversational activity. Objective evaluation methods tend to follow this exercise by comparing the original speech with recorded one to obtain the speech quality. The architecture of the experiments in our study is designed accordingly.

Intrusive or non-intrusive methods can be used to objectively evaluate perceptual quality of speech. While intrusive methods use original and recorded/distorted signals, non-intrusive methods use only recorded/distorted signals. PESQ is an intrusive model. An example of a non-intrusive model is the E-model.

PESQ is a widely used, standardized, convenient for automation, objective voice quality test methodology. The principle behind PESQ is to predict the subjective quality of degraded speech signal by comparing it with the original reference signal [4]. In principal, PESQ results model mean opinion scores (MOS). It has a mapping function, which translates raw PESQ scores (in the range -0.5 to 4.5) to the MOS-LQO (Mean Opinion Score - Listening Quality Objective) scale in a range from 1 (bad) to 5(excellent).

III. WEBRTC OVERVIEW

The WebRTC concept of peer-to-peer communication directly between browsers is a new development in web technology. Web technology has been, so far, operating only according to the basic web browser and web server model where web browser talks only to the web server over the HTTP/HTTPS or WebSockets protocols. Figure 1 depicts the conventional web browser model.

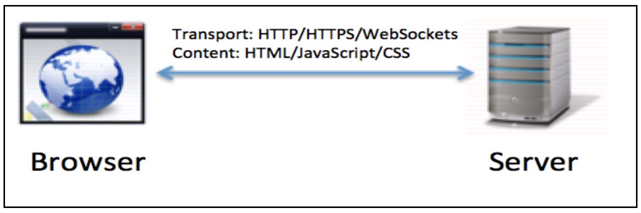
Transport: HTTP/HTTPS/WebSockets
Content: HTML/JavaScript/CSS
Browser
Server

Figure 1. Web browser model

In the usual RTC scenario the peers go through a signaling phase before they start transmitting voice, video, or data. This phase is not standardized in WebRTC and is left to the developer's choice. In the signaling phase, the caller sends an invitation the called party, which might cause, for instance, the called endpoint to ring or give a notification to the user depending on the design. Another signaling information that is passed between the peers is the Session Description Protocol (SDP) packets. SDP allows the peers to exchange a list of information that enables them to start a secure RTP session between themselves. Examples are media capabilities and preferences of the browsers [5]. Once the signaling is handled over HTTP/HTTPS/WebSockets, browsers can directly exchange realtime data in a peer-to-peer manner as shown in Figure 2.

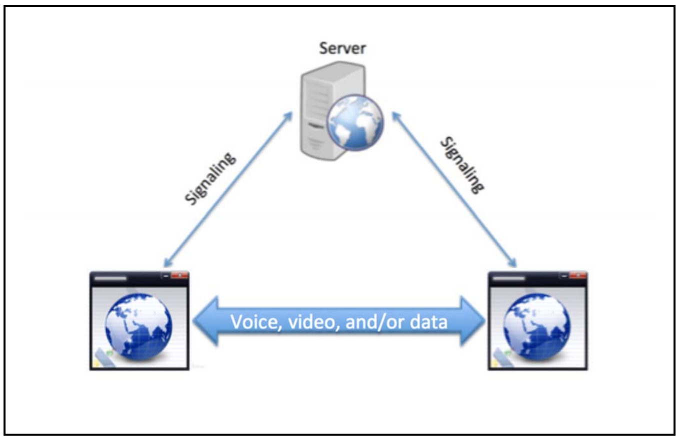

Figure 2. Web browser model with WebRTC

WebRTC is a collective standardization effort between W3C and IETF [6]. While IETF defines the underlying formats and protocols that browsers use between each other [7], W3C focuses on the definition of the APIs that a Web application can consume to manage this communication [8].

WebRTC functionality is available in some of the mainstream web browsers, including Google Chrome, Mozilla Firefox, and Opera. There is an open project initiative, which is also called WebRTC[1] – same as the name of the standardization activity in W3C, to enable web browsers with realtime communication capabilities. Microsoft Internet Explorer and Safari do not yet support WebRTC.

IV. TEST-BED AND MEASUREMENT METHODOLOGY

A. High-level Test-bed Architecture

The high-level design for the experiments is shown in Figure 3. As per the PESQ guidelines [4], the realtime data flow was a one-way stream between peers, from speaker browser to listener browser with no RTP data flowing in the reverse direction. In order to avoid unnecessary traffic, WebRTC's video streaming capability was not enabled. As speech input, an ITU sample [4] file was used. On the speaker side, the ITU sample speech file, which is referred as original file, was read and fed into the WebRTC session. Similarly, on the listener side, the speech output from the WebRTC session was recorded to a file, which is referred as the recording. The network conditions were manipulated to alter base delay and jitter between the peers.

Chromium was used as the WebRTC browser. Chromium[2] is an open source project and has already some code for browser testing. We used and extended them to implement our WebRTC experiments. In that regard, open source nature of Chromium eased the development of the test-bed. Chromium

[1]http://www.webrtc.org/
[2]http://www.chromium.org/

978-1-4799-3303-7/14/$31.00 ©2014 IEEE

uses the libraries from the WebRTC project and supports the Opus as preferred audio codec.

The experiments were executed on LinuxMint distribution, which is based on Ubuntu and kernel version of 3.8.0-19-generic.

B. *Test-bed*

As shown in Figure 3, there was a speaker peer and a listener peer in the system and simulated call was a one-way conversation. Speaker peer can be considered as sender or initiator side of the call.

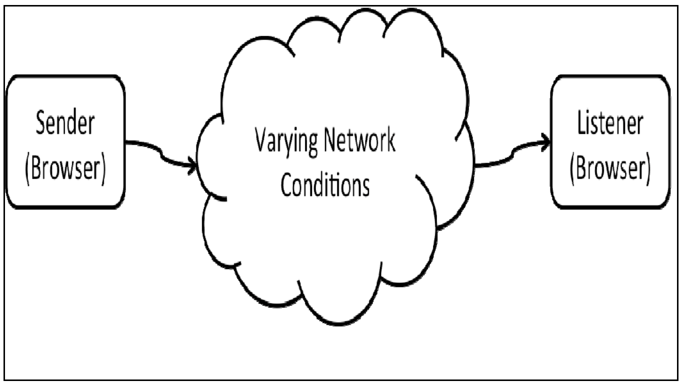

Figure 3. High-level test-bed architecture

Sender peer read an speech file, an ITU-T speech sample, using Web Audio APIs [9] and sent the voice streams to the network through WebRTC components. Web Audio APIs are standard Java Script APIs and they are commonly used in web development and they are not directly part of WebRTC APIs. On the receiver/listener side, WebRTC components received the voice stream from the network and played it out. The output on the listener side was recorded into an speech file to evaluate its quality. Next, the original speech file and recorded file were passed to the PESQ algorithm to predict the quality score.

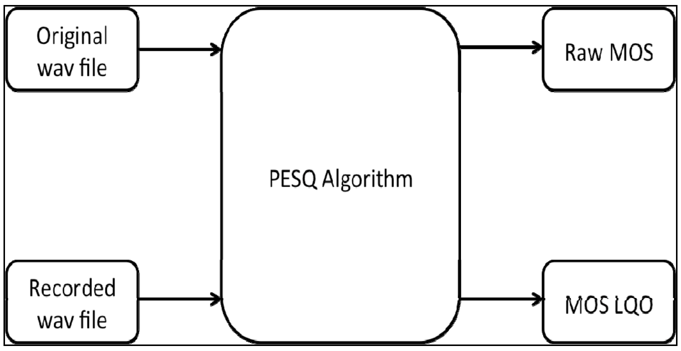

Figure 4 PESQ Block Diagram

PESQ source code is publicly available from ITU-T website and can be compiled using a C/C++ compiler. PESQ outputs MOS and MOS-LQO results as explained in Section II. Combining Figure 3 and Figure 4, the entire test-bed is depicted in Figure 5.

The web application directly read an speech file and fed it into WebRTC components using Web Audio API instead of capturing from the microphone. On the listener side, received stream was recorded into a file. At the start of study, we had assumed that either WebRTC or Web Audio would have the necessary API to accomplish this, however, the research showed that neither of them provided such an API as of yet. It was also found that this functionality, as defined by W3C [10], is expected to be delivered by the browser vendors in the future. In the absence of a recording API, a recording application, *arecord*, was used to record the voice coming from the browser. *arecord* comes from the ALSA project and is easy to use with scripting for test automation purposes [11].

Initially, the test bed was considered to have two browsers on two different machines connected through network, one machine as sender and another as listener. The required network conditions and parameters, such as delay and jitter, were varied using a network emulator, eg. dummynet. Figure 6 shows this two-machine test architecture.

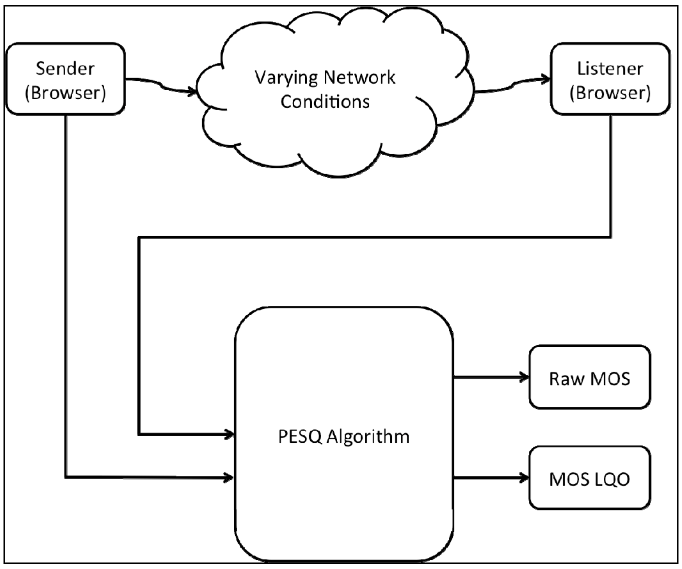

Figure 5. Test architecture block diagram

However, by connecting two machines over a network through a switch, router, or an access point, there is also some amount of uncertainties and variations that come into the network due to the unpredictable aspects of networking, especially with wireless access points and other software that maybe running on the system. Additionally, the automation of experiments on multiple machines is more difficult and expensive.

We thus decided to have both sender and receiver peers on the same machine, even on the same browser. Using WebRTC APIs, a sender peer and a receiver peer were created with the same browser session as illustrated in Figure 7. Another motivation behind this architecture was the convenience of an end-to-end test on single machine.

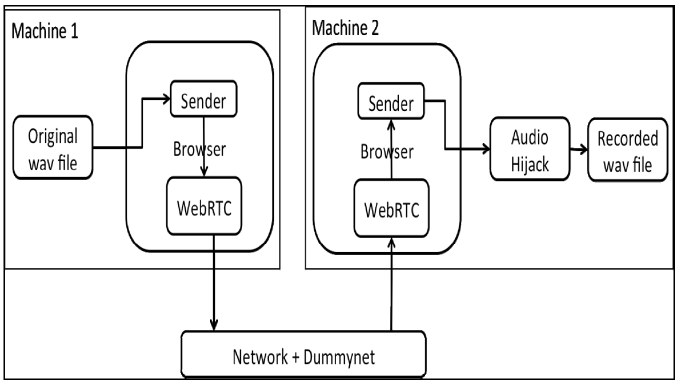

Figure 6. Two-machine test architecture

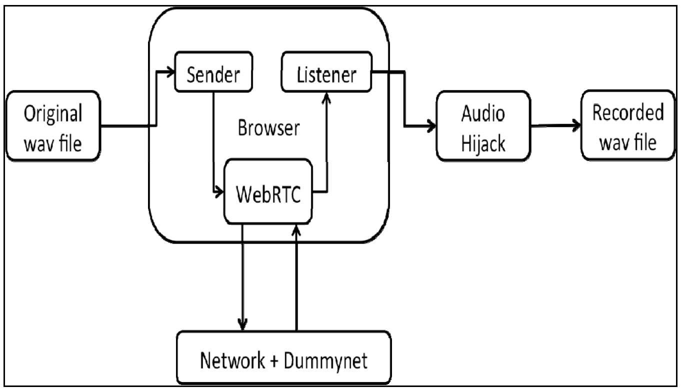

Figure 7. Single machine test architecture

The idea here was to have the test bed behave in a loopback manner, which is that the sender peer would send the speech packets that it gets from the file to the network interface and the packets would go to the listener peer without reaching the access point. It was thought that this would give a better test bed, which is near to an ideal network, and it could produce repeatable results since networking conditions would not change unless the network simulator, dummynet, manipulates it. However, the experiments showed that WebRTC implementation in Google Chrome does not support loopback model, i.e. the packets have to pass through the physical network interface and come back. Chromium does not have this limitation.

V. RESULTS

Our study concentrated on performing a black-box type voice quality evaluation in a one-way call within a WebRTC browser under varying network delay and network jitter profiles.

Initial results indicated that changing base delay and jitter affects the PESQ results. The higher delay/jitter values are, the more distortion/loss occurs, which results in lower PESQ values. Table 1 shows the initial results for full recording.

TABLE I. PESQ RESULTS FOR FULL RECORDING OF 9.32 SECONDS

Delay Profile Min/Max/Jitter (ms)	*Raw MOS*	*MOS-LQO*
50/60/10	4.0121	4.1651
100/120/20	3.9525	4.0946
150/190/40	3.993	4.1457
200/280/80	3.1642	3.0526
300/400/100	2.263	1.8862

a. Results are average of 20 executions per each delay profile

However, listening to the recorded speech files showed that, there was only distortion/loss at the beginning, first ~3 seconds, of the entire conversation and there was no noticeable difference for the rest. After extracting only the last 6 seconds of the recorded files and applying PESQ showed that the scores are consistently around 4.

TABLE II. PESQ RESULTS EXTRACTED RECORDING OF LAST 6 SECONDS

Delay Profile Min/Max/Jitter (ms)	*Raw MOS*	*MOS-LQO*
50/60/10	3.9434	4.0959
100/120/20	3.9408	4.0931
150/190/40	3.9277	4.0795
200/280/80	3.9758	4.1288
300/400/100	3.9269	4.0761

a. Results are average of 20 executions per each delay profile

We then checked the waveforms of the reference file and recorded files and the result was consistent with the PESQ scores. Figure 8 shows the missing waveform in the beginning of the recording for a test run with a delay profile where minimum delay was 300ms, maximum delay was 400ms, and jitter was 100ms. The PESQ results for the full recording was MOS: 2.555 and MOS LQO: 2.203.

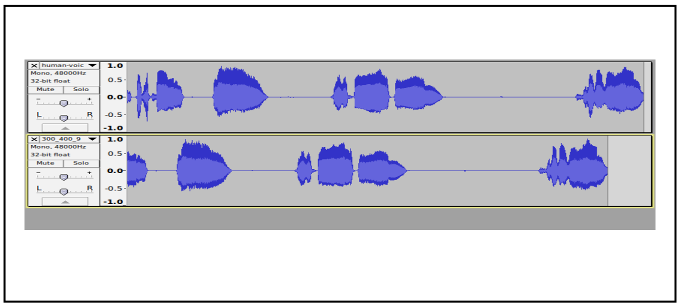

Figure 8. Waveform for full original and full recording file

Figure 9 takes a closer look at Figure 8 and focuses on the last 6 seconds, where the waveforms show almost a perfect match between the reference and recoding file. PESQ results for the last 6 seconds in Table II also support this. In this particular case, the PESQ results were MOS: 3.93 and MOS LQO: 4.082.

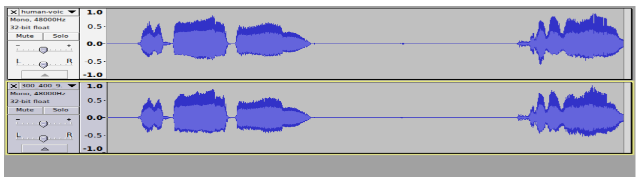

Figure 9. Waveform of the last 6 seconds of original and recording file

One of the important conclusions from the study is that black box testing results might be misleading. The impact of a single machine also needs to be investigated. There are several problems with realizing a network connection between two peers on a single machine. For instance, the network communication occurs over the loopback interface between two peers on the same machine and it is not possible to differentiate sender and receiver side arrival times of a packet when it is passing through the network interface. We have one arrival time since it is passing the loopback interface only once. Test-bed architecture with peer browsers deployed on separate machines would align better with the real world scenarios. The variations in the system caused by multiple machines can be minimized and/or factored into the results.

We have found from initial analysis that emulators, both dummynet and netem, are not exactly meeting the network criteria we specify. While netem has built-in jitter functionality, it does reordering and produces out of order packets, which is not the desired case in our experiments, and generally uncommon. On the other hand, dummynet doesn't have built-in jitter functionality; hence the jitter behavior is performed by constantly changing the base delay with no out of order packets. Yet, this didn't even create the desired jitter behavior as we found that time delta between each packet arrival is almost always 30ms for all different test profiles

VI. Conclusion and Future work

In this study, we conducted a series of experiments on a test-bed to emulate real world conditions and measure the voice quality with PESQ under network impairments. Initial low PESQ results for high delay profiles are due to the missing packets at the beginning of the recordings. The reason for the missing packets is to be further researched. We conclude that two-machine test architecture for WebRTC endpoints, plus an intermediary machine for the network emulator, is certainly a better topology for this kind of research from many perspectives, such as traceability and transparency. In future studies, network emulators should be validated separately to ensure they generate desired network conditions and align with real world network scenarios.

References

[1] Singh, V., Lozano, A. A., & Ott, J. (2013, December). Performance Analysis of Receive-Side Real-Time Congestion Control for WebRTC. In *Proc. of IEEE Packet Video* (Vol. 2013).

[2] Nussbaum, L., & Richard, O. (2009, March). A comparative study of network link emulators. In *Proceedings of the 2009 Spring Simulation Multiconference*(p. 85). Society for Computer Simulation International.

[3] Jurgelionis, A., Laulajainen, J., Hirvonen, M., & Wang, A. I. (2011, July). An empirical study of netem network emulation functionalities. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on* (pp. 1-6). IEEE.

[4] ITU-T P. 862. (2001). Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs.

[5] Handley, M., Perkins, C., & Jacobson, V. (2006). *SDP: session description protocol. Internet Engineering Taskforce RFC 4566.*

[6] Johnston , A. B., & Burnett , D. C. (2012). *WebRTC APIs and RTCWEB Protocols of the HTML5 Real-Time Web.* St. Louis, MO: Digital Codex LLC.

[7] Hardie, T., Jennings, C., & Turner, S. Real-Time Communication in WEB-browsers. IETF, https://tools.ietf.org/wg/rtcweb/

[8] Bergkvist, A, Burnett, D. C., Jennings, C., & Narayanan, A. WebRTC 1.0: Real-time Communication Between Browsers. W3C, http://www.w3.org/TR/webrtc/

[9] Rogers, C. Web audio API. W3C, http://www.w3.org/TR/webaudio/

[10] Barnett, J & Leithead, T. W3C, MediaStream Recording. http://www.w3.org/TR/mediastream-recording/

[11] ALSA. Soundcard Testing. http://www.alsa-project.org/main/index.php/SoundcardTesting