Video QoE Killer and Performance Statistics in WebRTC-based Video Communication

Doreid Ammar*, Katrien De Moor*, Min Xie†, Markus Fiedler‡, Poul Heegaard*
*Department of Telematics,
NTNU, Norwegian University of Science and Technology, Trondheim, Norway
{doreid.ammar|katrien.demoor|poul.heegaard}@item.ntnu.no
†Telenor Research, Next Generation Services, Trondheim, Norway
{Min.Xie}@telenor.com
‡Blekinge Institute of Technology, Karlskrona, Sweden
{markus.fiedler}@bth.se

***Abstract*—In this paper, we investigate session-related performance statistics of a Web-based Real-Time Communication (WebRTC) application called appear.in. We explore the characteristics of these statistics and explore how they may relate to users' Quality of Experience (QoE). More concretely, we have run a series of tests involving two parties and according to different test scenarios, and collected real-time session statistics by means of Google Chrome's WebRTC-internals tool. Despite the fact that the Chrome statistics have a number of limitations, our observations indicate that they are useful for QoE research when these limitations are known and carefully handled when performing post-processing analysis. The results from our initial tests show that a combination of performance indicators measured at the sender's and receiver's end may help to identify severe video freezes (being an important QoE killer) in the context of WebRTC-based video communication. In this paper the performance indicators used are significant drops in data rate, non-zero packet loss ratios, non-zero PLI values, and non-zero bucket delay.**

I. Introduction

Applications and services enabling synchronous, audio- and video-mediated communication have become more and more popular over the last years, as reflected in the wide range of video-chat and -conferencing solutions, each with their own features and affordances, that are available today (e.g., Skype, Google Hangouts, appear.in, Cisco WebEx, AnyMeeting, etc.). Such solutions are commonly used, not only in a professional setting, but also in a more private context, e.g., for socialising with friends and family that are not co-located. As the term 'videoconferencing' is mainly associated with business contexts, [1], [2] proposed to use the term 'telemeeting' to encompass both business and private contexts in which audiovisual mediated communication takes place.

Even thought the performance of such telemeeting systems and applications is continuously improving and even though some very basic real-time video quality optimisation strategies are already being employed [3], [4], it remains very challenging to offer users an excellent Quality of Experience (QoE) with real-time audio- and video-mediated communication, all the time and in all circumstances. First of all, multi-party audiovisual conversations are often characterised by a certain technical asymmetry: the technical circumstances under which the call takes place may differ from party to party (e.g., device, network connection, etc.). This has important implications from a Quality of Experience (QoE)-point of view: 3 out of 4 parties may from a technical point of view participate under ideal conditions for positive and pleasurable QoE, yet still, their actual QoE might be very bad because one of the parties is connected through a weak mobile network, resulting in a very distorted conversation. Factors such as the context and nature of the call (e.g., business vs. leisure), the conversation dynamics and interactivity patterns, features and functionalities of the used videoconferencing system, etc. may also play an important role. A number of recent studies (see Section II) already pointed in this direction. Yet, the actual influence of these technical and non-technical factors and their implications for telemeeting QoE are still poorly understood.

Additionally, and despite the high processing power of current devices, there is always some inherent and unavoidable delay due to the technology-mediated character of telemeetings. The messages of sender and receiver (audio and video input) need to be recorded, prepared and encoded for transmission, transmitted over a network, decoded by the receiving device and finally presented to the other party [5]. During each of these steps, delay and additional technical artefacts that may impact the produced and experienced audio and video quality, as well as the synchronisation between both modalities [6] may be introduced. However, whether and to which extent different types of quality degradations are problematic during a conversation (e.g., in which circumstances, why, how) and how they interplay with other impact factors still requires further investigation. The root causes of impairments and indicators of bad QoE may also differ depending on the service's underlying technology and protocols: e.g., Web Real-Time Communication (WebRTC) in the browser that is realised by RTP (real-time transport protocol) over UDP (user datagram protocol) vs. proprietary IP telephony architectures as used by e.g., Skype (requiring additional software support).

In this paper, we investigate session-related performance statistics linked to the use of an off the shelf, WebRTC-based application called appear.in. This application enables video communication for up to 8 parties and can be accessed using browsers that support WebRTC (e.g., Chrome, Firefox). When using Google Chrome, data from both the sending and receiving parties in a WebRTC-based telemeeting can be gathered via the WebRTC internals page (chrome://webrtc-internals/), thus making it possible to get a more complete overview of the conversation. Such session-related statistics may help to identify root causes and track the origins of performance issues in multi-party conversations, and as a second step, to better understand how these technical factors may impact user behaviour and users' QoE as a conversation is unfolding. Gathering such insights is crucial and may steer the development of real-time, intelligent optimisation strategies in the future. However, the characteristics of the gathered Chrome stats and their relevance for QoE issues are currently poorly understood. Our aims with this paper are therefore to (1) explore the properties and potential value of session statistics gathered by Google Chrome for research on QoS and QoE of WebRTC-based telemeetings and (2) explore which factors may say something about the QoE during a session.

The paper is organised as follows: Section II briefly points to related work on telemeeting QoE. In Section III the set-up of the conducted measurements is described. It is followed by Section IV, which shares our main findings on the characteristics of the Chrome Stats and a number of additional results and observations from the conducted measurements. Finally, Section V concludes the paper.

II. Related work

Current research on QoE of telemeetings and video-mediated multi-party conversations is focusing on influence factors situated at three different levels, namely the human level (e.g., role and involvement of a person in the conversation, previous experiences with telemeetings, personality traits), the system level (e.g., network conditions, application-level aspects, type of device) and context (e.g., purpose and setting of the call, acoustic and visual environment) [7], [2], [8]. Due to this multitude of potential influence factors, the evaluation of quality and users' QoE in the context of multi-party telemeeting situations is particularly challenging. New methods and approaches (beyond those described in [9]) are therefore under development in ITU-T Study Group 12 (Question 18). Several methodological issues and questions need to be addressed in this respect, e.g., how to assess and take into account potential asymmetric call conditions? Which type of 'task' is most suitable (natural, free conversation vs. scripted approaches)? Which quality aggregation level / perspective should be used? How to include and evaluate the impact of interactivity and contextual factors?

Despite the lack of agreed quality assessment methods for multi-party video conversations, several studies have already been conducted in this area. In [1], Berndtsson et al. report on a series of both one-way and conversational tests to assess subjective quality for different types of telemeeting scenarios. A two-party, conversational audiovisual test indicated that the synchronisation between audio and video is very important, and that it is even better to delay the audio a bit to ensure better synchronisation in case the delay is below 600 ms. Another set of multiparty telemeeting tests indicated that 800 ms end-to-end delay is considered as unacceptable and showed to affect the experienced interaction quality in a negative way. Some interesting nuances were made however: the tolerance towards delay was higher in the audio test. Tolerance levels also differed according to the type of task (free conversation vs. a quiz-alike task) and according to the social context (alone in a room vs. together with others) [1]. In [10] it was found that also the level of involvement in the conversation (listening task vs. being part of the conversation) influences the experienced telemeeting quality. For their specific set-up, they found that this influence can moreover be even larger than the impact of the considered technical factor (i.e., different packet loss rates).

In [7], a flexible and customisable testbed for investigating QoE of video-mediated group communication in controlled lab settings was introduced. This testbed was subsequently used in a number of studies. In [5], the impact of asymmetric delay on QoE was investigated for multi-party telemeetings. The study findings confirmed the assumed negative impact of asymmetric delay: if a delay of more than 500 ms is introduced for only one participant, this has a severe negative impact on the QoE of the whole group. Interestingly, they also found that the impact of delay depends on the involvement in the conversation: less active participants were more tolerant towards delay and hardly noticed it up to 650 ms, whereas active participants became aware of a delay between 100 and 600 ms. In [11], the impact of network limitations, different layout and stream (low vs. high quality) configurations was studied. The results indicated that packet loss and packet-loss based distortions have a larger influence on users' QoE than a reduction of the video quality.

In [8], QoE issues in the context of mobile multi-party telemeetings via WebRTC were investigated. A series of interactive, three-party audiovisual telemeeting tests with different smartphone and laptop configurations was conducted in a natural environment. The results indicated amongst others that especially for videoconferencing on smartphone, participants have lower expectations. The authors also argue that it might be necessary to shift the processing burden (or part of it) required for multi-party telemeetings to a centralized conference media server (as opposed to a full-mesh topology), since many smartphones may not be able to meet the high CPU requirements needed to ensure a smooth QoE [8].

Even though the above-mentioned controlled lab studies are highly relevant in order to gain a better understanding of pertinent QoE issues in this context, we here take an alternative approach. More concretely, we gather real-time session statistics gathered by means of Google Chrome's WebRTC-internals tool and we explore their characteristics and potential value for QoE research. To this end, we ran a number of initial tests with different scenarios, which are – along with other aspects of the measurement set-up – described in detail in the next section.

III. Measurement set-up

A. Test Scenarios

We ran a series of tests in which we focused on two-party video conferencing using the appear.in application. Google Chrome was used as browser to access appear.in. Although appear.in supports both media transmission (audio and video) and screen sharing, the tests described here were limited to media transmission, and specifically on video. In this initial study, audio was not considered. Both parties involved in the calls used a laptop and were connected to a WLAN.

The two parties were synchronized to start the appear.in session and downloaded the Chrome stats at the same time. Different scenarios were tested and we here report on two of these cases.

Case 1: *Both parties have good network conditions*: Both parties experienced good network conditions, and the collected stats are used as a reference and further evaluated in order to gain a better understanding of the Chrome stats

Case 2: *Both parties have bad network conditions*: Both parties experienced poor network conditions (*e.g.*, when moving around, taking the elevator, in weak spot of WLAN connectivity)

The case where only one party has a bad connection is not highlighted in this paper since the perceived effect on QoE is the same as in the case where both parties have a bad connection that leads to similar impairment in both directions. Each scenario was ran several times with different time intervals, ranging from 5 minutes to 20 minutes, to collect as many performance variations as possible. In addition to inherited network variations, different user behavior activities were conducted such as refreshing the page, quitting and then rejoining the session, or switching video from high to low quality or vice versa. Throughout the tests, we observed how these activities affect the gathered Chrome stats. In the following, we provide more background on the collected stats.

B. WebRTC Statistics

In WebRTC services, audio, video, and data packets are transmitted over a *peer-connection*. W3C specifies a set of APIs that provide performance statistics of these peer connections. A *data channel* is a connection between two browsers for data exchange, and a *track* is a media specific channel (audio, video, screen sharing) inside the data channel. The W3C WebRTC statistics defines objects to observed RTP statistics for the data channels and tracks. The statistics can be classified with respect to codec used, certificate used, transport protocol used, and contains counters, such as number of peer connections. For a comprehensive description see [12].

Here we share some more details about the selection of statistics (referred to as *stats*), which we regard as useful for detecting potential QoE killers in this paper:

- *RTP Stream Stats* - Codec, Inbound and Outbound RTP statistics, such as packets received/sent, bytes received/sent, packets lost, jitter, round trip time, target bit rate
- *Media Stream Stats* - stream property and track identity, and media stream track statistics, including audio-stream performance (audio level, echo return loss) and video-stream statistics (frame width/height, frames per second, frames received/sent, frames decoded/dropped/corrupted).
- *Data channel Stats* - statistics for data channels, including bytesReceived/sent, messagesReceived/sent (over API), protocol used

Other stats do not directly reflect the performance of packet transmissions, but are nevertheless relevant to the quality of WebRTC services. For example, ICE Candidate stats contain information from the TURN or STUN server, which may have a significant influence on the end-to-end delay and should be considered in the overall QoE study. The specification of WebRTC APIs is work in progress so extensions and changes are to be expected in the future. Both browsers and WebRTC application developers can collect these statistics. The cases presented in this paper are primarily based on performance statistics collected via the statistics collection functionality called *webrtc-internals*, embedded in Google Chrome. In the following, we take a closer look at the stats that can be obtained from *webrtc-internals*, focusing more particularly on those stats that may be useful to detect potential QoE killers. All browsers supporting WebRTC (*i.e.*, Chrome, Opera and Firefox) provide API to developers for tracking statistics. Currently only Chrome and Opera provide such an interface to external users to access the statistics.

C. Statistics in Chrome

The *webrtc-internals*[1] functionality has been used by WebRTC application developers to understand the features and functions of their WebRTC services. To the best of our knowledge, it is relatively new to explore and utilize these stats to study Quality of Experience aspects of WebRTC services. The *webrtc-internals* functionality enables observation of the performance of the WebRTC connections locally in the browser.

In webrtc-internals, a JSON file contains all PeerConnection objects defined in W3C APIs as well as some Google-specific stats. The reports include several media and network statistics, such as (video)frame rate, packet loss, and bitrate. End users can view these statistics in real-time or download them in a single file any time during a session, or download the statistics immediately after a session (however, in that case it has to be downloaded before closing the browser window or before quitting the session in the application). The statistics are collected per browser, which means that in order to assess the performance of a multi-party session, the statistics from all browsers used in the session need to be recorded, downloaded, and (manually) combined and synchronized.

[1]Crome: chrome://webrtc-internals, and in Opera: opera://webrtc-internals

In a two-party video conference call, each end point's media stats contains at least four tracks, identified by SSRC ID, including two audio tracks (one for sending and one for receiving), and two video tracks (one for sending and one for receiving)[2]. See Fig. 1 for an illustration. In a Peer Connection

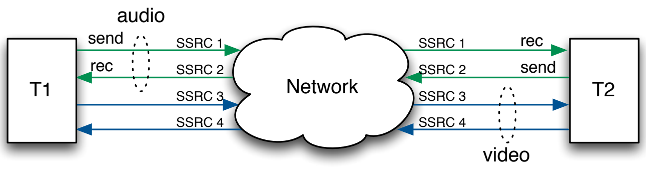


Figure 1: A two-party video conference call opens four tracks

we have $c = 2\delta_{audio} + 2\delta_{video} + 2\delta_{screen}$ tracks, where $\delta_x = 1$ if track type x is active, 0 otherwise. An n-party conversation consists of $\frac{n(n-1)}{2}$ Peer Connections which requires $\frac{n(n-1)}{2} \times c$ tracks (SSRCs). SSRC is a key ID that links the two parties of one WebRTC connection. For a Peer Connection between peer A and B, c SSRC ID (one per track) is shared between them. Peer A, identified as a *sender*, has stats labeled as *sent* (*e.g.*, *bitsSentPerSecond*) whereas peer B, identified as a *receiver*, has stats labeled as *received* (*e.g.*, *bitsReceivedPerSecond*). The stats that can be paired as *sent* and *received* include: *bitsPerSecond*, *bytes*, *packets*, *packetsPerSecond*, and *frame rate* for video streams. There are many other stats such as *RTT*, *jitter*, *packetsLost* that can not be paired but that may still be useful to determine the service quality.

D. *Other WebRTC Analytics tools*

As indicated earlier, the performance statistics provided through the the WebRTC API can also be used by WebRTC application developers. Appear.in recently launched a customized and customizable WebRTC Analytics interface called *getstats.io*. It provides real-time visualization of WebRTC service performance stats. This interface integrates the performance stats of all parties involved in a call. The stats files collected from Google Chrome and getstats.io are both stored in a JSON format, but with a different structure and to some extent a different content. In our tests, we also collected the stats from getstats.io for all scenarios. These stats are not presented or discussed in more detail here, but for the statistics that are gathered by both used tools, we compared Chrome stats and getstats.io in order to verify the accuracy of the data gathered by the former. Considering that the collected statistics are time sequences, we calculated two measures: normalized dynamic time warping (DTW) and ratio of norm-2[13]. The former gives an absolute distance between the two sequences whereas the latter shows how different are the two sequences in magnitude. DTW and norm-2 are used to validate Chrome webrtc-internals against getstats.io. They are acquired from the same source of WebRTC API, so it is expected that they are highly correlated (DTW close to 0, and norm-2 value close to 1) [13].

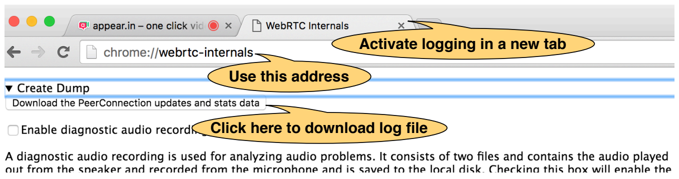
For the three parameters (bytes received/sent, packets received/sent, packets lost) common in both Chrome stats and getstats.io, the normalized DTW distance ranges from 0.03 to 0.07 and the ratio of norm-2 is between 0.98 and 0.995, both of which indicate that the two sets of data are highly similar and that the Chrome statistics can be used for our purpose.

E. *Other performance measurement tools*

Other performance measurement tools (such as WireShark and even probing nodes) are definitely complementary to WebRTC API stats and may be taken into account in the future study. Compared to the data gathered from these network performance measurement tools, WebRTC API stats are more application-oriented and can be directly linked to user experience. The network performance measurement can be used in the future for QoE troubleshooting and network diagnosis, etc. (ref [14] is on network layer for video conferencing).

F. *Data Acquisition*

1) WebRTC statistics acquisition: As mentioned above, the objective data[3] (performance data related to the appear.in session) was gathered by Google Chrome. Chrome webrtc-internals (chrome://webrtc-internals) must be activated in Google Chrome by opening *chrome://webrtc-internas* in a new tab or window. While the appear.in session is running you might leave the webrtc-internals tab, but it must not be closed. At the end of, or any time during, the appear.in session, click on "Download the PeerConnection updates and stats data" before closing the webrtc-internals tab, to make sure that the JSON file is downloaded. For illustration, see Fig. 2 for a screenshot take from Chrome browser with a webrtc-internals tab open.


appear.in – one click vid
WebRTC Internals
Activate logging in a new tab
chrome://webrtc-internals
Use this address
▼ Create Dump
Download the PeerConnection updates and stats data
Click here to download log file
Enable diagnostic audio recordi

A diagnostic audio recording is used for analyzing audio problems. It consists of two files and contains the audio played out from the speaker and recorded from the microphone and is saved to the local disk. Checking this box will enable the

Figure 2: Chrome webrtc-statistics acquisition

2) Subjective data acquisition: In our subjective study, the number of participants is limited to 4 (experts). More specifically, the two parties involved in the calls each consisted of two people. Whereas one of them was actively participating in the call, the other one was observing (video) quality issues and documenting the nature, timing and duration of the perceived quality issues during each of the sessions.

The subjective data was documented using a traditional pencil and paper approach, which is sufficient for our purpose since we are not conducting a quantitative study but rather investigating the applicability of WebRTC for QoE studies. Several quality issues were documented, but in this paper, we limit the focus to video-related quality issues (frozen image,

[2]In *appear.in* screen sharing is possible and activating this will create another pair of video tracks

[3]The experimental data of our study is available online at: https://github.com/doreidammar/webrtc-statistics#webrtc-statistics.

Table I: An example of the subjective data acquisition

Time		Video			
Start	End	Frozen image	Slow movement	Black/blank screen	Screen flash
18″	24″		✗		
2′01″	2′20″	✗			

slow movement, black/blank screen, screen flash), and more concretely, to severe video freezes. Table I shows an example of the subjective data acquisition.

G. Notation and statistics

Table II: Key variables and notations used in the paper.

notation	meaning
α	set of attributes, $\alpha = \{$bitsSentPerSecond, PacketLost,...$\}$
a	attribute, $a \in \alpha$
x_{at}	sample at t of attribute a
X_a	sample set of attribute a
τ	sample interval size

The statistics presented in next section are rate, ratio and accumulated numbers of different attributes $a \in \alpha$ observed over intervals of length τ.

- *accumulated numbers* $n_t = (x_t - x_{t-\tau})$
- *rate* $r_t = n_t/\tau$
- *ratio* $\rho_t = n_{a_1 t}/n_{a_2 t}$

for $t = 0, \tau, \cdots, m\tau$, where m is the number of intervals in the trace.

IV. Observations

A. Chrome stats in the plots

Chrome data can create performance statistics of a video conference conversation by combining the JSON files downloaded by each of the peers in the conversation. End to end statistics between peering pairs are obtained by integrating the performance stats according to the tracks identifier SSRC ID.

For example, *bitrate* is observed as *sendBitRate* and *receivedBitRate* with the same SSRC ID. This means that we have both original (or reference) data from the sender and (potentially) distorted data from the receiver, which is similar to the full reference scenario in general QoE methods. As a consequence, insights may be gained by looking at the difference between the original sending stats and the distorted receiving stats.

However, the synchronization between different streams needs to be carefully considered. When estimating the widely used QoS parameter packet loss ratio (PLR), caution is needed as the counters of received packets and lost packets are not necessary synchronized. Note that *packetsLost* is a parameter collected at the side of both sender and receiver, yet they cannot be paired as *sent* and *received* because they represent a loss of packets of different streams with different directions (different SSRC ID). In case of the Picture Loss Indication (PLI), a decoder informs the encoder about the loss of an undefined amount of coded video data belonging to one or more pictures [15]. The bucket delay is defined as "... the time since the oldest queued packet was enqueued". Note that, the latter definition was found in the source code of the WebRTC project [16].

The focus in this paper is on WebRTC performance attributes that can be paired by common SSRC ID to provide performance statistics per track. In this section we plot different statistics using WebRTC performance attributes from observed Chrome stats. Fig. 3-5 plot the following statistics for $t = 0, \cdots, 300$ ($\tau = 10$) [sec] for a specific SSRC ID:

(a) Throughput and bandwidth
- *Available bandwidth*: r_t with $a =$ 'bweforvideo-googAvailableSendBandwidth'
- *Bits sent*: r_t with $a =$ 'bitsSentPerSecond'
- *Bits received*: r_t with $a =$ 'bitsReceivedPerSecond'

(b) Packet lost
- *Packets lost*: ρ_t with $a_1 =$ 'send-packetsLost' and $a_2 =$ 'send-packetsSent'

(c) Picture Loss Indication (PLI)
- *PLIs received*: n_t with $a =$ 'googplisreceived'
- *PLIs sent*: n_t with $a =$ 'googPlisSent'

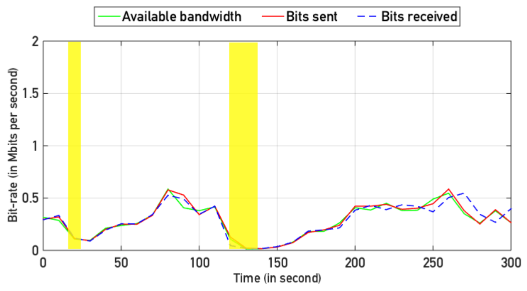
(d) Bucket delay
- *Bucket delay*: n_t with $a =$ 'bweforvideo-googBucketDelay'

See Section III-G for the notation and the definition of statistics.

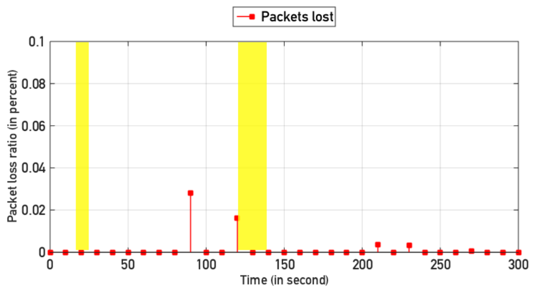
B. Observations related to the test cases

1) Case 1: both parties have good network conditions: In this scenario, both parties are located in an office with good network conditions provided by the shared WLAN connectivity. In most test cases, the media transmission between two parties turned out to be smooth and users have a rather good experience, despite of some minor disturbances such as jerkiness, flickers and glitches. However, the parties also encountered phases during which the quality was seriously impeded, potentially beyond acceptance. In Fig. 3(a)–4(d), these phases are indicated with a yellow background. Obviously, the direction from sender A to receiver B is affected to a higher degree.

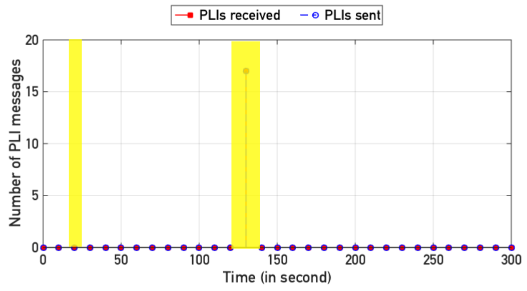
In particular, user group B perceived a short video freeze of some seconds at around 20 seconds, and a rather long freeze of about 5 seconds at 2 minutes since the start of the session. The latter made one participant to remark that this problem could have triggered that participant to terminate the session. As shown in Fig. 3, the critical phases correlate well with severe decreases of the throughput values both at 20 and at 120 seconds, respectively. These decreases are observed at both sides, which means that they have most likely been caused by sender A, and not by the network, cf. also [17]. This



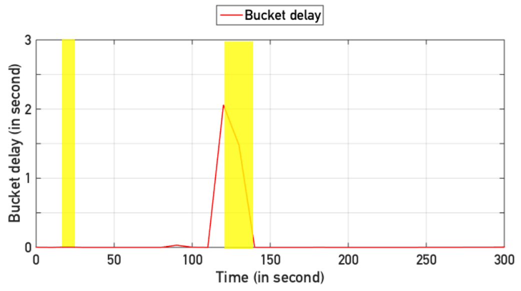
((a)) Throughput and bandwidth



((b)) Packets lost

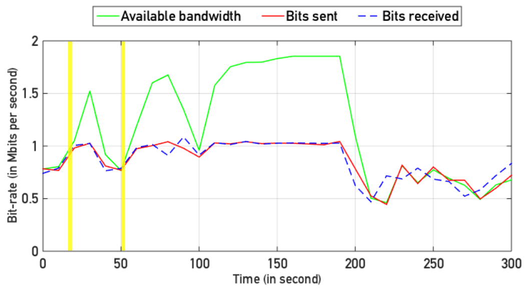


((c)) Picture Loss Indication (PLI)

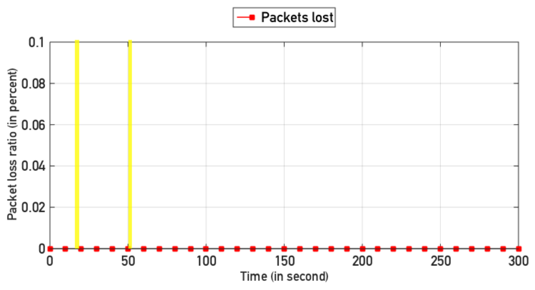


((d)) Bucket delay

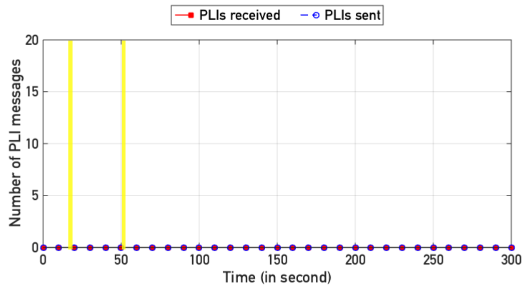
Figure 3: Case 1: both parties have good network conditions (sender A to receiver B). Critical phases of heavy picture quality degradations and freezes are highlighted in yellow.



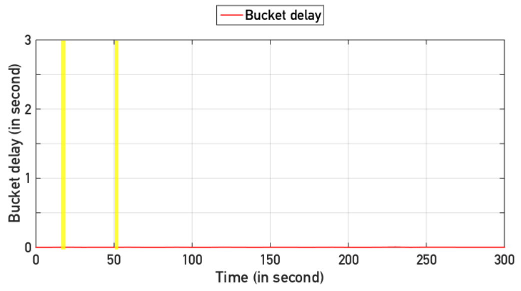
((a)) Throughput and bandwidth



((b)) Packets loss



((c)) Picture Loss Indication (PLI)



((d)) Bucket delay

Figure 4: Case 1: both parties have good network conditions (sender B to receiver A). Critical phases of heavy picture quality degradations and freezes are highlighted in yellow.

observation is underpinned by the lack of loss at the critical times, as seen from Fig. 3(b). Yet, the Picture Loss Indication (PLI) reports a significant amount of loss at 130 seconds, cf. Fig. 3(c), already at the sender. Furthermore, the bucket delay indicates the most severe disturbance at 120 seconds, cf. Fig. 3(d).

In the opposite direction, there are only two short glitches at around 15 and 50 seconds, which correlate with temporarily reduced throughput and bandwidth values, cf. Fig. 4(a). Yet, later, such reductions at 100 seconds and beyond 200 seconds did not coincide with any quality reductions that any of the observers pointed out as a potential reason for abandoning the session. In the direction from sender B to receiver A, no other statistics (packets lost; PLI; and bucket delay) show any deviation from normal behaviour, cf. Fig. 4(b)–4(d).

2) Case 2: Both parties have bad network conditions: In order to provoke bad QoE, both groups left the meeting room in favor of places that are known for bad WLAN coverage. While group A went just outside the building (where WLAN is not even supposed to provide coverage), group B was using the elevator (a Faraday's cage) in a non-stop manner during the test. We are restricting ourselves to reporting the results for one direction, namely from sender B to receiver A.

In general, both groups experienced many critical phases of heavy picture quality degradations and freezes, which are again highlighted with yellow background in Fig. 5. Generally, both throughput values and bandwidth estimations are low; typically, these data rates decrease just ahead of or during those critical phases, and the receiver rate is most of the time found below the sender rate, cf. Fig. 5(a). While the packet loss as such colocates well with most of the critical phases, cf. Fig. 5(b), the PLI does not at all, cf. Fig. 5(c). On the other hand, the bucket delay again shows a nice correlation with most of the critical phases, cf. Fig. 5(d).

C. Lessons learned so far

Two types of lessons were learned from this study. The first lesson is on the collected chrome statistics. Chrome statistics are acquired from WebRTC API and fully visualized when observing in real-time. However, after we downloaded the statistics for post-processing, some issues arise. For instance, the downloaded statistics are limited to 1000 sample points. If the appear.in (or WebRTC) session lasts longer than 1000 seconds, only the latest 1000 sample data is recorded, *i.e.*, the data older than 1000 seconds is lost. Moreover, the sampling time of the downloaded statistics varies with device and OS. Even though the real-time statistical visualization exhibits an identical sampling time of one second, the recorded samples at different devices demonstrate variance from 1 second to 3 or 4 seconds, which causes a problem with synchronizing the data from different parties. These issues need to be carefully handled when performing post-processing analysis.

The second takeaway we observed is on the candidate parameters that could be used to represent unacceptable QoE:

- Dropping data rates at the receiver;
- Non-zero packet loss ratios that are applicable to WLAN settings, while mobile links tend to turn loss into delays [18]);
- Non-zero PLI values (potentially at both sides);
- Bucket delays that indicate freezes.

The findings on these parameters are not only consistent with but also complementary to previous findings. For instance, [14] presented a model focusing on outages on mobile links, with a metric similar to bucket delay that can indicate freezing for general services from sender-side buffer behaviour observations. Our findings extend to video conferencing services. In [17], a study of network-layer performance metrics for video conferencing revealed a similar behavior as what we observed from the dropping data rates, in terms of throughput and/or available bandwidth estimation. Our work exhibits such as relationship from a perspective of application-layer performance metrics.

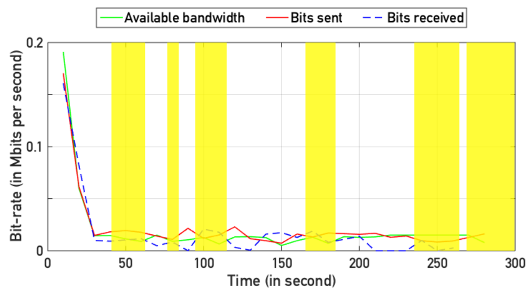
Several of those indications may appear together, which itself can be seen as an indication of nature and severeness of the disturbance. Furthermore, the comparison between sender and receiver statistics helps identifying the main contributor to the problem [17]. The good news is that the rather coarse time resolution of ten seconds does not hamper the interpretation; indeed, that kind-of low-pass filter seems to highlight the really bad conditions, while shorter and less critical issues are suppressed.

V. Conclusion

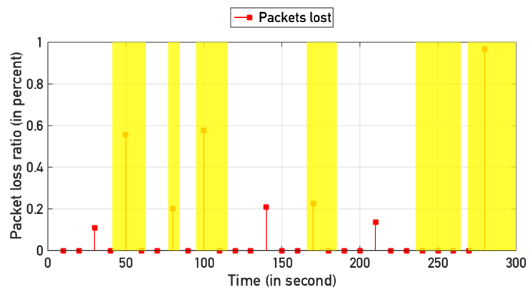
In this paper, we explored the characteristics of session statistics collected by Google Chrome's WebRTC-internals functionality and explored their potential relevance for identifying QoE issues in the context of WebRTC-based video communication. More concretely, we collected a set of statistics related to two-party video conversations in different scenarios, using the WebRTC-based video communication application appear.in.

Our observations show that – despite the limitations of the WebRTC-internals stats – several of the gathered session statistics (in particular: throughput, packet loss, and bucket delay) seem to be good candidate indicators of severe video freezes, as a key QoE killer and are therefore worthy of further in-depth investigation. The work presented here represents however just a first step, which was limited to only one type of video-related QoE killer and which did not consider audio impairments, despite the fact that audio is often considered as the natural fall-back solution in case of bad video quality.

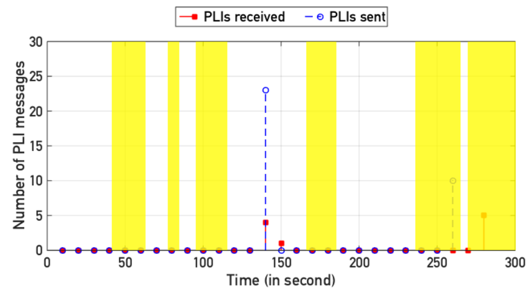
In our future work, we therefore aim to extend the focus to other types of video and audio quality impairments that can be considered as key QoE killers. In addition, more complex usage situations with multiple parties will be considered in order to gain a better insight into common and realistic problems, their causes, their implications for QoE and the behavioral response that they trigger (by jointly analyzing session statistics and behavioral measures). Moreover, we aim to investigate the impact of performance-related potential QoE killers not only in one single session, but over multiple



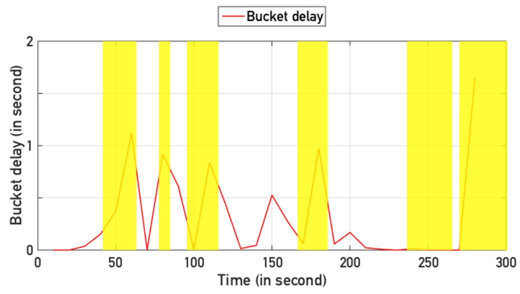
((a)) Throughput and bandwidth



((b)) Packets lost



((c)) Picture Loss Indication (PLI)



((d)) Bucket delay

Figure 5: Case 2: Both parties have bad network conditions (sender B to receiver A). Critical phases of heavy picture quality degradations and freezes are highlighted in yellow.

sessions. Such more robust and in-depth insights are needed in order to develop strategies that enable QoE improvement in real time, so that the best possible experience can be enabled, regardless of the circumstances of a call.

Acknowledgment

Doreid Ammar's work was carried out during the tenure of an ERCIM 'Alain Bensoussan' Fellowship Programme. The work presented in this paper was also supported by the NTNU - Telenor Research collaboration project on Delivering Robust End-to-End Differentiated Services (E2E DS), as well as by the Swedish Knowledge Foundation through the BigData@BTH research profile, involving BTH and Telenor Research. Finally, the authors gratefully thank the appear.in team and getstats.io team for their help and input.

References

[1] G. Berndtsson, M. Folkesson, and V. Kulyk, "Subjective quality assessment of video conferences and telemeetings," in *Packet Video Workshop (PV), 2012 19th International*, May 2012, pp. 25–30.

[2] J. Skowronek, K. Schoenenberg, and G. Berndtsson, "Multimedia Conferencing and Telemeetings," in *Quality of Experience: Advanced Concepts, Applications, and Methods*. Springer, 2014.

[3] L. De Cicco, S. Mascolo, and V. Palmisano, "Skype Video Responsiveness to Bandwidth Variations," in *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV '08. New York, NY, USA: ACM, 2008.

[4] M. Venkataraman and M. Chatterjee, "Inferring video QoE in real time," *Network, IEEE*, vol. 25, no. 1, pp. 4–13, 2011.

[5] M. Schmitt, S. Gunkel, P. Cesar, and D. Bulterman, "Asymmetric Delay in Video-Mediated Group Discussions," in *International Workshop on Quality of Multimedia Experience (QoMEX)*, Sep 2014, pp. 19–24.

[6] M. Vaalgamaa and B. Belmudez, "Audiovisual Communication," in *Quality of Experience*, ser. T-Labs Series in Telecommunication Services, S. Moeller and A. Raake, Eds. Springer International Publishing, 2014, pp. 195–212.

[7] M. Schmitt, S. Gunkel, P. Cesar, and P. Hughes, "A QoE Testbed for Socially-aware Video-mediated Group Communication," in *Proceedings of the 2Nd International Workshop on Socially-aware Multimedia*, 2013.

[8] D. Vucic and L. Skorin-Kapov, "The impact of mobile device factors on QoE for multi-party video conferencing via WebRTC," in *Telecommunications (ConTEL), 2015 13th International Conference on*, July 2015.

[9] "ITU-T P.1301: Subjective Quality Evaluation of Audio and Audiovisual Multiparty Telemeetings ," Jul 2012.

[10] J. Skowronek, F. Schiffner, and A. Raake, "On the influence of involvement on the quality of multiparty conferencing," in *4th International Workshop on Perceptual Quality of Systems, Vienna*, 2013, pp. 25–30.

[11] S. Gunkel, M. Schmitt, and P. Cesar, "A QoE study of different stream and layout configurations in video conferencing under limited network conditions," in *Quality of Multimedia Experience (QoMEX), 2015 Seventh International Workshop on*, May 2015, pp. 1–6.

[12] H. Alvestrand and V. Singh, "Identifiers for WebRTC's Statistics API," W3C, W3C Working Draft, Feb. 2015.

[13] E. Keogh and A.Ratanamahatana, "Everything You Know About Dynamic Time Warping is Wrong," in *3rd Workshop on Mining Temporal and Sequential Data, in conjunction with the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[14] M. Fiedler, J. Shaikh, and V. Elepe, "Exponential on-off traffic models for Quality of Experience and Quality of Service assessment," *Praxis der Kommunikationstechnik (PIK)*, vol. 37, no. 4, pp. 297–3046, 2014.

[15] J. Ott and D. S. Wenger, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)," RFC 4585, Mar. 2013. [Online]. Available: https://rfc-editor.org/rfc/rfc4585.txt

[16] Chromium, "The WebRTC project," https://chromium.googlesource.com/external/webrtc/.

[17] M. Fiedler, K. Tutschku, P. Carlsson, and A. Nilsson, "Identification of performance degradation in IP networks using throughput statistics," in *Proc. 18th International Teletraffic Congress (ITC 18)*, Sept. 2003.

[18] T. Minhas and M. Fiedler, "Quality of Experience Hourglass Model," in *Proc. IEEE ComManTel*, Jan. 2013.