Revealing the Dark Side of WebRTC Statistics Collected by Google Chrome

***Abstract*—Google Chrome provides a built-in tool to collect real-time session-related performance statistics of Web-based Real-Time Communication (WebRTC). Although the Chrome statistics have a number of limitations, we believe that they can be used in studies of Quality of Experience (QoE) aspects of WebRTC services. In this paper, we first reveal the limitations of the collected statistics and its consequences. We then discuss how to overcome these issues.**

Keywords*—*WebRTC, QoE, performance statistics

I. Introduction

Web Real-Time Communication (WebRTC) has attracted much attention recently due to its simplicity in developing and delivering services such as multimedia communication services, without the need for proprietary plug-ins or applications. Compatible with HTML5, WebRTC allows direct audio, video, and data communications between participating parties. The communications functionality is embedded in a browser, which provides a set of buildings blocks for interactive communications to application developers through a JavaScript API. In addition to the web and smartphone platform (Chrome, Firefox, Opera, Android, and iOS), WebRTC can also be integrated into the existing IMS (IP Multimedia Subsystem) network architecture [1]. The communication protocols are specified by the IETF whereas the JavaScript API specification is done in the W3C.

Similar to other services delivered over a best effort network, it is challenging to provide service guarantees for WebRTC and for WebRTC based services (e.g. google hangouts, appear.in). Therefore, IETF Working Group (WG) RTP Media Congestion Avoidance Techniques (RMCAT) develops proper congestion control algorithms that are the key to optimize Quality of Service (QoS) of WebRTC and Quality of Experience (QoE) of WebRTC based services, and IETF has specified packet markings for QoS in [2] to monitor the performance of WebRTC.

The objective of the paper is to present the potential and challenges with using the WebRTC statistics that is embedded in the Chrome browser, and to discuss how this can be used in studies of QoE aspects of WebRTC services.

II. WebRTC Statistics in Chrome

W3C specifies a set of APIs that provide performance statistics of the audio, video, and data packets that are transmitted over peer-connections in WebRTC services. The W3C WebRTC statistics defines *PeerConnection* objects for RTP statistics for the data channels and media streams between two browsers. Media (audio, video, screen sharing) content is represented in the *MediaStream* interface. Each MediaStream contains several *tracks* for specific audio and video content. Non-media data exchange is recorded in the *DataChannel* interface. For a comprehensive description see [3].

Chrome provides functionality for WebRTC monitoring. The *webrtc-internals* (chrome://webrtc-internals) partly implements the statistic identifiers of WebRTC peer-connections as defined by W3C [3]. It is primarily developed as a tool for testing and bug fixing while implementing WebRTC based services, but can also be applied for measuring quality of WebRTC services and facilitate QoE studies. The *webrtc-internals* records to a JSON file all the PeerConnection objects defined in W3C APIs as well as some Google-specific statsitics. End users can view statistics in real-time or offline by downloading the JSON file any time during or immediately after a session. The *webrtc-internals* functionality enables observation of the performance of the WebRTC connections locally in the browser. The main objective of webrtc-internals is to help WebRTC application developers to understand the features and functions of their WebRTC services, but it can also be used in studies of QoE aspects of WebRTC services.

As an example, consider a WebRTC connection implementing a two-party video conference call. The media stats of each end point's contain at least four tracks, which are identified by a unique SSRC ID. The SSRC ID links the two parties in a PeerConnection, including two audio and two video tracks (one for sending and one for receiving). See Figure 1 for an illustration.

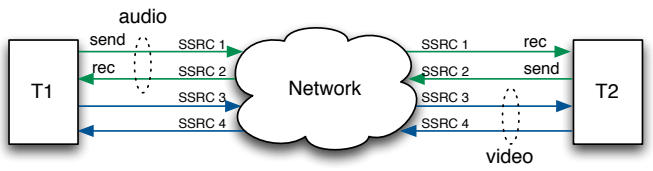

Figure 1: A two-party video conference call opens four tracks

III. The Dark Side of the Statistics in Chrome

In this section, we list the challenges that we have encountered in [4] with the use of the WebRTC statistics in Chrome.

A. Manual download of statistics: The statistics in Google Chrome are fully visualized when observing in real-time, and can also be downloaded to perform post-processing analysis, which is a crucial task in knowledge discovery. End users must download the statistics before closing the browser window or before quitting the session in the application to prevent data loss. Crashed sessions entail loss of the corresponding statistics. The manual download of the statistics is cumbersome

and must be automated, in particular for longitudinal "living lab" empirical study.

B. Limited number of sample points: The downloaded statistics are limited to 1000 sample points, which means only the latest 1000 sample data are recorded and the older data will be lost.

C. Undocumented Chrome statistics extensions: The analysis of the downloaded statistics is difficult due to the lack of clearly documented definitions of the Chrome statistics attributes. More specifically, the extension of the attributes defined in WebRTC statistics is undocumented, which causes some challenges in reliably analyzing the collected data.

D. Imprecise sampling time: Chrome 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. This implies that the statistics in the log files must been recorded at the same time, which means that the devices must use the same sampling times intervals and have synchronized clocks. Currently, Google Chrome does not provide the possibility to change the sampling time of the downloaded statistics. Moreover, the sampling time varies with device and OS. We have observed [4] that the recorded samples at different devices was ranging from $1-3$ seconds, which cause a problem with synchronizing the data from different parties. Figure 2 shows an example (the devices used to collect these data are related in Table I) where packets are apparently received before sending them in a two-party video conference call. This clearly wrong, but when the synchronization is off with no such visible effect, this is more difficult to detect. A general advice is to always address these issues when performing post-processing analysis.

Table I: Sampling time with two different devices

Device	Platform	OS	Chrome version	Sampling time
M1	Macintosh	Mac OS X 10.8.5	46.0.2490.71	1 second
W1	Windows	Windows 7	45.0.2454.101	*random* $\in [1.05; 1.08]$

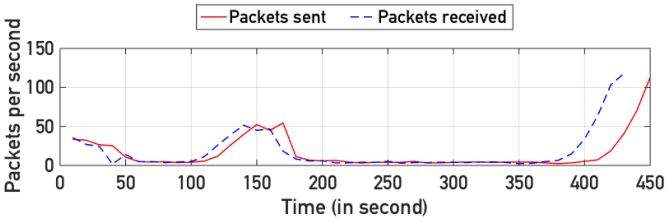

Figure 2: Packet transfer rate (sender *M1* to receiver *W1)*

E. Integrating Chrome statistics with other data sources: When conducting QoE studies on WebRTC we want to combine different measures from various data sources to gain valuable insight in root causes and effect on QoE. Chrome statistics will be one such a source which provides information about status and performance of the peering point in WebRTC based services. Integrating and aligning data sources with (completely) different data formats, although this is not unique for Chrome statistics integration, this must be addressed carefully and will require some effort - in particular if the format is not well documented.

IV. Proper Use of the Statistics in Chrome

The fact that only the most recent 1000 sample points are captured (Section III.B) shifts the focus from long-term to recent time horizons; in particular, experiments should not exceed 1000 seconds. Indeed, statistics that are downloaded after the occurrence of a major QoE-relevant event can be very helpful for root-cause analysis (thinking a.o. of the "Black Box" in an aircraft). Ideally, this would be initiated automatically in case of a service and/or browser crash, which is however not the case so far.

If the clocks of the users' devices are not properly synchronized, or in case of sampling time mismatches (Section III.D), the collected statistics need to be adjusted by shifting them to the correct starting time while taking into account the time difference between the different devices. For instance, a procedure for removing non-causality (receiver ahead of sender, *cf.* Figure 2) is found in [5]. Here, the availability of individual time stamps per sample would support such correction measures.

Increasing averaging time scales reduce the sensitivity to time-stamping errors. Thus, it may be beneficial to determine the maximal timescale that still allows to identify the misbehaviors of interest. For instance, the investigations for [4] revealed the applicability of 10 second-averages for indicating acceptance-critical freezes.

Actually, certain summary statistics, such as moments and distributions of bit or packet rates, have shown to be rather robust to timestamp divergences [6], [5]. Still, they are of great help when it comes to identifying bottleneck behaviour [6]. Thus, we recommend the use of such summary statistics as much as possible. However, other insufficiently documented statistics extensions should be used with great care.

References

[1] 3GPP Technical Specification Group Services and System Aspects, "Study on Enhancements to Web Real Time Communication (WebRTC) Access to IP Multimedia Subsystem (IMS): Stage 2," Sept. 2015.

[2] P. E. Jones, S. Dhesikan, C. Jennings, and D. Druta, "IETF Draft: DSCP and Other Packet Markings for WebRTC QoS," Mar. 2016.

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

[4] "Video QoE killer and performance statistics in WebRTC-based video communication," submitted to *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*, Jul. 2016.

[5] M. Fiedler, L. Isaksson, S. Chevul, J. Karlsson, and P. Lindberg, "Measurement and analysis of application-perceived throughput via mobile links," in *Proc. HetNets05*, Jul. 2005.

[6] 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.