Measuring Objective Visual Quality of Real-time Communication Systems in the Wild

Chih-Fan Hsu
Department of Computer Science
University of California, Davis
US
hsuchihfan@gmail.com

Chun-Ying Huang
Department of Computer Science
National Yang Ming Chiao Tung University
Taiwan
chuang@cs.nctu.edu.tw

Xin Liu
Department of Computer Science
University of California, Davis
US
xinliu@ucdavis.edu

***Abstract*—During the outbreak of epidemic diseases, the demand for real-time communication (RTC) systems has dramatically increased. People use RTC systems to communicate with each other, attend online courses, present projects, and share videos. An open question, how to choose an appropriate system for high-quality communication with different network conditions and usage scenarios, comes RTC users. In this paper, we propose a scheme to systematically measure the visual quality of RTC systems. Our scheme does not need to modify systems and thus can be easily adapted to other RTC systems and video transmission systems, i.e., streaming systems. Based on this scheme, we measure six commercial RTC systems in four common usage scenarios. We measure the received video quality (graphical quality and frame rate) at the receiver and the upload bitrate at the sender. Furthermore, we propose a novel metric, area loss, to measure the system ability to handle insufficient bandwidth using video freezing and frame delay. Our detailed analysis reveals the advantages and disadvantages of each system. We expect the measurement scheme, the metric, and our findings can help the future development of RTC systems.**

***Index Terms*—Real-time communication systems; Performance evaluation; Performance analysis**

I. Introduction

Real-time communication (RTC) systems, such as video conferencing systems or webinar systems, are common services in our daily lives. People use RTC systems to communicate with friends, attend online courses, present projects, or share videos. Due to the outbreak of epidemic diseases and the stay-at-home policy, countless companies, government facilities, and schools are temporarily closed and decided to move meetings, presentations, and lectures to RTC services.

Given the diversity of RTC systems on the market, a systematic comparison is desirable for consumers to select their most appropriate service and developers to improve their product. However, understanding commercial RTC systems is challenging because the systems are not required to report their performance. Even if they do, the performance metrics could be inconsistent or non-intuitive to consumers [1]. For example, the packet loss itself is not a meaningful metric to consumers. However, it may cause video distortion or freezing, which can directly deteriorate the user experience [2]. Therefore, to fairly compare different systems, researchers and developers strive to measure performance using carefully designed methods [1], [3]. However, what performance should be considered and how to measure the performance for RTC systems need to be further explored.

This paper focuses on visual quality from consumer's perspective and targets two performance categories of RTC systems. We first discuss the upload bitrate and perceived video quality. We then discuss the ability to handle the insufficient bandwidth situation. The former investigates the trade-off between visual quality, frame rate, and bitrate consumption. Ideally, RTC systems should use bandwidth efficiently when the network condition is good. The latter aims to investigate a system's responses when the bandwidth is below their normal requirement (insufficient bandwidth situations). A well-designed system may trade its visual quality for smoother communication. Adaptive bitrate mechanisms are often used to improve user experience by degrading the visual quality to reduce bitrate to prevent video freezing in poor network conditions, as in streaming and live broadcasting services [4]. However, those mechanisms designed for video streaming cannot be directly applied to RTC systems since RTC systems require immediate communication (much shorter response time and much smaller buffer acceptable) [5]. In this case, RTC users still suffer from video freezing during communication when the bandwidth is insufficient and eventually deteriorates their conversation quality [2]. The situation becomes severer when users are sharing their screen for attending an online presentation or lecture. To quantify the impact level caused by insufficient bandwidth, we introduce a novel metric, **area loss**, which compares the difference between the ideal frame sequence and the received frame sequence to consider video freezing and frame delay in combination. Besides, we design a measurement method to evaluate the target performance of RTC systems. Our scheme avoids the performance degradation caused by modifying the original systems with embedding additional codes or functions into systems. Besides, the method can be adapted to other systems, such as streaming, web broadcasting systems, or mobile devices.

Overall, we consider four critical performance metrics related to visual quality in this study, (1) the graphical quality and (2) the frame rate at the video receiver, (3) the upload bitrate at the video sender, and (4) the ability to handle the insufficient bandwidth situation. It covers four common scenarios in two major usages scenarios, video conferencing

(VC)-based and screen sharing (SS)-based scenarios. We adopt our method to six heterogeneous RTC systems, including web-based systems: Facebook Messenger Rooms (FB), Google Meet (GM), and Jitsi Meet (Jitsi), as well as software-based systems: Microsoft Teams (MST), Skype, and Zoom.

II. Related Work

Early-stage performance measurement for RTC systems mainly focused on VoIP services such as Skype [6]–[8]. As the development of WebRTC becoming mature, this free and open-source service allows users to directly start a real-time communication by a web browser that is independent of operating systems or platforms. Besides, WebRTC provides built-in APIs to collect statistics for researchers directly [2], [9]–[11]. Gradually, the research focus shifted to WebRTC. Carlucci et al. [9] proposed a congestion control algorithm (Google Congestion Control) to WebRTC to adapt the sending rate for handling dynamic network capacity and evaluate the effectiveness using metrics including network utilization, queueing percentile, loss ratio, and fairness index. Garcia et al. [10] proposed a toolbox by adding lightweight WebRTC clients to test the scalability of WebRTC and monitor the quality of services (QoS) provided by APIs. Ammar et al. [2] investigated the relationship between the WebRTC-internal statistics (throughput, packet loss, and bucket delay) and severe video freezes. The authors found that the video freezes dramatically impact the quality of experience (QoE) in WebRTC. Jansen et al. [11] conducted a detailed evaluation of the WebRTC system. They revealed that (1) the selective forwarding unit (SFU) topology significantly improve the performance of multi-party video call and (2) WebRTC suffered from poor performance over the wireless network due to bursty packet losses and retransmission. Dunja Vučić and Lea Skorin-Kapov [12] used WebRTC APIs to monitor mobile device session-related data to investigate the relationship between different video encoding parameters (bitrate, frame rate, and resolution) and the QoE of WebRTC. They further investigated the relationship between two video quality impairments, blockiness and blurriness, and the QoE. The result reveals the relationship between the blockiness and the QoE is Birnbaum-Saunders distribution; the relationship between the blurriness and the QoE is Burr and Gamma distributions.

To directly collect performance statistics of an RTC service, using built-in APIs is highly efficient and accurate. However, comparing numerous public available systems by built-in APIs might be problematic for two reasons. First, not all systems provide callable APIs. Second, the statistics reported from different services providers might not be the same and might not be fair to compare. Therefore, a method to collect the performance statistics for commercial systems is desirable.

Embedding codes in the system is another approach to collect performance statistics of RTC. Garcia et al. [13] proposed a benchmark platform to record videos and audios at the receiver side correspondingly. Various full-reference metrics of visual and audio qualities of WebRTC are comparable with the reference videos and audios. They revealed that the video's graphical quality is sensitive to packet loss and jitter, while the audio quality is sensitive to packet loss but less sensitive to jitter.

However, the code embedding method inevitably impacts performance metrics' preciseness, which is highly sensitive to the computing resources (CPU cycles and memory), such as frame rate and latency, because the operating system has to allocate resources to execute the additional functions. Besides, embedding codes to the systems might be problematic for some public available RTC systems, such as Zoom and Skype, because the source code is secured.

Using the third-party testbed or equipment to measure performances among different systems seems practical to avoid recording performance statistics by APIs or embedding codes into systems. The indirect measurement methods treat the target systems as black-boxed, and no additional functions or codes to compete for computing resources. The method has been adopted to compare the performance of commercial screencast systems [14] and a few other RTC systems [3]. Fouladi et al. [3] proposed a testbed to embed a QR code-like barcode in the video and send the preprocessed video to the video sender. Then, the testbed records the video on the receiver side to measure the graphical quality and frame delay of the RTC system. The authors use the testbed to demonstrate the performance advantages of the proposed system. Hence, they merely compared the received graphical quality and the end-to-end delay in a fixed laboratory scenario.

III. Experiment Scope

A. Target RTC Systems

We compare six commercial RTC systems in our experiment. These systems run on Windows and macOS operating systems. According to the software installation requirement, we separate the systems into web- and software-based systems. The web-based systems use a web browser to attend an RTC session, including FB, GM, and Jitsi; the software-based systems use their software package to join the session, including MST, Skype, and Zoom. Google Chrome (v84.0.4147.135 64bit) is served as the web browser for web-based systems. The MST, Skype, and Zoom versions are v1.300.21759, v8.63076, and v5.1.3, respectively. We note that MST, Skype, and Zoom also have web-based versions. To evaluate the full performance of the services, we evaluate the performance of the software versions.

B. Target Scenarios

Two major usage scenarios, video conferencing (VC) and screen sharing (SS)-based scenarios, are selected as our targets. For each major usage scenario, two common usage scenarios are included. The VC-based scenarios include a one-on-one communication scenario and an online exercise class scenario. The SS-based scenarios include a lecture presentation scenario and a movie sharing scenario. Ideally, in the VC-based scenarios, a webcam is used to capture most of the user's frontal face and upper body while he/she talks or the trainer's body movements in real-time. However, even if we

(a) Interview

(b) Yoga

(c) Lecture

(d) Movie

Fig. 1: Four common scenarios of RTC systems.

carefully design and ask a user to play by our script during the experiment, the user's motion, gestures, and voice would have inevitably changed in each recording. Then, the slight changes result in incomparable performances. To avoid such disturbance, we set up and show a pre-recorded video clip on a second screen for the webcam to capture to ensure the scenarios are identical and consistent.

Four video clips are carefully selected, representing the four common usage scenarios, (1) one-on-one video communication (Interview), (2) a remote exercise class (Yoga), (3) lecture presentation (Lecture), and (4) movie sharing (Movie). Each video is three-minute-long with a resolution of 1280×720 pixels (px) at 24 frames per second (FPS). Figure 1 shows an example frame of each scenario. All videos contain 180 seconds, and the raw sizes of the videos are 5.56 GB. We compressed each video by the H.264 video codec with Constant Rate Factor 23. The compressed sizes of the Interview, Yoga, Lecture, and Movie are 19.8, 17.1, 4.6, 84.2 MB, respectively. The compression rate can indicate the level of dynamics of each scenario. Overall, the Movie and the Lecture scenarios are the most dynamic and static scenarios, respectively. The Yoga scenario achieves a better compression rate than the Interview scenario because of the total area of moving objects in the scene. Although the body movement in the Yoga scenario is larger than the Interview scenario, the number of moving pixels in the Yoga scenario is fewer than the number of moving pixels in the Interview scenario.

IV. Methodology

A. Target Performance Metrics

To compare performances among RTC systems, we measure four critical performance metrics.

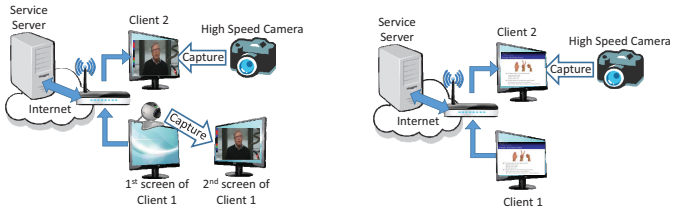
- **Two visual qualities: graphical quality and frame rate at the video receiver**. Video quality on the receiver side is critical to an RTC user because video impairments directly damage the QoE of users [13]. We further separate the video quality to graphical quality and frame rate (FPS). The graphical quality represents the compression artifacts caused by the codec. The frame rate represents video freezing and delays caused by packet losses and jitter.
- **Upload bitrate at the video sender**. The upload bitrate indicates (1) the bandwidth requirement of an RTC system, (2) the compression ability of a codec, and (3) the upper bounds of the video quality for all receivers. Besides, the video sender's upload bitrate affects the video receiver's download bitrate. Therefore, we measure the video sender's upload bitrate to investigate the relationship between the upload bitrate and the corresponding received video quality.
- **The ability to handle insufficient bandwidth**. From the RTC users' perspective, they are looking for smooth communication. In this case, we investigate the events of video freezing and frame delay caused by the insufficient bandwidth to evaluate the system's service quality.

  Compared to controlling the video receiver's download bandwidth, the upload bandwidth attracts our attention because (1) the upload bitrate is considered the performance bottleneck of an RTC system, that is relatively smaller than the download bandwidth, and (2) the insufficient upload bandwidth deteriorates the service quality of all video receivers, while the download bandwidth only affects one user.

B. Experiment Setup

Our experiment mainly consists of two clients, a video sender (Client 1) and a video receiver (Client 2). Both Client 1 and Client 2 have their cameras and connect to an RTC session via Wi-Fi. Figures 2(a) and 2(b) show the setup difference between the VC-based and the SS-based scenarios. For the VC-based scenario, we set up and show a pre-recorded video clip on the second screen for the webcam to capture, as shown in Figure 2(a), to ensure the scenes are identical and consistent in every recording. For the SS-based scenarios, we directly share the same content displayed on the screen to the remote participant without a second screen, as shown in Figure 2(b). During the experiment, we found out that the target systems except MST and GM connect as peer-to-peer because both clients connect to the same Wi-Fi router. To avoid the unfair comparison, we introduce a third client (the video and audio are always turned off) located in a different geographic continent to force packets transmitted through a service server. We force the service server to locate on the same continent to reduce the impact of the geolocation of the server. We verified that adding the third client will not affect the system's performance. During the experiment, only the camera on Client 1 side is on, as a sender, and Client 2 plays as a receiver with its camera off. The setting is to prevent any undesirable bandwidth fluctuation due to the bandwidth competition in the local wireless network.

For **graphical quality**, we firstly record the video displayed on Client 1 in advance as the reference video. Then, we record the received scenes on Client 2 side by screen recording software in full-screen mode as the target video. We select the reference video and the target video with the same frame



(a) VC-based scenarios (b) SS-based scenarios

Fig. 2: The experiment setup for recording the system performance. The blue arrows represent the packet flows.

number by the color codes to calculate graphical quality. To avoid the unnecessary graphical quality degradation caused by each RTC system's user interface, we only compare the center area of the frames cropped from the reference and target videos. Specifically, we carefully (1) align the physical positions of the reference and the target videos and (2) crop the center area by removing 10% of the width and the height of the periphery area.

For **frame rate**, a high-speed camera on Client 2 emulates what human see at the receiver. We recover the frame numbers captured by the high-speed camera to measure the frame rate. The **upload bitrate** is recorded on Client 1 for different systems and scenarios. We limit the upload bandwidth at the video sender to test the system's ability against the insufficient bandwidth conditions. Specifically, we use a traffic shaper to limit the network bandwidth. Defining a fair bandwidth limitation to test all systems is difficult because the encoding strategies and the system implementations are diverse. The limited bandwidth can be seriously insufficient to a system but far above the bandwidth requirement of the others. Hence, we limit the bandwidth by a certain percentage of each system's normal upload bitrate usage on each scenario (Figure 4) to ensure the limited bandwidth is lower than the corresponding bandwidth requirement.

C. Experiment Procedure

To avoid unnecessary bandwidth fluctuation during the recording, we recorded our experiment at midnight and constantly checked the available bandwidth. Once the Internet condition is stable, the recording is started. If the Internet fluctuates during recording, we immediately terminate the recording until the Internet is stable. We check the target RTC systems' configuration accordingly, then check the IP address for transmitting and receiving packets to ensure packets are transmitted through the service server in advance. During the experiment, the video sender starts playing a video after the recording starts on the receiver side in the recording stage. During video playing, the targets to be recorded are different according to the scenarios. For the graphical quality, we record the screen by a screen capture software; for the frame rate, we record the screen by a high-speed camera. We adopt WireShark to monitor the bitrate usage on the sender side.

In the cases of insufficient bandwidth situations, the video starts with unlimited bandwidth. After 40 seconds, the bandwidth is set to a certain percentage of the ordinary upload bitrate for one minute to simulate a sudden bandwidth change. Then, the bandwidth limit is set to unlimited. An additional 80-seconds period is used to monitor the system recovery.

D. Color Code Embedding

we design a color code and embed the code in the videos to match the frames between the sender and receiver. The color code embedding method greatly reduces unnecessary matching errors by matching with image similarity. The code is designed by Quaternary numeral system. Namely, four colors, black, blue, green, and red, represent four numbers from zero to three. We embed eleven color codes at the left of each video frame vertically. The first seven color codes indicate the frame number. The rest of the four codes are the reference colors used to obtain the true colors of four numbers captured by the webcam. The size of each color code is 50×50 px. Overall, the embedded color codes occupy about 3% area of a video frame.

E. Hardware

Client 1 (sender) is a laptop with an Intel® Core™ i7-9750H@2.60GHz, 16GB system RAM, 8GB video RAM, a 15.6-inch display (1440×900 px at 60 FPS), and Windows 10 operating system. The second screen of Client 1 is a ASUS 27-inch display (VZ279HE, 1440×900 px at 60 FPS). The webcam installed on Client 1 is Logitech C170 (640×480 px at 30 FPS). The Client 2 (receiver) is a laptop with an Intel® Core™ i5-8259U@2.30GHz, 16GB system RAM, 1.5GB video RAM, a 13.3-inch display (2880×1800 px at 60 FPS), a built-in camera (1280×720 px at 30 FPS), and macOS Catalina v10.15.5 operating system. The Internet's download (upload) speeds are 100 (5) Mbps. An iPhone 8 serves as the high-speed camera, and we set the recording parameters to 1080p at 240 FPS.

V. Measurement Results

A. Received Video Quality

Frame Rate. Table I shows the frame rates (FPS) of received videos after preprocessing. There is not much difference between the Interview and the Yoga scenarios in VC-based scenarios, in both web- and software-based systems. We can also see this trend between the SS-based scenarios. We conclude that the received frame rates of the target systems do not depend on the video content. In the VC-based scenarios, the frame rate of target systems are mostly larger than 20, except FB and GM, which are only around 14.4 and 18.9, respectively. In the SS-based scenarios, the received frame rates between web-based and software-based systems are significantly different. The web-based and most software-based systems only receive around 5 FPS and 15 FPS, respectively. The low FPS indicates that both system categories are not designed to transmit a dynamic movie when sharing screens, especially web-based systems. For the Movie scenario, Zoom outperforms others on FPS because Zoom introduces a special mode for sharing dynamic videos. Specifically, Zoom lifts the

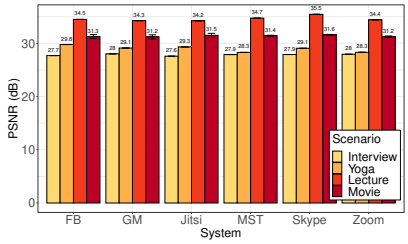


Fig. 3: The PSNR values of the target scenarios and systems. The intervals are the 95% confidence intervals.

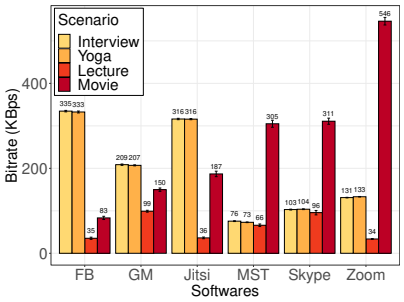


Fig. 4: The average of ordinary upload bitrate for each scenario of the investigated systems. We limit the bitrate to simulate the insufficient bandwidth situation.

implicit bitrate restriction when sharing a dynamic video to maintain high FPS for user experience. Therefore, a significant increase in upload bitrate usage occurs (Figure 4). The result reveals that there is still room to be improved on FPS for the screen sharing scenarios. **Overall, the received frame rate only depends on the usage scenario and the system implementation.**

TABLE I: The FPS of the Target Scenarios and Systems.

		Web			Software		
		FB	GM	Jitsi	MST	Skype	Zoom
VC	Interview	14.4	18.9	22.4	22.9	22.6	21.0
	Yoga	14.7	19.6	23.4	23.7	23.6	21.7
SS	Lecture	4.8	4.9	4.9	15.2	15.0	11.2
	Movie	4.8	5.0	4.8	15.0	15.2	21.3

Graphical Quality Two full-referenced objective metrics are considered to measure the graphical quality, Peak-signal-to-noise ratio (PSNR) and Structural Similarity Index (SSIM). Because the trend of the SSIM values is similar to the PSNR values, we only report the values of PSNR in Figure 3 to satisfy space limitation. We align the videos recorded at the sender and the receiver by the video frame numbers decoded from the color code. As we can observe, there is no significant difference in graphical quality among the target systems. Generally, the SS-based scenarios achieve higher graphical quality than the VC-based scenarios. The result is intuitive because the graphical quality is the basic requirement for SS-based scenarios. Between SS-based scenarios, the Lecture scenario achieves higher graphical quality than the Movie scenario. It is because the lecture presentation is usually monotonic and static. **Overall, the graphical quality depends on the usage scenario and video content.**

B. Upload Bitrate

Figure 4 reports the average upload bitrate in the target scenarios of each system. The difference between the two categories of the target systems is noticeable. Although the frame rates and graphical quality of the web-based systems are slightly lower and similar to the software-based systems, web-based systems consume more bitrate than software-based systems in the VC-based scenarios. The result reveals that web-based systems are generally less efficient than software-based systems on bitrate usage. Among all target systems, FB performs worst on the bitrate usage efficiency in the VC-based scenarios. Between the Interview and the Yoga scenarios, both scenarios have similar bitrate usage. As we mentioned before, both VC-based scenarios have similar received frame rates for each system, even though the Interview scenario is more dynamic than the Yoga scenario. The result concludes that the bitrate usage and the frame rate in the VC-based scenarios are identical and limited by the systems. It is not intuitive because we expect the bitrate should be varied based on the transmitted content to maintain a stable perceived video quality to users as the system does in the SS-based scenarios. Therefore, there is still room for further improvement.

On the other hand, web-based systems achieve a relatively lower bitrate than software-based scenarios for SS-based scenarios. The benefit is caused by the low-frame-rate strategy of the web-based systems for handling the screen sharing scenarios. Between the Lecture and the Movie scenario, the bitrate usage of the Movie scenario is greater than the Lecture scenario. The result is intuitive because the bitrate usage of a dynamic video is usually larger than the static one when they have similar received graphical qualities and frame rates after the temporal compression. The bitrate usage of the Movie scenario of Zoom is higher than the other software-based systems because of the special mode. The mode aggressively consumes the bitrate for achieving a higher frame rate without decreasing the graphical quality.

Overall, the upload bitrate is restricted in the video conferencing scenarios. The upload bitrate depends on the transmitted content in the screen sharing scenarios. The software-based systems are more efficient than the web-based systems on the upload bitrate in our investigated systems.

C. Ability to Handle Insufficient Bandwidth

We measure the received frame rate, the video freezing, and the frame delay by analyzing the ideal and actual sequences of received video frame numbers. Three bandwidth limits below their normal bitrate usages are investigated in our experiment, without limit, 70% limit, and 40% limit. Figure 5 reports the percentage of received frames among three bandwidth limits. As expected, the number of received frames significantly drops under insufficient bandwidth conditions in most of the target systems, except FB and GM. Besides, the bandwidth variation

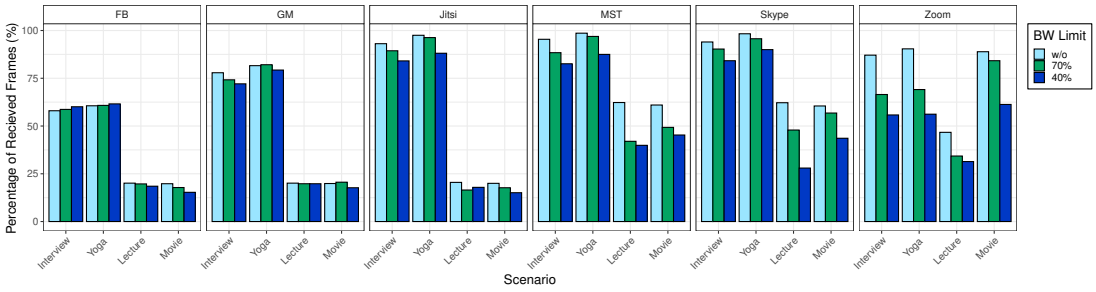


Fig. 5: The percentage of received frames under three bandwidth limits.

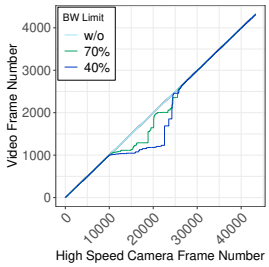
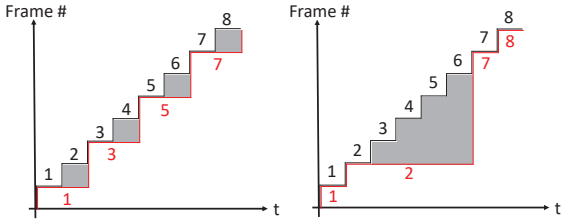


Fig. 6: The received frame numbers under three bandwidth limits of the Lecture scenario of Zoom.



(a) FPS adjustment (b) Video freezing

Fig. 7: The area loss is less sensitive to the FPS reduction.

has less impact on web-based systems than on software-based systems.

We count the number of video freezing where the duration is longer than one second (Table II). Without any bandwidth limitation, it appears that all investigated systems are free of video freezing in VC- and SS-based scenarios. FB, GM, MST, and Skype achieve almost no video freezing in the VC-based scenarios under all bandwidth limits. Generally, the SS-based scenario is more vulnerable than VC-based scenarios under insufficient bandwidth conditions.

TABLE II: The Number of Video Freezing (unlimited/70%/40% Bandwidth Limits).

	Interview	Yoga	Lecture	Movie
FB	0/0/0	0/0/0	0/0/3	0/0/16
GM	0/0/1	0/0/0	0/1/0	0/0/2
Jitsi	0/0/6	0/0/4	0/6/2	0/4/16
MST	0/0/1	0/0/0	0/2/10	0/5/3
Skype	0/2/1	0/1/1	0/8/16	0/4/5
Zoom	0/10/2	0/11/2	0/12/15	0/5/3

We further investigate the difference in display timing based on the received video frame sequences captured by the high-speed camera. Figure 6 shows an example of the received video frame numbers and their displayed timing in the three bandwidth limits. In this figure, video freezing and frame delay can be observed. Generally, *the more bandwidth is limited, the received video frame sequence more deviates from the sequence without bandwidth limitation*. A horizontal pattern indicates the video freezing on a specific frame number. The high-speed camera captures the number for a while. Once the system detects the bandwidth decreasing, the system tries to adapt the transmission rate to satisfy the bandwidth by reducing the frame rate or the graphical quality. If the freezing time is too long, the system will drop the late frames to catch up with the timely frame. In this case, we can observe a vertical pattern in the figure. Then, the received frame number sequence gradually gets closer to the sequence without bandwidth limitation.

There is an ideal frame number sequence because the original frame rates of the videos and the capturing speed of the high-speed camera are fixed. By comparing the received sequences to the ideal sequence, we can measure the system ability to handle insufficient bandwidth. Figure 9 shows the timing difference between the ideal and the received sequences. The line deviates to zero also indicates the delayed display timing of a frame. The dramatically decreasing and increasing indicates the frame freezing and the frame jump to catch up with the display timing, respectively. This figure also reveals the previously concluded result: the SS-based scenarios are more vulnerable than the VC-based scenarios in the insufficient bandwidth.

Among scenarios and systems, the Lecture scenarios of MST and Zoom are most affected by the insufficient bandwidth. In the MST case, under the 70% bandwidth limit, the video at the receiver side freezes for about 50 seconds, which indicates the insufficiency of the system's rate control mechanism, and the outgoing packets are accumulated in the network. The dramatic packet accumulation results in severe packet losses. Then, the video starts freezing. In the 40% limit condition, the system's rate control mechanism adjusts the bitrate to adapt to insufficient bandwidth twice. However, the insufficient bandwidth is still beyond the system's ability to handle. Then, the video freezes for the longest period until

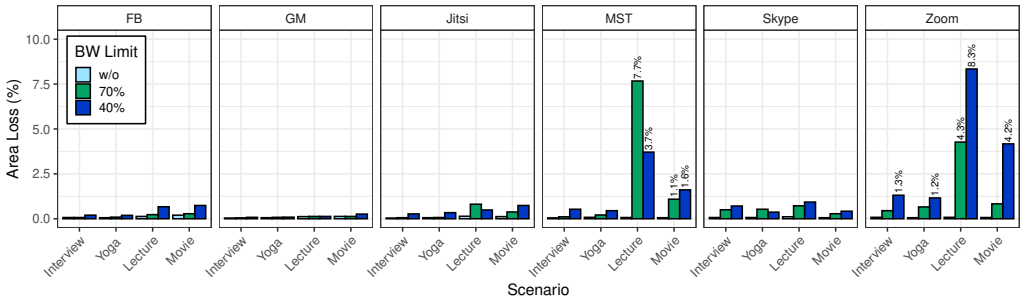


Fig. 8: The area loss of each target system and scenario. Smaller values represent a system that avoids frame freezing when insufficient bandwidth.

the bandwidth limit is lifted. The result is not intuitive because we expect the freezing time would be longer in the experiment of the 40% limit than the time in the experiment of the 70% limit. We redo the experiment setting multiple times, and all experiments show similar results. We suspect that the rate adaptation mechanism of MST adopts a fast response strategy to handle the dramatic bandwidth decreasing, but the system cannot handle very harsh situations.

We calculate the area-under-line A of the ideal sequences and the received sequences by $A = \sum_{t=0}^{T} F_t$, where F_t is the captured video frame number at time t. If a system can handle insufficient bandwidth, the area difference between A_{ideal} and $A_{received}$ should be small. Then, a new metric **area loss**, $ALoss = \frac{A_{ideal} - A_{received}}{A_{ideal}}$, is proposed to measure the adaptation ability of an RTC system.

To further explore the area loss. We look into the impact caused by the frame rate changing. Figure 7 shows examples of a 50% frame rate degradation caused by the frame rate adjustment (Figure 7(a)) and the video freezing (Figure 7(b). The step curves in the figure indicate the captured video frame number F_t at time t. The black and the red curves represent the ideal and the received sequences, respectively. Figure 7(a) shows that the system regularly skips one frame in two frames to satisfy the network bandwidth. Figure 7(b) shows the example that the video is freezing at the 2^{nd} frame, and the freezing is recovered at the 7^{th} frame. As we can observe, the area losses caused by the frame rate adjustment (4/36) are relatively lower than the area loss caused by the video freezing (10/36). We can conclude that the area loss is *less sensitive to the frame rate adjustment*. It is a vital characteristic because regular decreasing the frame rate is usually more acceptable than video freezing to users.

Figure 8 shows the area losses of the three bandwidth conditions for all investigated systems. The values of the area loss above 1% are listed above the bars. Based on the figure, we can observe: (1) Generally, the SS-based scenarios are more vulnerable than the VC-based scenarios. The result is intuitive because the SS-based scenarios are video quality demanding, which means that the video is more likely to suffer from the video freezing and frame delay for maintaining high graphical quality. (2) The web-based systems generally achieve better ability to handle insufficient bandwidth than the software-based systems in both VC- and SS-based scenarios. (3) In the software-based systems, the Lecture scenario is more vulnerable than the Movie scenario. The result is not intuitive because the Movie scenario is much dynamic than the Lecture scenario. We suspect that it is caused by sharing the static and monotonic content in the screen sharing mode. The system can significantly reduce the video bitrate because of temporal consistency. The instant bandwidth drops cause packet losses and result in the retransmission of I-frames (the reference frame for temporal compression). In this case, the limited bandwidth is extremely low because ordinary bitrate usage is very low. Therefore, the bandwidth cannot satisfy the suddenly increased bitrate for retransmitting the I-frame and eventually causes the video freezing and the frame delay.

We calculate an overall loss based on the area loss to directly compare the system's ability to handle the insufficient bandwidth. The intuitions of the overall loss follow: (1) we expected the system adapts the transmission rate to handle the insufficient bandwidth conditions, and (2) we can slightly tolerate the video freezing and the frame delay in harsh situations. Based on the intuitions, we design the overall loss based on a weighted sum of the area loss. Namely,

$$Loss = \frac{1}{|S|} \sum_{s \in S} \sum_{b \in B} b \times ALoss_s^b, \tag{1}$$

where s represents a scenario in a given target scenario set S, b represents a bandwidth limit in a given bandwidth limit set B, and $ALoss_s^b$ represents the measured area loss of the scenario s with the bandwidth limit b. The indicator of a target system is omitted in the equation for conciseness. The overall losses of the FB, GM, Jitsi, MST, Skype, and Zoom are 0.41, 0.21, 0.51, 2.28, 0.68, and 2.66, respectively. Overall, GM and Zoom achieve the best and worst ability to handle the insufficient bandwidth, respectively. We expect that the area loss can help researchers and engineers to evaluate the effectiveness of a rate adaptation mechanism adopted in RTC systems.

VI. Conclusion

In this paper, we systematically measure the performance of commercial real-time communication systems. Our study focuses on the visual quality of one-on-one communication. With the carefully designed measurement scheme, we can

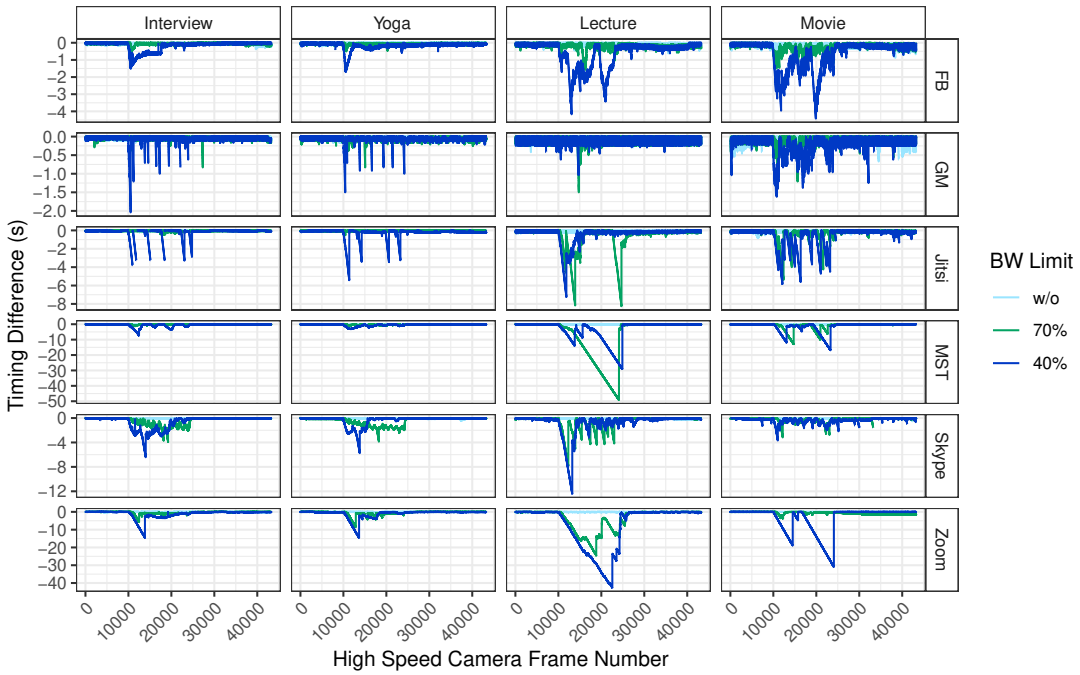


Fig. 9: The timing differences of the target systems and scenarios under three bandwidth limits. The y-axes are rescaled to highlight the difference. Generally, the SS-based scenarios are more vulnerable than the VC-based scenarios under insufficient bandwidth conditions.

measure RTC systems' performance without damaging system integrity. Our measurement scheme suits not only RTC systems but also other video transmission systems. Furthermore, the proposed scheme is easy to extend to multiple users and mobile devices. Besides, we also propose a novel metric, area loss, to measure the system ability to handle insufficient bandwidth for RTC systems. The proposed metric is designed based on two video impairments, the video freezing and the frame delays, which directly deteriorate user experience in real-time communication. We expect our measurement scheme, the new performance metric, and our findings can benefit communities to evaluate the performance of RTC systems to reveal opportunities for future development.

VII. Acknowledgement

The work was partially supported by NSF through grant CNS 1901218.

References

[1] B. García, M. Gallego, F. Gortázar, and A. Bertolino. Understanding and estimating quality of experience in webrtc applications. *Computing*, (101):1585–1607, 2019.

[2] D. Ammar, K. Moor, M. Xie, M. Fiedler, and P. Heegaard. Video qoe killer and performance statistics in webrtc-based video communication. In *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*, pages 429–436, 2016.

[3] S. Fouladi, J. Emmons, E. Orbay, C. Wu, R. Wahby, and K. Winstein. Salsify: Low-latency network video through tighter integration between a video codec and a transport protocol. In *Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation*, NSDI'18, page 267–282. USENIX Association, 2018.

[4] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann. A survey on bitrate adaptation schemes for streaming media over http. *IEEE Communications Surveys Tutorials*, 21(1):562–585, 2019.

[5] J. Fang, M. Ellis, B. Li, S. Liu, Y. Hosseinkashi, M. Revow, A. Sadovnikov, Z. Liu, P. Cheng, S. Ashok, D. Zhao, R. Cutler, Y. Lu, and J. Gehrke. Reinforcement learning for bandwidth estimation and congestion control in real-time communications. In *Workshop on ML for Systems at NeurIPS 2019*, 2019.

[6] S. Baset and H. Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pages 1–11, 2006.

[7] K. Chen, C. Huang, P. Huang, and C. Lei. Quantifying skype user satisfaction. *SIGCOMM Comput. Commun. Rev.*, 36(4):399–410, 2006.

[8] Y. Xu, C. Yu, J. Li, and Y. Liu. Video telephony for end-consumers: Measurement study of google+, ichat, and skype. *IEEE/ACM Transactions on Networking*, 22(3):826–839, 2014.

[9] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo. Analysis and design of the google congestion control for web real-time communication (webrtc). In *Proceedings of the 7th International Conference on Multimedia Systems*, MMSys '16, pages 1–12. Association for Computing Machinery, 2016.

[10] B. Garcia, F. Gortazar, L. Lopez-Fernandez, M. Gallego, and M. Paris. Webrtc testing: Challenges and practical solutions. *IEEE Communications Standards Magazine*, 1(2):36–42, 2017.

[11] B. Jansen, T. Goodwin, V. Gupta, F. Kuipers, and G. Zussman. Performance evaluation of webrtc-based video conferencing. *SIGMETRICS Perform. Eval. Rev.*, 45(3):56–68, March 2018.

[12] D. Vučić and L. Skorin-Kapov. Qoe assessment of mobile multiparty audiovisual telemeetings. *IEEE Access*, 8:107669–107684, 2020.

[13] B. García, F. Gortázar, M. Gallego, and A. Hines. Assessment of qoe for video and audio in webrtc applications using full-reference models. *Electronics*, 9:462, 2020.

[14] C. Hsu, C. Fan, T. Tsai, C. Huang, C. Hsu, and K. Chen. Toward an adaptive screencast platform: Measurement and optimization. *ACM Trans. Multimedia Comput. Commun. Appl.*, 12(5s), 2016.