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

Chih-Fan Hsu

Department of Computer Science, University of California, Davis, One Shields Avenue, Davis, CA 95616
hsuchihfan@gmail.com

Yu-Lun Chang

Individual Researcher
helenafterglow@gmail.com

Chun-Ying Huang

Department of Computer Science, National Yang Ming Chiao Tung University, 1001 University Road, Hsinchu, Taiwan 300, ROC
chuang@cs.nctu.edu.tw

Xin Liu

Department of Computer Science, University of California, Davis, One Shields Avenue, Davis, CA 95616
liu@cs.ucdavis.edu

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

During the outbreak of epidemic diseases, the importance of real-time communication systems (RTC) systems dramatically increases. People use RTC systems for communicating with others, presenting projects, attending online courses, and sharing videos. With different network conditions, applications, and scenarios, how to choose an appropriate system for high-quality RTC is an open question. To the best of our knowledge, there is no general and unified method to comprehensively evaluate the performance of the publicly available RTC systems. In this paper, we systematically evaluate several performances of RTC systems. Our method treats systems as a black box, which can be easily adapted to other systems. Our method is also available for other video transmission systems, such as streaming and live broadcasting systems. According to our measurement method, we evaluate three web-based and three software-based RTC systems on two video conferencing (VC)-based and two screen sharing (SS)-based scenarios. We measure the received video quality (graphical quality and frame rate) at the receiver, the upload bitrate at the sender, and four usages of local resource. Furthermore, we propose a new metric to measure the ability of the system to handle insufficient bandwidth situations. Our proposed metric is the first one directly measure the ability of the rate adaptation mechanism for RTC systems. We expect the measurement method, the metric, and our findings can help system development in the future.

Our detailed analysis reveals that (1) the software-based systems are more efficient than the web-based systems for bitrate usage; (2) the web-based systems are more tolerant to insufficient bandwidth conditions than the software-based systems; (3) the studied RTC systems currently are not designed for the scenario of sharing dynamic videos because of their low-frame-rate strategy, which limits the usage; and (4) decreasing graphical quality is more likely to be recognized than decreasing the frame rate. Frame rate adjustment for rate adaptation can be considered.

Keywords: Real-time communication systems, Performance evaluation, Performance analysis

1. Introduction

Real-time communication (RTC) systems, such as video communication system or webinar system, is one of the popular services in our daily life. People use RTC systems for communicating with others, attending online courses, presenting projects, or sharing videos. No matter in which scenario, RTC systems greatly reduce the distance among people all around the world. Because of the outbreak of epidemic diseases and stay-at-home policy, numerous companies, government facilities, and schools, are temporarily closed and decide to move meetings, presentations, and lectures to publicly available RTC services. To catch up on the boom of requirement, numerous companies successively proposed new RTC services for the public. During this period, the surge of video traffic makes the network more congested [1]. Under this circumstance, the failure rate for delivering videos during conferencing increases, eventually affecting the user experience.

Nowadays, maintaining the user experience of end-users in any circumstance becomes extremely important. To achieve this goal, adapting the transmission rate to utilize the bandwidth is a possible solution to handle the dynamic and congested network. The concept is widely adopted in streaming services [2, 3, 4]. Generally, both RTC service and streaming system are transmitting video from one or multiple video senders (such as content providers) to multiple receivers (end-users). However, rate control in RTC is still in its infant stage [5, 6] and there is no general and unified metric to evaluate the effectiveness of a rate adaptation system.

In this paper, we start with a key question "which service or system I should use?", from the user's perspective, as there are numerous RTC services available to the public and no systematic method to comprehensively compare performances among systems. We design two methods to indirectly measure the performance of RTC systems to avoid performance degradation caused by embedding additional codes or functions into systems. One method aims to measure the transmitted video quality which includes the frame rate and graphical quality; the other method aims to measure the local resource consumption. The carefully designed methods allow us to measure the target statistics of RTC systems as a black-box, which can be adopted to any system related to video transmission, such as streaming system or web broadcasting system, and it is also available for mobile devices. Three sets of performance metrics (eight metrics in total), the video qualities on the receiver side, the upload bitrate usage on the sender side, and the local resource consumption, are compared among the systems. We also propose a novel metric that indicates the system ability to handle insufficient bandwidth conditions, in order to evaluate the effectiveness of the bitrate adaptation mechanisms. The metric is calculated from two key impairment events, video freezing and frame delay, which should be prevented during an online conversation. From the providers' perspective, how to evaluate the quality of adaptation mechanism is also important for system development.

We adopt our measurement method to six publicly available RTC systems with heterogeneous characteristics, including browser-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; four scenarios in two categories of usage scenarios of real-time communication, video conferencing (VC)-based scenarios and screen sharing (SS)-based scenarios; and three bandwidth limits.

We highlight the contribution of this work as follows:

- We systematically and comprehensively evaluate the performances of heterogeneous RTC systems and reveal their advantages and disadvantages.
- We propose a new metric, namely **area loss**, to indicate the ability of RTC systems for handling insufficient bandwidth conditions.
- We conduct a user study to investigate the perceived video quality when watching a video online. The result shows that the frame rate and the value of peak signal-to-noise ratio (PSNR) should be at least 18 FPS and 35.5 dB, respectively.
- We compile several useful suggestions to help users choosing the appropriate systems according to their needs.

2. Related Work

Early-stage of performance measurement for RTC systems mainly focused on VoIP services such as Skype [7, 8, 9]. With 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 the operating systems or platforms. Besides, WebRTC provides built-in APIs to directly collect statistics for researchers. Gradually, the research focus shifted to WebRTC. Carlucci et al. [5] 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 a 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. [11] 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. [12] conducted a detailed evaluation of the WebRTC system. They revealed that (1) the selective forwarding unit (SFU) topology significantly improves the performance of multi-party video call and (2) WebRTC is suffered from poor performance over the wireless network due to bursty packet losses and retransmission. Dunja Vučić and Lea Skorin-Kapov [13] used WebRTC APIs to monitor the session-related data on mobile devices 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 because (1) not all systems provide APIs and (2) the covered statistics provided by different service providers might not be the same, and might not be fair to compare with each other. 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. [14] 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 some performance metrics, 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, for some public available RTC systems, such as Zoom and Skype, embedding codes to the services might be problematic because of the secured source code.

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 are competing for computing resources. The method has been adopted to compare the performance of commercial screencast systems [15] and a few RTC systems [16]. Fouladi et al. [16] 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 merely compared the received graphical quality and the end-to-end delay in a fixed laboratory scenario to demonstrate the performance improvement of the proposed system.

3. Experiment Materials

In this section, we detail the target RTC systems and usage scenarios.

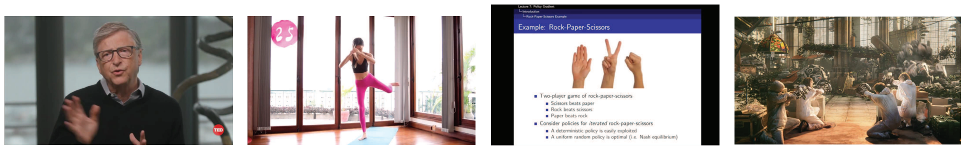
3.1. *Target RTC Systems*

We select six common RTC systems: (1) Facebook Messenger Rooms (FB), Google Meet (GM), Jitsi Meet (Jitsi), Microsoft Teams (MST), Skype, and Zoom.The systems are separated into the web- and software-based systems. The web-based systems use web browsers to attend an RTC session, which includes FB, GM, and Jitsi; the software-based systems use the software to join the session, which includes MST, Skype, and Zoom. We note here that the software-based systems also have web-based versions. We evaluate the performance of the software versions to evaluate the full performance of the services. In our experiment, Google Chrome (v84.0.4147.135 64bit) is served as the web browser for web-based systems. The versions of the MST, Skype, and Zoom are v1.300.21759, v8.63076, and v5.1.3, respectively.

3.2. *Target Scenarios*

We investigate four common usage scenarios of RTC systems, which includes a one-on-one communication, an online exercise class, a lecture presentation, and a movie sharing scenarios. We select four three-minute-long video clips with the resolution of 1280×720 pixels (px) and the frame rate at 24 frames per second (24FPS) to represent the four scenarios, respectively. Figure 1 shows the example frames. We detail the content in each video clip as follows:

- **Online Interview (Interview)**. An online interview scenario represents the one-on-one video communication. Specifically, a user is sitting in front of a screen to communicate to the remote party. The frontal shot captured by the webcam is mostly occupied by the user's face and/or the user's upper body. Verbal and non-verbal languages convey during the conversation, which includes eye directions or hand gestures.
- **Online Class (Yoga)**. A Yoga class represents a remote exercise class. The teacher demonstrates yoga postures. Each posture holds about 30 seconds. A five-second break follows after a posture demonstration. The webcam is set at a certain distance to the trainer to capture her body movements.
- **Lecture Presentation (Lecture)**. An online lecture represents the presentation in a meeting or a webinar session. The video contains a few sentences and figures in two slides. The teacher explains several concepts to students supported by the slides.
- **Movie Sharing (Movie)**. To make points more concrete or to make the audience more engaging, showing video clips during a meeting or a presentation is common. The scenario also represents sharing videos with friends. We select a short film, Tears of Steel, as the shared movie because the video contains



(a) Interview (b) Yoga (c) Lecture (d) Movie

Fig. 1: Four common usage scenarios of RTC systems.

numerous dynamic action and camera movements in contrast to the slides sharing scenario, which is relatively static.

The raw size of each video is about 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 scenarios are 19.8, 17.1, 4.6, 84.2 MB, respectively. The compression rate indicates the dynamic level of each scenario. Overall, Movie and Lecture are the most dynamic and static scenarios, respectively. The Yoga scenario has a better compression rate than the Interview scenario because of the camera distance. 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 that in the Interview scenario.

4. Methodology

We detail our measurement methodology in this section.

4.1. *Target Performance Metrics*

We set the role of each RTC user to either the video sender or the video receiver. In practice, a user can be the sender and the receiver simultaneously, We measure eight critical performance metrics to compare performances among systems.

- **Graphical quality and frame rate (FPS)**. The video quality, which includes graphical quality and frame rate (FPS), at the receiver are critical to an RTC user because the video directly observed by users and the video impairments directly damage the QoE of users [14]. In the video quality, the graphical quality represents the compression artifacts caused by codecs; the frame rate represents video freezing and delays caused by packet losses and jitter.
- **Upload bitrate at the video sender**. We measure the upload bitrate at the video sender to investigate the relationship between the upload bitrate at the video sender and the video quality at the video receiver. We focus on the upload bitrate because the highest video qualities received at all video receivers are bounded by the upload bitrate of the video sender. However, the download bitrate only affects the quality of the receiver.

- **Tolerance for insufficient bandwidth**. From the consumer's perspective, RTC users are looking for a smooth communication. From the service provider's perspective, how to handle a suddenly insufficient bandwidth to maintain the communication quality is highly important. However, the network becomes congested and bandwidth changes over time because of the Internet resource competition. In this case, the system ability to handle the insufficient bandwidth becomes more important because a video freezing [11] or a dramatic change of the frame rate directly interrupt a conversation and result in an unsatisfied experience during the conversation. To the best of our knowledge, there is no general and unified metric to measure the ability.
- **Four local resource usages**. We investigate the CPU, GPU, memory, and power usages of each system. CPU, GPU, and memory usages indicate the lowest resource requirements for participating an RTC session. The power usage indicates the maximum communication time when the device is unplugged.

4.2. *Color Code Embedding*

To trace the frames received at the receiver, we design a set of color code by a quaternary numeral system and vertically embed the code on the left side of the video. Eleven color boxes are used to represent the color code that occupies about 3% area of a video frame (the color box size is 50×50 px). Specifically, the first seven color boxes indicate the frame number, while the rest of the four boxes are constant in reference to the four colors, black, blue, green, and red. They represent the numbers from zero to three, respectively.

4.3. *Experiment Setup*

The four target scenarios, Interview, Yoga, Lecture, and Movie can be further separated into two usage scenarios, video conferencing (VC) and screen sharing (SS)-based scenarios. The Interview and the Yoga scenarios belong to the video conferencing (VC) scenarios. On the other hand, the Lecture and the Movie scenarios are classified as screen sharing (SS)-based scenarios. In the VC-based scenarios, a webcam is used to capture most of the user's frontal face and upper body or the trainer's body movements. However, when real people communicate with each other in a video conference, even if we carefully designed and asked the user to play by our script, their motions, gestures, and voices would have inevitable changes in different recording sessions. The changes result in incomparable performances. To avoid such disturbance, we show a pre-recorded video clip on the second screen for the webcam to capture, as shown in Figure 3(a) and Figure 4(a), to ensure the consistency in recording sessions. On the other hand, the SS-based scenarios directly share with the screen content of the user without such additional screen, as shown in Figure 3(b) and Figure 4(b).

In our experiments, two clients participate in an RTC session. Both Client 1 and Client 2 have cameras and connect to the session via WiFi. We found an

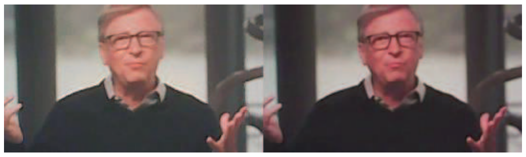


Fig. 2: Example frames of the videos recorded at the sender (left) and receiver (right). We only compare the central area to prevent the negative impact caused by the system user interface.

abnormal high frame rate of the investigated systems except for MST and GM during recordings because of the peer-to-peer connection when Client 1 and Client 2 are connected to the same WiFi router. The peer-to-peer connection is not fair to the systems that connect to service servers. To ensure that the data transmission goes through a service server, a Client 3 located in a different region is added with disabled video and audio. Besides, we force the service server to locate at the same continent where we recorded the experiments to reduce the unnecessary impact during packet transmission.

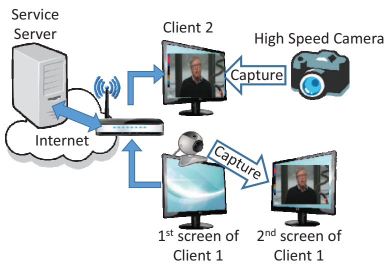
Two recording setups were designed to record (1) the video quality and bitrate and (2) the resource consumptions, respectively. For different setups, the setting and the role of Client 1 and Client 2 are slightly different.

4.4. *The Setup for Video Quality and Bitrate*

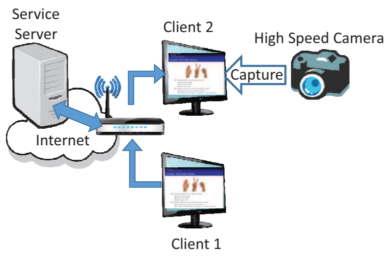
For graphical quality, we firstly record the video displayed at Client 1 in advance as the reference video. Then, we record the received scenes at Client 2 by screen recording software in full-screen mode as the target video. According to the color code, the frames of reference video and the target video are paired to calculate graphical quality. To avoid the unnecessary graphical quality degradation caused by the system user interface, we cropped the center part of the video frame by removing 10% of the width and the height of the periphery area for quality comparison. Figure 2 illustrates an example of the cropped image pair of the Interview scenario. The example is selected from the Jitsi experiment. The left and right images are the examples captured by Client 1 and Client 2, respectively.

For FPS, a high-speed camera is set at Client 2 representing what users observe. As shown in Figure 3, only the camera on Client 1 side is on, as a sender, and Client 2 is playing as a receiver with its camera off. The setting is to prevent any undesirable bandwidth fluctuation due to the bandwidth competition in the wireless network. The bitrate usage is monitored at Client 1 to evaluate the impact of different systems and scenarios. Figure 3(a) and Figure 3(b) shows the setup of the VC-based scenarios and the SS-base scenarios.

We limit the upload bandwidth at the video sender to evaluate the system's ability against insufficient bandwidth situations. Defining a fair bandwidth limit to



(a) VC-based scenarios



(b) SS-based scenarios

Fig. 3: In the setup for recording the received video qualities and the upload bitrate usages, we close the camera at the video receiver to prevent unnecessary bandwidth fluctuation caused by the bandwidth competition in the wireless network. The blue arrows indicate the packet flows transmitted from the sender (Client 1) to the receiver (Client 2).

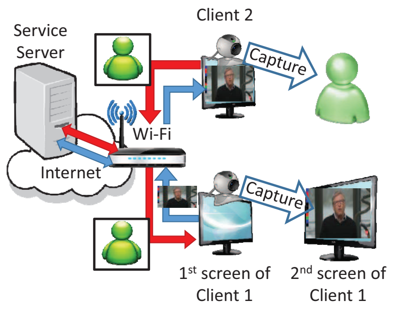
test all systems is difficult because the bandwidth might be seriously insufficient to a certain system but far above the bandwidth requirement to other systems. Hence, we limit the bandwidth by a certain percentage of the ordinary bitrate usage of each system on each scenario to mimic the insufficient bandwidth situations.

4.5. *The Setup for Local Resource Usages*

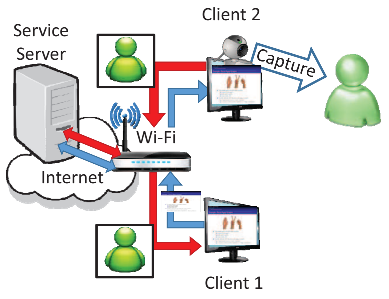
We investigate the CPU, GPU, memory, and power usage of each system. During the recording, we turned on the cameras of Client 1 and Client 2, which means that every user both play sender and receiver in the session simultaneously and it fully drives the consumption of computer resources. We record the resource consumption on Client 1 side. Figure 4 shows the two setups for the VC-based scenarios and the SS-based scenarios. In the VC-based scenarios, a second screen is set to play a video at Client 1 to simulate the scenario. A real user sat in front of the screen at Client 2 to present a person who pays attention to the content transmitted from Client 1. On the other hand, the SS-based scenarios have no such screen.

4.6. *Hardware*

Client 1 (sender) is a Windows 10-64bit laptop with an Intel® Core™ i7-9750H@2.60GHz, 16GB RAM, an Intel® UHD graphics 630 with 8GB memory, and a 15.6-inch Full HD display. The display setting of Client 1 is set to 1440×900 px at 60 FPS. The second screen of Client 1 is a 27-inch display (ASUS VZ279HE) with 1440×900 px at 60 FPS. The webcam adopted on Client 1 is Logitech C170 with 640×480 px capturing resolution at 30 FPS. The Client 2 (receiver) is a macOS Catalina v10.15.5 laptop with an Intel® Core™ i5-8259U@2.30GHz, 16GB RAM, an Intel® Iris Plus graphics 655 with 1,536 MB memory, and a 13.3-inch Retina



(a) VC-based scenarios



(b) SS-based scenarios

Fig. 4: In the setup for recording the local resource usages, CPU, GPU, memory, and power usages, we open the camera at the video receiver to capture a user who pays attention to the content transmitted from Client 1. The blue and red arrows indicate the packet flows transmitted from Client 1 and Client 2, respectively.

display. The display of Client 2 is set to 2880x1800 px at 60 FPS. The webcam of Client 2 is the built-in 720p FaceTime HD camera with 1280×720 px at 30 FPS. The WiFi router is a D-Link DIR-860L IEEE 802.11ac wireless router. A cable modem (Cisco DPC3008) connects the local network and the Internet. The download and upload speeds of the Internet are 100 Mbps (12.5 MBps) and 5 Mbps (0.625 MBps), respectively. An iPhone 8 with iOS 13.6 serves as the high-speed camera with the 1080p at 240 FPS (240.52 or 240.19 in practice) recording settings.

4.7. *Experiment Procedure*

Both experiments can be separated into a preparation stage and a recording stage. The preparation stages of both experiments are the same. In the preparation stage, we first check the setting of the target systems accordingly, such as the sending/receiving resolution (set to maximum), then check the IP address for transmitting and receiving packets. Once the Internet condition is stable, the recording stage is started.

During the recording stage, the video sender starts playing a video after the recording starts at the receiver. During video playing, the targets to be recorded are different according to the experiment as we aforementioned. In the video quality experiment, the videos are recorded at the receiver. The bitrate usage is monitored at the sender. We repeat the video quality experiment three times with different bandwidth conditions to record the impact of the insufficient bandwidth conditions. In the cases of insufficient bandwidth conditions, the video starts with a normal (unlimited) network condition. After 40 seconds, the bandwidth is set to a certain limit for one minute. Afterward, the limit is lifted and an 80-seconds time slot is used for recording the recovery. In the case of the ideal bandwidth condition, the

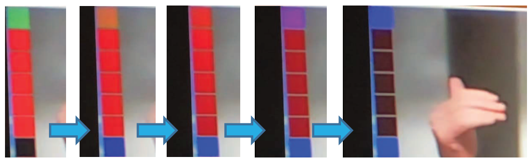


Fig. 5: Two errors caused the invalid frame numbers: (1) the intermediate color error (the orange color at the first color box in the 2^{nd} image), the color is undefined previously; and (2) the color updating error (the blue color at the 7^{th} box in the 2^{nd} image), the color should be changed from black to blue in the 5^{th} image.

network is set to unlimited.

In the local resource usages experiment, the system resource, including CPU, GPU, memory, and power usages, are recorded every 10 seconds at Client 1 to minimize the unnecessary performance impacts.

4.8. *Decode the Video Frame Number*

The video frame number of the received video at time t, F_t, can be decoded from the color code by the nearest color strategy between the captured color and the reference colors. By comparing the camera frame and video frame number, the graphical quality and received frame rate can be obtained.

Decoding color codes inevitably generates invalid video frame numbers, even though the reference colors are set. Two possible causes are listed as follow:

- **Intermediate color error**. Since the capturing timing of the high-speed camera is not precisely aligned with the fresh rate of the screen, the camera sometimes captures the moment of the color transition. In this case, an intermediate color is captured instead of black, blue, green, and red. For example, as we can see in the second frame of Figure 5, the first color box is captured orange when it changes from green to red.
- **Color updating error**. The 2^{nd} to 7^{th} color boxes in Figure 5 shows that they do not update simultaneously. The 7^{th} color box updates from black to blue at the 2^{nd} frame but the color transition should be started in the 4^{th} frame. The early update causes an invalid video frame number. We suspect that it is the temporal distortion caused by the system codec.

Fixing the invalid Frames. A preprocessing algorithm is proposed to handle the invalid frame numbers. For the intermediate color error, assigning each intermediate color to a predefined number is impractical because the intermediate colors vary according to the captured timing. After carefully investigating the decoded sequence of video frames, we found that the intermediate color errors usually appear for a short time (within two consecutive camera frames) followed by the video frame number shown for a relatively long time. Therefore, we fix the error by replacing the frame number with the followed frame number because the color should only be changed when transiting to the next frame. Specifically, let F_t indicate the decoded video frame number at time t. Once F_t differs from F_{t-1}, we check the followed two video frame numbers F_{t+1} and F_{t+2}. If F_t, F_{t+1} and F_{t+2} are all the same, it appears no intermediate color error, and F_t stays. Otherwise, we replace F_t by F_{t+2}. The process iterates until no frame numbers changed.

For the color updating error, disordered video frame numbers are observed. The timing of the early update varies and the error usually occurs several consecutive video frame numbers, which increase the difficulty to decide a proper frame number to fix the error. However, the color updating error can be easily detected by the ideal sequence of the video frame number. According to the frame rates of the selected video (24 FPS) and the high-speed camera (240.52 or 240.19 FPS), a video frame number will be shown about ten consecutive camera frames. In this case, we can generate an ideal sequence of the video frame numbers. We then align the ideal and decoded sequences of video frame numbers by (1) the smallest non-zero decoded video frame number and (2) the sequence slop of the first 30 seconds in the decoded video frames (the ideal bandwidth period). By following the facts, (1) *the timing of each video frame number shown on the sender side is coherently displayed* and (2) *a video frame number cannot be shown before the timing it should be displayed.* Once a decoded frame number is larger than the corresponding ideal number. It is an invalid frame number caused by the color updating error and can be simply ignored.

Overall, an average of 2.20% (0.59%) of frame numbers are fixed due to the intermediate color error and 2.55% (2.53%) of frame numbers are ignored due to the color updating error across all scenarios. The values in the parentheses are the standard deviation.

5. Experiment Results

In this section, we report the experiment result and comprehensively analyze the results.

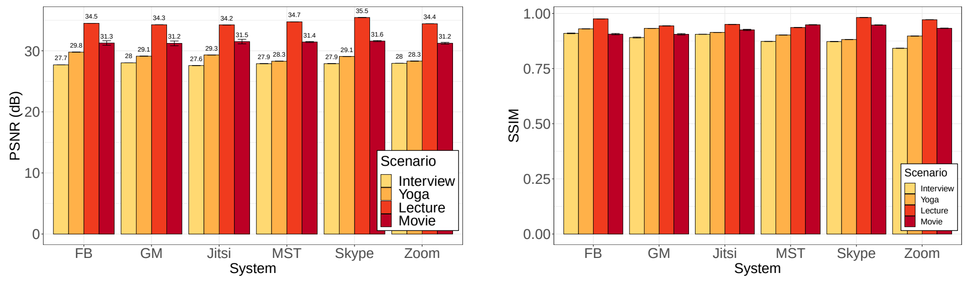
5.1. *Received Video Quality*

Frame Rate. Table 1 shows the frame rates (FPS) of received videos. Overall, the FPS in the VC-based scenarios is usually higher than the FPS in the SS-based scenarios. However, there is no significant difference between the scenarios in the VC-based and SS-based scenarios on both web- and software-based systems. We conclude that the received frame rates of the target systems are not depending on the video contents. In the VC-based scenarios, the FPS of the target systems is usually 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 for transmitting a dynamic movie when sharing screens to others, especially for web-based systems. However, in the situation that lectures and meetings are held online, sharing a short video clip during a presentation becomes common. The current system especially the web-based systems might not satisfy the surge of the requirement. As we can observe that Zoom achieves relatively higher FPS than other systems in the Movie scenario. It is because Zoom introduces an additional mode for sharing dynamic videos. Specifically, Zoom intends to lift the implicit bitrate restriction when sharing a dynamic video to maintain higher FPS for a better user experience. However, this strategy also brings a significant increase in bitrate usage (as shown in Fig. 7). The result reveals that there is still room to be improved on FPS for the SS-based scenarios. **Overall, the received frame rate only depends on the usage scenario and the system implementation.**

Table 1: 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 calculated, Peak-signal-to-noise ratio (PSNR) and Structural Similarity Index (SSIM), to estimate the degradation of the graphical quality due to the codecs, as reported in Figure 6. The intervals in the figure are the 95% confidence intervals. We mention here that the frames of the videos recorded at the sender and the receiver are paired by the decoded video frame numbers. For each video frame number, we only select one image from both videos, respectively, for comparison. 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 screen sharing is used to share slides or screen contents with the remote participants. Hence, providing high graphical quality for the transmitted contents should be the basic requirement. Between the two SS-based scenarios, the Lecture scenario achieves higher graphical quality than the Movie scenario. It is because the Lecture scenario is highly monotonic and static. **Overall, the graphical quality depends on the usage scenario and the video content.**



(a) Peak-Signal-to-Noise Ratio (b) Structural Similarity Index

Fig. 6: The graphical qualities, PSNR and SSIM, of the target scenarios and systems. All systems have similar graphical qualities. The intervals are the 95% confidence intervals.

5.2. *Upload Bitrate*

Figure 7 reports the average upload bitrate in the target scenarios of each system. The difference between the two categories of the target systems is obvious. Although the frame rates and graphical quality of the web-based systems are slightly lower and similar to the software-based systems, respectively, the web-based systems consume more bitrate than software-based systems in the VC-based scenarios. The result reveals that the web-based systems are generally less efficient than the 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 video conferencing mode are identical and limited by the systems. It is not intuitive because we expect the bitrate will be varied based on the transmitted contents to maintain a stable perceived video quality because the video quality is directly observed by users. Therefore, it is still room for further improvement.

On the other hand, for SS-based scenarios, web-based systems achieve a relatively lower bitrate than the software-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. This is intuitive because a dynamic video usually uses more bitrate than the static one as they have similar graphical qualities and frame rates after temporal compression. The bitrate usage of the Movie scenario of Zoom is higher than the other software-based systems. As we previously mentioned, it is caused by the additional mode for sharing a dynamic video. The mode aggressively consumes the bitrate for achieving a higher frame rate

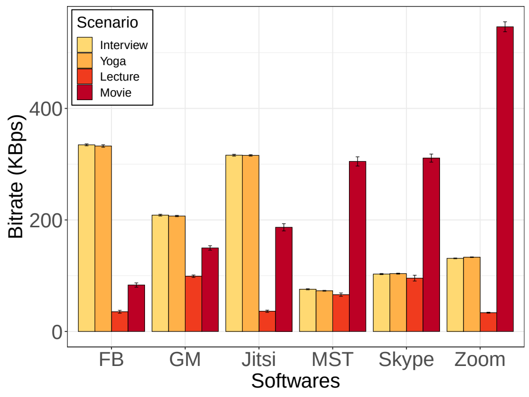


Fig. 7: The average bitrate usage for each scenario of the investigated systems. The bitrate of the VC scenarios is highly similar in each system.

without decreasing the graphical quality.

Overall, the bitrate is restricted in the VC-based scenarios. The bitrate usage depends on the transmitted contents in the screen sharing scenarios. The software-based systems are more efficient than the web-based systems on the upload bitrate usage.

5.3. *Graphical Quality, Frame Rate, and Quality of Experience*

To provide an acceptable user experience to RTC users, service providers should maintain the quality of the service for their clients. Considering the perceived frame rate and the graphical quality are directly impact the QoE of users. We conducted a user study on Amazon Mechanical Turk to investigate the relationship among the satisfaction of users, the perceived frame rate, and the perceived graphical quality when they watch a video online. We randomly select 30 video clips (ten seconds each) from Tears of Steel. Each video is compressed with a random bitrate (ranged from 200 to 2,000 kbps), a random frame rate (ranged from 5 to 30 FPS), and a random video resolution (random from 1920×1080, 1706×960, 1494×840, 1280×720, 1066×600, 854×480) by FFmpeg. The level of graphical quality depends on the combination of the bitrate, the frame rate, and the video resolution. Forty subjects (26 males and 14 females) participated in our user study. Their ages range from 18 to 56 and the average value (standard deviation) is 28.4 (7.4). Subjects were asked to carefully observe a random video in an experiment round. After observation, subjects are asked to score their opinion for the observed video from one (terrible) to seven (best). Each experiment session contains 20 rounds. We only take completed experiment sessions into account.

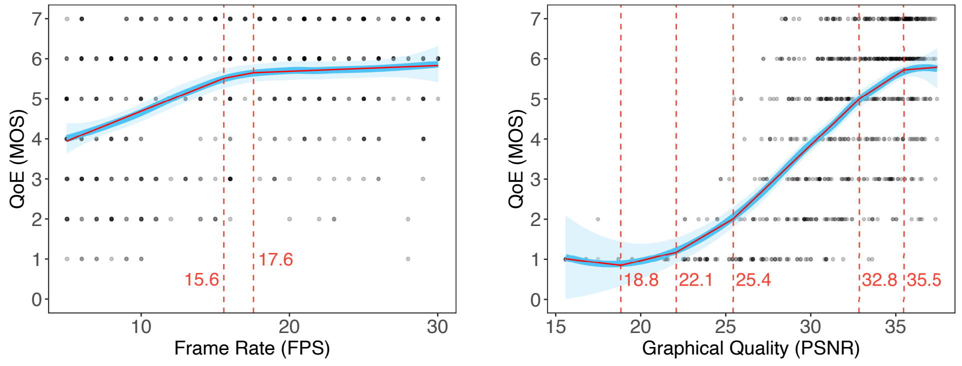
We firstly assume that the acceptable value of PSNR value is 30 (slightly lower than the average PSNR values in the Movie scenario). We select the experiment results in which the PSNR values of the observed videos are larger than and equal to 30. Figure 8(a) shows the relationship between the users' opinions and frame rates. We smooth the result by LOWESS smoother [17] (blue line). The light blue area is the confidence interval of LOWESS. we detect the change points of the smoothed curve by fitting with line segments (red segments in the blue line) to monitor the QoE score changes. The change points are highlighted with red dashed lines. The result shows that the QoE scores decrease as the frame rate is smaller than 17.6 (the largest change point). The result indicates that the frame rate of the video should at least be 18, otherwise the users will notice the frame rate drop and finally damage the user experiences.

We then investigate the relationship between the QoE scores and the graphical quality (PSNR) by selecting the experiment rounds with the frame rate of the observed video larger than 17. We redo the same process as we have done for the frame rate. The result is shown in Figure 8(b). We can observe that the QoE scores dramatically decrease as the PSNR values that are smaller than 35.5. The result indicates that users are likely to recognize the image impairment caused by the video compression if the average PSNR value of a video is smaller than 36. Between the frame rate and the graphical quality, we can observe that the range of the graphical quality drop is more severe than the range of frame rate drop.

Overall, subjects are more sensitive to the degradation of the graphical quality than the decrease of the frame rate. Besides, the frame rate and the graphical quality on the receiver side should at least be 18 and 36, respectively, to maintain a high-quality viewing experience for the user. We link the results of the user study to the design consideration of RTC systems. To satisfy the bitrate fluctuation and maintain the user experience, reducing the frame rate is less likely to be perceived by users. We also suggest that the rate adaptation mechanism can be considered for the VC-based scenarios to prevent the unstable graphical quality for the consumers. The result of the user study also strengthens the previous conclusion: the investigated RTC systems especially the web-based systems might not satisfy the usage scenario for sharing a dynamic video.

5.4. *Ability for Handling Insufficient Bandwidth*

Under the insufficient bandwidth condition, we expect that an RTC system will decrease the video quality to reduce the bitrate usage to fulfill the bandwidth condition and the video freezing should be avoided because it greatly damages the quality of a conversation. Hence, between the video qualities, we focus on the received frame rate because the video freezing and the frame delay directly affect the frame rate. To measure the ability of the tolerance for insufficient bandwidth, which is highly related to the implementation of the system, three bandwidth limits are investigated in our experiment, without limit, 70% limit, and 40% limit. Note that the upload bandwidth at the video sender is limited from 40^{th} to 100^{th} second. Figure 9 reports the percentage of received frames at the receiver among three bandwidth limits. As



(a) QoE vs. FPS (b) QoE vs. PSNR

Fig. 8: The user-perceived qualities under different frame rates and graphical qualities. We smooth the result by the LOWESS smoother (blue line) and fit the smoothed results by a set of line segments (red lines). Then, we obtain the change points (red dashed lines) by the line segments.

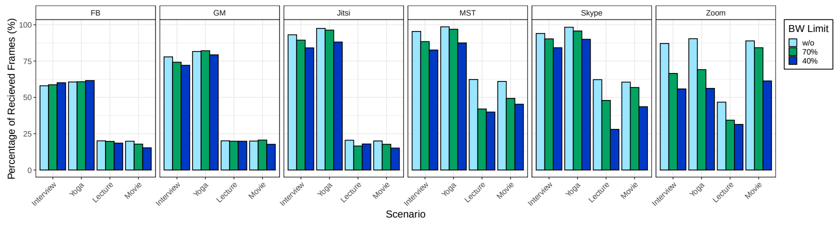


Fig. 9: The percentage of received frames under the bandwidth limits, without limit, 70% limit, and 40% limit. Generally, the received frame rate decreases when insufficient bandwidth.

we expect, the values significantly drop under insufficient bandwidth conditions in most of the target systems, except FB and GM. Besides, the bandwidth variation has less impact on the web-based systems than the software-based systems.

The drop in frame rate might be caused by the frame rate adjustment of the rate adaptation mechanism of the system or by the frame freezing and frame delay. Table 2 shows the number of video freezing where the duration is longer than 1 second. Without any bandwidth limitation, it appears that all investigated systems are free of video freezing in VC- and SS-based scenarios. In the VC-based scenarios, under the 70% bandwidth limit, FB, GM, Jitsi, and MST remain no video freezing. In the four systems, GM achieves almost no video freezing under all bandwidth limits. Generally, the SS-based scenario is more vulnerable than VC-based scenarios under insufficient bandwidth conditions.

Table 2: The Number of Video Freezing under the Bandwidth Limits (100%/70%/40%). An RTC system that has a reliable rate adaptation mechanism or model should prevent video freezing as the insufficient bandwidth conditions.

	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

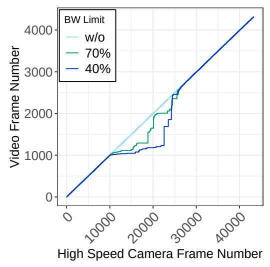
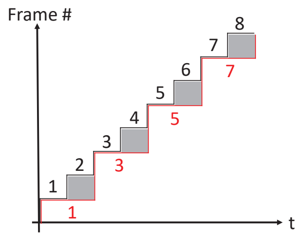


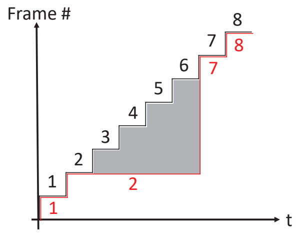
Fig. 10: The received frame numbers under three bandwidth limits of the Lecture scenario of Zoom. Horizontal and vertical lines indicate the events of video freezing and skip frame, respectively. The frame delay can also be observed.

We further look into the sequence of video frame numbers at the receiver. Figure 10 shows an example of the Lecture scenario in Zoom to demonstrate the relationship between received frame numbers and their displayed timing. The display timing can be indicated from the frame number of the video captured by the high-speed camera. In this figure, video freezing and frame delay can be observed. A horizontal pattern indicates that the video freezing on a certain frame number and is captured by the high-speed camera for a while. The more bandwidth is limited, the more the received video frame sequence deviates from the frame sequence without bandwidth limitation. During the frame freezing, the system would try to adapt the rate of the video to the current bandwidth, by decreasing the frame rate or the graphical quality. If the freezing time is too long, the system will drop the late frames to catch up the recent frames. In this case, a vertical pattern can be observed. Then, the received frame number sequence gradually gets closer to the sequence without bandwidth limitation.

As we mentioned before, there is an ideal frame sequence according to the frame rate of the selected video and the high-speed camera. By introducing the ideal sequence into the sequence comparison, the difference between the ideal sequence and



(a) FPS Adjustment



(b) Video freezing

Fig. 11: The new metric, area loss, is designed to indicate the level of video freezing and the frame delay caused by insufficient bandwidth. The metric is less sensitive to the frame rate drops caused by the frame rate adjustment of systems.

the sequences under the three bandwidth limits can be calculated. 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 system.

To further explore the area loss. We look into the impact caused by the frame rate changing to the area loss. Figure 11 shows examples of a 50% frame rate degradation caused by the frame rate adjustment (Figure 11(a)) and the video freezing (Figure 11(b). The step curves in the figure indicate the captured video frame number F_t at time t. The black and the red step curves represent the ideal and the received sequences, respectively. Figure 11(a) shows that the system regularly skips one frame in two frames to satisfy the network bandwidth. Figure 11(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 12 shows the percentage of the loss areas of the three bandwidth conditions. The number of loss percentages above 1% is marked above each bar. 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. (2) The web-based systems generally achieve better insufficient bandwidth tolerance 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 transmitted video of the Movie scenario is much more dynamic than the

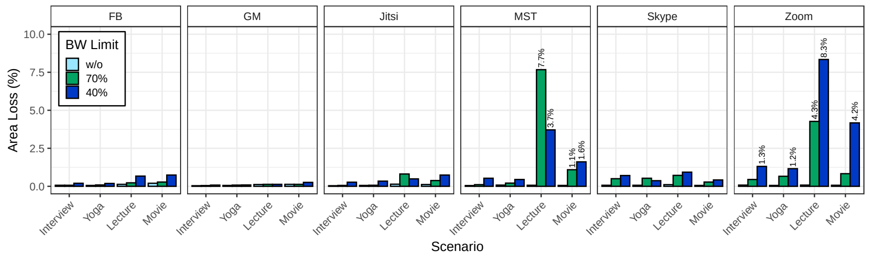


Fig. 12: The area loss in each target system and scenario. The loss is calculated from the sequence of the decoded video frame numbers and the corresponding ideal sequence.

Lecture scenario. We suspected that it is caused by the low-bitrate advantage that is benefited by the static and monotonic content. The system can greatly reduce the video bitrate because of temporal consistency. The instant bandwidth drop causes packet losses and results in a retransmission of the I-frame which is the reference frame for temporal encoding. However, the limited bandwidth is extremely low because its ordinary bandwidth is very low originally. In this case, the extremely low bandwidth limitation cannot afford the retransmission of the I-frame and eventually causes the video freezing and the frame delay.

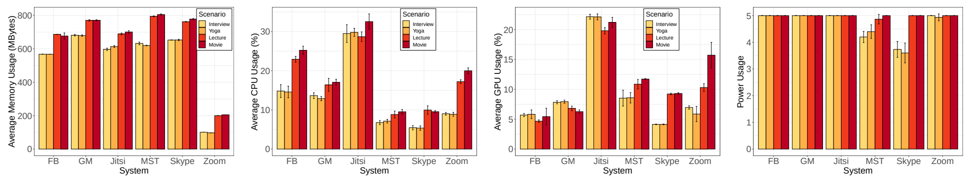
We calculate an overall loss based on the area loss to directly compare the system's ability to handle the insufficient bandwidth. The intuitions of the overall loss follow: (1) we expected the system adapts the transmission rate to handle the insufficient bandwidth conditions, and (2) we can slightly 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.

5.5. *Local Resource Usages*

We report the local resource usages of the target systems without the bandwidth limit. All resource usages are recorded at Client 1.



(a) Memory usage (b) CPU usage (c) GPU usage (d) Power usage

Fig. 13: The local resource usage of the target scenarios and systems.

- **Memory usages**. Figure 13(a) reports the memory usages for each experiment setting. The memory usage of Zoom is much lower than other systems. We suspect this advantage is caused by the service implementation. Generally, the SS-based scenarios use more memory than the VC-based scenario. The result is intuitive because the resolution of the desktop transmitted in the SS-based scenarios is usually higher than the camera resolution in the VC-based scenarios.
- **CPU/GPU usages**. The CPU and GPU usages are highly related to the codec adopted in the system implementation. Figure 13(b) and Figure 13(c) report the average CPU and GPU usages, respectively. In CPU usage, the usage of the web-based systems is generally larger than the software-based systems. It is because the system should locate CPU resources to the web browser. Among the two usage scenarios, the CPU usages for SS-based scenarios are generally higher than VC-based scenarios. The result is also as we expected because the codec should use more CPU resources to compress a video with higher resolution. Overall, Skype and Jitsi achieve the lowest and the highest CPU usage.

  In the GPU usage, there is no significant difference whether it's VC- or SS-based scenarios except Zoom. It is the additional mode of Zoom that aggressively uses the GPU resources for transmitting more frames to the receiver in a short time. In the web-based systems, there is no significant difference in GPU usage between the video conferencing scenarios and screen sharing scenarios. It is because the three web-based systems are based on WebRTC, and WebRTC still does not support GPU acceleration when we conducted our experiment [18].
- **Power usages**. The power usage in Figure 13(d) is also recorded from the task manager of Windows OS. The value is highly related to the combination of the CPU and GPU usages. Five digits, one to five, represent "very low", "low", "moderate", "high", and "very high", respectively. We can observe that VC-based scenarios consume less power than the SS-based scenarios in MST, Skype, and Zoom. Between the target systems, Skype consumes less power than other systems.

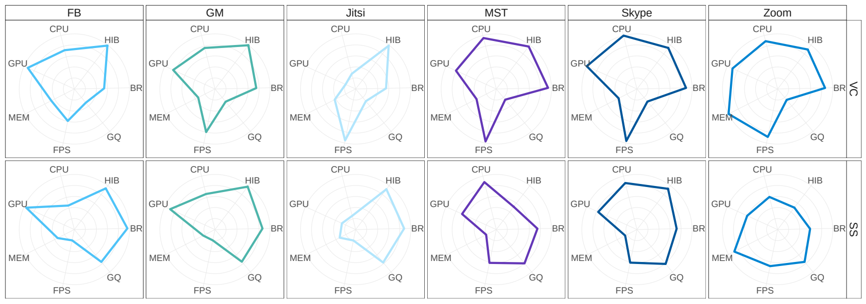


Fig. 14: The overall performance of the target systems in the video conferencing (VC) and the screen sharing (SS)-based scenarios. The TIB, BR, GQ, POW, and MEM represent the Tolerance for Insufficient Bitrate, BitRate, Graphical Quality, POWer, and MEMory, respectively. The values are normalized and the larger values represent the better performance.

5.6. *Overall Performance*

Overall, we measure eight performance metrics for each system. In the performance metrics, the large values of the graphical quality (GQ) and the frame rate (FPS) are preferred. Otherwise, the small values are preferred, which includes the bitrate (BR), the tolerance for insufficient bandwidth (TIB), memory (MEM), power (POW), GPU, and CPU. We rescale each measured performance metric from one to five by the corresponding maximum and minimum value among the target systems and scenarios. Except for FPS, we set the maximum value to 24 according to the suggested frame rate for a smooth video. For each small-value-preferred metric, we flip the rescaled value to change the metric to the large-value-preferred metric. Finally, we average the values in the VC- and SS-based scenarios for each system.

Figure 14 shows the radar chart of the final results for the VC- and SS-based scenarios of each system. As we can observe the target systems have their advantages and disadvantages. Generally, VC- and SS- based scenarios prefer to maintain the FPS and GQ, respectively. In the VC-based scenarios, Skype achieves the best but the performances of the three software-based systems are similar. In the SS-based scenarios, Skype can also be considered. In the case that users are not allowed to install software for online communication, GM can be considered. For users who take security issues seriously, Jitsi can be considered because users can set up a private service server for online communication but users need to take care of local resource consumption especially the CPU and GPU usages.

6. Conclusions and Future Work

In this paper, we systematically and comprehensively measure the performance of commercial real-time communication systems. With the carefully designed measurement methods, we can measure RTC systems' performance without damaging system integrity. Our measurement methods suit not only RTC systems but also other video transmission systems. Furthermore, the proposed measurement method is easy to extend to multiple users and mobile devices. We further propose a novel metric to measure the ability to handle insufficient bandwidth situations for RTC systems. The proposed metric is designed based on the sequence of the received video frame, which can point out two critical visual video impairments, the video freezing and the frame delays. The impairments directly deteriorate user experience.

We adopt our measurement method to measure six RTC systems on four common usage scenarios. Our study reveals that (1) there is no single superior RTC system in the target systems; (2) the graphical qualities of all target systems are similar; (3) in the video conferencing-based scenarios, the transmitted video content has no impact on the upload bitrate usages at the video sender and the received frame rate. To maintain a stable graphical quality to users, adapting the frame rate according to video contents can be considered; (4) the web-based RTC systems are not designed for sharing a dynamic video to users with screen-sharing mode. It limits the usage scenario of web-based RTC systems. An aggressive bitrate usage strategy and a GPU acceleration method can be considered to increase the transmitted frame rate; (5) users are less likely to recognize the image degradation caused by the codec when the frame rate and the PSNR value are larger than 17.6 FPS and 35.5 dB, respectively; (6) the software-based systems achieve higher bitrate efficiency than the web-based systems. Users with relatively low Internet bandwidth can consider the software-based systems; and (7) the web-based systems generally achieve a higher ability to handle the insufficient bandwidth. Users who want to avoid video freezing can consider adopting the web-based systems.

Our study mainly focuses on the visual quality of one-on-one communication. The auditory quality and end-to-end latency are excluded. The end-to-end latency is also vital to RTC systems. ITU-T G.114 suggested that the end-to-end latency should be less than 150 ms [19]. Generally, the end-to-end latency can be separated into a basic delay caused by packet traveling and an incremental delay caused by insufficient bandwidth. In this paper, we cover the dramatic changes in incremental delay, which inevitably cause video freezing. The commercial system's basic delay differs depending on service servers' geolocation, the routing strategy, and service topology. Decompressing the basic delay is also an exciting topic. Studying the relationship between the area loss and subjective quality is an other interesting topic. As we know that frame freezing has a severe impact on QoE. However, different types of video freezing can have the same area loss value but could have very different QoE values. In this case, investigating the correlation between the area loss and subjective QoE of the RTC system should introduce additional performance metrics, such as the number of video freezing, graphical quality, and the freezing duration. The experiment should be carefully designed. We hope our measurement method can be a start to design a fair and identical performance measurement framework to evaluate all kinds of video transmission systems.

References

[1] C. Kang, D. Alba, and A. Satariano. Surging traffic is slowing down our internet. *The New York Times, Online article, https://www.nytimes.com/2020/03/26/business/coronavirus-internet-traffic-speed.html*, 2020.

[2] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over http. *SIGCOMM Comput. Commun. Rev.*, 45(4):325–338, 2015.

[3] H. Mao, R. Netravali, and M. Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, page 197–210. Association for Computing Machinery, 2017.

[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] 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. Association for Computing Machinery, 2016.

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

[7] S. A. Baset and H. G. 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.

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

[9] 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.

[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] D. Ammar, K. De 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.

[12] 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, 2018.

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

[14] 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.

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

[16] S. Fouladi, J. Emmons, E. Orbay, C. Wu, R. S. 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.

[17] W. S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368):829–836, 1979.

[18] Google Cloud Solutions. Gpu-accelerated streaming using webrtc. *Online article, https://cloud.google.com/solutions/gpu-accelerated-streaming-using-webrtc*, 2020.

[19] International Telecommunication Union. ITU-T Recommendation G.114, 2003.