Can You See Me Now? A Measurement Study of Zoom, Webex, and Meet

Hyunseok Chang
Nokia Bell Labs
Murray Hill, NJ, USA

Matteo Varvello
Nokia Bell Labs
Murray Hill, NJ, USA

Fang Hao
Nokia Bell Labs
Murray Hill, NJ, USA

Sarit Mukherjee
Nokia Bell Labs
Murray Hill, NJ, USA

ABSTRACT

Since the outbreak of the COVID-19 pandemic, videoconferencing has become the default mode of communication in our daily lives at homes, workplaces and schools, and it is likely to remain an important part of our lives in the post-pandemic world. Despite its significance, there has not been any systematic study characterizing the user-perceived performance of existing videoconferencing systems other than anecdotal reports. In this paper, we present a detailed measurement study that compares three major videoconferencing systems: Zoom, Webex and Google Meet. Our study is based on 48 hours' worth of more than 700 videoconferencing sessions, which were created with a mix of emulated videoconferencing clients deployed in the cloud, as well as real mobile devices running from a residential network. We find that the existing videoconferencing systems vary in terms of geographic scope, which in turns determines streaming lag experienced by users. We also observe that streaming rate can change under different conditions (e.g., number of users in a session, mobile device status, etc), which affects user-perceived streaming quality. Beyond these findings, our measurement methodology can enable reproducible benchmark analysis for any types of comparative or longitudinal study on available videoconferencing systems.

CCS CONCEPTS

• **Networks** → **Network measurement**; *Cloud computing*; • **Information systems** → *Collaborative and social computing systems and tools*.

ACM Reference Format:
Hyunseok Chang, Matteo Varvello, Fang Hao, and Sarit Mukherjee. 2021. Can You See Me Now? A Measurement Study of Zoom, Webex, and Meet. In *ACM Internet Measurement Conference (IMC '21), November 2–4, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3487552.3487847

IMC '21, November 2–4, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-9129-0/21/11…$15.00
https://doi.org/10.1145/3487552.3487847

1 INTRODUCTION

There is no doubt that the outbreak of the COVID-19 pandemic has fundamentally changed our daily lives. Especially with everyone expected to practice physical distancing to stop the spread of the pandemic, various online communication tools have substituted virtually all sorts of in-person interactions. As the closest form of live face-to-face communication in a pre-pandemic world, videoconferencing has practically become the default mode of communication (e.g., an order-of-magnitude increase in videoconferencing traffic at the height of the pandemic [23, 28, 37]). Thanks to its effectiveness and reliability, video communication is likely to remain an important part of our lives even in the post-pandemic world [27, 32].

Despite the critical role played by existing videoconferencing systems in our day-to-day communication, there has not been any systematic study on quantifying their performance and Quality of Experience (QoE). There is no shortage of anecdotal reports and discussions in terms of the usability, video quality, security, and client resource usage of individual systems. To the best of our knowledge, however, no scientific paper has yet investigated the topic thoroughly with a sound measurement methodology that is applicable across multiple available systems. Our main contribution in this paper addresses this shortcoming.

In this paper, we shed some light on the existing videoconferencing ecosystems by characterizing their infrastructures as well as their performance from a user's QoE perspective. To this end, we have devised a measurement methodology which allows us to perform controlled and reproducible benchmarking of videoconferencing systems by leveraging a mix of emulated videoconferencing clients deployed in the cloud, as well as real mobile devices running from a residential network. We provide the detailed description of our methodology as well as the open-source tools we used (Sections 3 and 4), so that anyone can replicate our testbed to repeat or further extend our benchmark scenarios. Driven by our methodology, we investigate three popular videoconferencing systems on the market: Zoom, Webex and Google Meet (Meet for short). Each of these platforms provides a free-tier plan as well as paid subscriptions, but we focus on their *free-tier plans* in our evaluation. Given these three systems, we conduct measurement experiments which take a combined total of 48 videoconferencing hours over more than 700 sessions, with 200 VM hours rented from 12 geographic locations and 18 hours of two Android phones hooked up at one location. Our findings include:

Finding-1. In the US, typical streaming lag experienced by users is 20–50 ms for Zoom, 10–70 ms for Webex, and 40–70 ms for Meet. This lag largely reflects the geographic separation of users (e.g., US-east vs. US-west). In case of Webex, all sessions created in the US appear to be relayed via its infrastructure in US-east. This causes the sessions among users in US-west to be subject to artificial detour, inflating their streaming lag.

Finding-2. Zoom and Webex are characterized by a US-based infrastructure. It follows that sessions created in Europe experience higher lag than those created in the US (90–150 ms for Zoom, and 75–90 ms for Webex). On the other hand, the sessions created in Europe on Meet exhibit smaller lag (30–40 ms) due to its cross-continental presence including Europe.

Finding-3. All three systems appear to optimize their streaming for low-motion videos (e.g., a single-person view with a stationary background). Thus high-motion video feeds (e.g., dynamic scenes in outdoor environments) experience non-negligible QoE degradation compared to typical low-motion video streaming.

Finding-4. Given the same camera resolution, Webex sessions exhibit the highest traffic rate for multi-user sessions. Meet exhibits the most dynamic rate changes across different sessions, while Webex maintains virtually constant rate across sessions.

Finding-5. Videoconferencing is an expensive task for mobile devices, requiring at least 2–3 full cores to work properly. Meet is the most bandwidth-hungry client, consuming up to one GB per hour, compared to Zoom's gallery view that only requires 175 MB per hour. We estimate that one hour's videoconferencing can drain up to 40% of a low-end phone's battery, which can be reduced to about 20–30% by turning off the onboard camera/screen and relying only on audio. All videoconferencing clients scale well with the number of call participants, thanks to their UI which only displays a maximum of four users at a time.

The rest of the paper is organized as follows. We start by introducing related works in Section 2. We then present our measurement methodology in Section 3, and describe the measurement experiments and our findings in detail in Sections 4–5. We conclude in Section 6 by discussing several research issues.

2 RELATED WORK

Despite the prevalence of commercial videoconferencing systems [36], no previous work has directly compared them with respect to their infrastructures and end-user QoE, which is the main objective of this paper. The recent works by [34] and [29] investigate the network utilization and bandwidth sharing behavior of existing commercial videoconferencing systems based on controlled network conditions and client settings. Several works propose generic solutions to improve videoconferencing. For example, Dejavu [25] offers up to 30% bandwidth reduction, with no impact on QoE, by leveraging the fact that recurring videoconferencing sessions have lots of similar content, which can be cached and re-used across sessions. Salsify [24] relies on tight integration between a video codec and a network transport protocol to dynamically adjust video encodings to changing network conditions.

Videoconferencing system	Low quality	High quality
Zoom [7]	600 Kbps	
Webex [8]	500 Kbps	2.5 Mbps
Meet [14]	1 Mbps	2.6 Mbps

Table 1: Minimum bandwidth requirements for one-on-one calls.

As a consequence of the COVID-19 pandemic, the research community has paid more attention to the impact of videoconferencing systems on the quality of education [21, 33, 40]. As educational studies, these works rely on usability analysis and student surveys. In contrast, our work characterizes QoE performance of the videoconferencing systems using purely objective metrics.

Videoconferencing operators do not provide much information about their system, e.g., footprint and encoding strategies. One common information reported by each operator is the minimum bandwidth requirements for one-on-one calls (Table 1). The results from our study are not only consistent with these requirements, but also cover more general scenarios such as multi-party sessions.

3 BENCHMARKING DESIGN

In this section, we describe the benchmarking tool we have designed to study existing commercial videoconferencing systems. We highlight key design goals for the tool first, followed by associated challenges, and then describe how we tackle the challenges in our design.

A videoconferencing system is meant to be used by end-users in mobile or desktop environments that are equipped with a camera and a microphone. When we set out to design our benchmarking tool for such systems, we identify the following design goals.

(D1) Platform compliance: We want to run our benchmark tests using *unmodified* official videoconferencing clients with *full audiovisual capabilities*, so that we do not introduce any artifact in our evaluation that would be caused by client-side incompatibility or deficiency.

(D2) Geo-distributed deployment: To evaluate web-scale videoconferencing services in realistic settings, we want to collect data from *geographically-distributed* clients.

(D3) Reproducibility: We want to leverage *a controlled, reproducible client-side environment*, so that we can compare available videoconferencing systems side-by-side.

(D4) Unified evaluation metrics: We want to evaluate different videoconferencing platforms based on *unified metrics* that are applicable across all the platforms.

It turns out that designing a benchmarking tool that meets all the stated goals is challenging because some of these goals

are in fact conflicting. For example, while geographically-dispersed public clouds can provide distributed vantage point environments (**D2**), cloud deployments will not be equipped with necessary sensory hardware (~~**D1**~~). Crowd-sourced end-users can feed necessary audiovisual data into the videoconferencing systems (**D1**), but benchmarking the systems based on unpredictable human behavior and noisy sensory data will not give us objective and reproducible comparison results (~~**D3**~~). On the other hand, some goals such as **(D3)** and **(D4)** go hand in hand. Unified evaluation metrics alone are not sufficient for comparative analysis if reproducibility of client-side environments is not guaranteed. At the same time, reproducibility would not help much without platform-agnostic evaluation metrics.

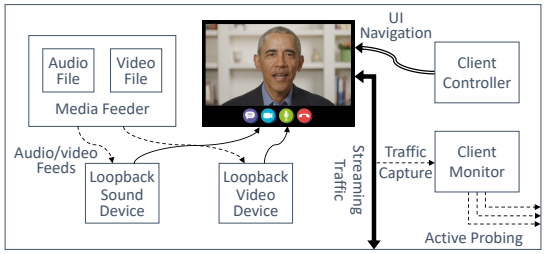


Figure 1: Cloud VM in a fully-emulated environment.

3.1 Design Approach

Faced with the aforementioned design goals and challenges, we come up with a videoconferencing benchmark tool that is driven by three main ideas: (i) client emulation, (ii) coordinated client deployments, and (iii) platform-agnostic data collection.

Client emulation. One way to circumvent the requirement for sensory devices in videoconferencing clients is to *emulate* them. Device emulation also means that the sensory input to a videoconferencing client would be completely under our control, which is essential to reproducible and automated benchmarking. To this end, we leverage *loopback pseudo devices* for audio/video input/output. In Linux, `snd-aloop` and `v4l2loopback` modules allow one to set up a virtual soundcard device and a virtual video device, respectively. Once activated these loopback devices appear to videoconferencing clients as standard audio/video devices, except that audiovisual data is not coming from a real microphone or a capture card, but is instead sourced by other applications. In our setup we use `aplay` and `ffmpeg` to replay audio/video files into these virtual devices. The in-kernel device emulation is completely transparent to the clients, thus no client-side modification is required.

Another aspect of client emulation is client UI navigation. Each videoconferencing client has client-specific UI elements for interacting with a videoconferencing service, such as logging in, joining/leaving a meeting, switching layouts, etc. We automate UI navigation of deployed clients by emulating various input events (e.g., keyboard typing, mouse activity, screen touch) with OS-specific tools (e.g., `xdotool` for Linux, and `adb-shell` for Android). For each videoconferencing system, we script the entire workflow of its client.

Coordinated client deployments. Fully-emulated clients allow us to deploy the clients in public clouds and/or mobile testbeds for automated testing. The fact that we control audiovisual data feed for the clients as well as their UI navigation provides unique opportunities for us to gain, otherwise difficult to obtain, insights into the videoconferencing systems under test. For example, one client can be injected with a video feed with specific patterns (e.g., periodic ON/OFF signals, or high-/low-motion videos), and other clients receive the feed through a videoconferencing service. By comparing the injected feed and received feeds, we can evaluate different videoconferencing services. We can easily coordinate the activities of multiple participants in a given conferencing session to facilitate our analysis (e.g., only one user's screen is active at a time).

Platform-agnostic data collection. Even with client emulation and coordination, the closed nature of the existing videoconferencing systems (e.g., proprietary client software and end-to-end encryption) poses as a hurdle to comparing the systems with objective and unified metrics. That has led us to perform data collection in a *platform-agnostic* fashion as follows.

First, we derive some of the evaluation metrics from network-level monitoring and measurements. For example, we measure streaming lag by correlating packet timestamps on sender-side and on receiver-side. That way, we can evaluate the videoconferencing infrastructures without being influenced by specifics in client deployments. This, however, requires accurate clock synchronization among deployed clients. Fortunately, major public clouds already provide dedicated time sync services for tenant workloads with their own stratum-1 clock sources [18, 26].

In order to supplement network-based metrics with user-perceived quality metrics, we record videoconferencing sessions from individual participants' perspective, and assess the quality of recorded audios/videos across different platforms. While Zoom provides a local recording option for each participant, other services like Webex or Meet only allow a meeting host to record a session. In the end, we adopt a desktop recording approach as a platform-agnostic measure. We run a videoconferencing client in full screen mode, and use `simplescreenrecorder` to record the desktop screen with audio, within a cloud VM itself.

Finally, we also evaluate the videoconferencing systems from their clients' resource-utilization perspectives, which is particularly important for mobile devices. While these metrics can be influenced by client implementation, we believe that platform-driven factors (e.g., audio/video codecs) may play a bigger role.

3.2 Deployment Targets

Based on the design approach described above, we deploy emulated videoconferencing clients on a group of cloud VMs and Android mobile phones. Each of the cloud VMs hosts a videoconferencing client in a fully emulated setting to generate (via emulated devices) and/or receive a streaming feed, while Android devices only receive feeds from videoconferencing systems without device emulation.[1] In the following, we describe each of these deployment targets in more details.

Cloud VM. A cloud VM runs a videoconferencing client on a remote desktop in a fully-emulated environment. It consists of several components as shown in Fig. 1. *Media feeder* replays audio and video files into corresponding loopback devices. The audio and video files are either synthetically created, or extracted individually from a video clip with sound. *Client monitor* captures incoming/outgoing videoconferencing traffic with `tcpdump`, and dumps the trace to a file for offline analysis, as well as processes it on-the-fly in a separate "active probing" pipeline. In this pipeline, it discovers streaming service endpoints (IP address, TCP/UDP port) from packet streams, and performs round-trip-time (RTT) measurements against them. We use `tcpping` for RTT measurements because ICMP pings are blocked at the existing videoconferencing infrastructures. *Client controller* replays a platform-specific script for operating/navigating a client, including launch, login, meeting-join/-leave and layout change.

In order to host the cloud VMs, a public cloud must meet the following requirements. First, the cloud must *not* be used to operate the videoconferencing systems under test. For example, the majority of Zoom infrastructure is known to be hosted at AWS cloud [1]. If we run our emulated clients in the same AWS environment, Zoom will be heavily favored in our evaluation due to potential intra-cloud network optimization. To prevent such bias, we exclude any public cloud being used by the videoconferencing systems we tested. The cloud must also have reasonably wide geographic coverage. In the end we choose Azure cloud [16] as our benchmarking platform.

Android devices. We use Samsung Galaxy S10 and J3 phones, representative of both *high-end* and *low-end* devices (Table 2). The battery of the J3 is connected to a Monsoon power meter [31] which produces fine-grained battery readings. Both devices are connected to a Raspberry Pi (via WiFi to avoid USB noise on the power readings) which is used to automate Android UI navigation and to monitor their resource utilization (e.g., CPU usage). Both tasks are realized via Android Debugging Bridge (`adb`). The phones connect to the Internet over a fast WiFi – with a symmetric upload and download bandwidth of 50 Mbps. Each device connects to its own WiFi realized by the Raspberry Pi, so that traffic can be easily isolated and captured for each device.

[1]While Android devices can generate sensory data from their onboard camera/microphone, we mostly do not use them for reproducible benchmarking.

Name	Android Ver.	CPU Info	Memory	Screen Resolution
Galaxy J3	8	Quad-core	2GB	720x1280
Galaxy S10	11	Octa-core	8GB	1440x3040

Table 2: Android devices characteristics.

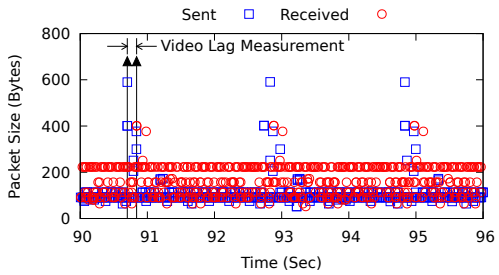


Figure 2: Video lag measurement.

4 QUALITY OF USER EXPERIENCE

In this section, we present QoE analysis results from our benchmark analysis of three major videoconferencing systems: Zoom, Webex and Meet. The experiments were conducted from 4/2021 to 5/2021.

4.1 Cloud VM Setup

Each cloud VM we deploy has 8 vCPUs (Intel Xeon Platinum 8272CL with 2.60GHz), 16GB memory and 30GB SSD. We make sure that the provisioned VM resources are sufficient for all our benchmark tests, which involve device emulation, videoconferencing, traffic monitoring and desktop recording. The screen resolution of the VM's remote desktop is set to 1900×1200. We use the native Linux client for Zoom (v5.4.9 (57862.0110)), and the web client for Webex and Meet since they do not provide a native Linux client.

4.2 Streaming Lag

First we evaluate streaming lag experienced by users (i.e., time delay between audio/video signals ingested by one user and those received by another). While measuring streaming lag in real-life videoconferencing is difficult, our emulated clients with synchronized clocks allow us to quantify the lags precisely. We purposefully set the video screen of a meeting host to be a blank-screen with periodic flashes of an image (with two-second periodicity), and let other users join the session with no audio/video of their own. Using such a bursty, one-way video feed allows us to easily determine the timing of sent/received video signals from network traffic monitoring.

For example, Fig. 2 visualizes the packet streams observed on the meeting host (sender) and another user (receiver). The squares represent the sizes of packets sent by a meeting host, and the circles show the sizes of packets received by a user. As expected there are periodic spikes of "big" packets (>200 bytes) that match periodic video signals sent and received. The first big packet that appears after more than a second-long quiescent period indicates the arrival of a non-blank video signal. We measure streaming lag between the meeting host and the other participant with the time shift between the first big packet on sender-side and receiver-side.

Region	Location	Name	Count
US	Iowa	US-Central	1
	Illinois	US-NCentral	1
	Texas	US-SCentral	1
	Virginia	US-East	2
	California	US-West	2
Europe	Switzerland	CH	1
	Denmark	DE	1
	Ireland	IE	1
	Netherlands	NL	1
	France	FR	1
	London, UK	UK-South	1
	Cardiff, UK	UK-West	1

Table 3: VM locations/counts for streaming lag testing.

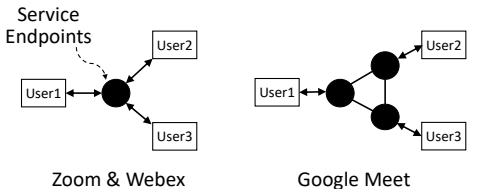


Figure 3: Videoconferencing service endpoints.

Admittedly, this network-based metric discounts any potential delay caused by a receiver-side client (e.g., due to stream buffering/decoding). However, it is effective to evaluate and compare lags induced by streaming infrastructures and their geographic coverage.

In the first set of experiments we deploy seven VMs in the US, as indicated by the "VM count" field in Table 3. We create a meeting session with one VM (in either US-east or US-west) designated as a meeting host, which then broadcasts periodic video signals to the other six participating VMs for two minutes before terminating the session. We collect 35-40 lag measurements from each participant during the session. For more representative sampling, we create 20 such meeting sessions with the same meeting host. Thus in the end we have a total of 700-800 lag measurements from each of the six VMs for a particular meeting host. We repeat this experiment on Zoom, Webex and Meet. In the second set of experiments we use seven VMs deployed in Europe, as shown in Table 3, and redo the above experiments with meeting hosts in UK-west and Switzerland.

We observe that the multi-user sessions created in this experiment are all relayed via platform-operated service endpoints with a designated fixed port number (`UDP/8801` for Zoom, `UDP/9000` for Webex, and `UDP/19305` for Meet).[2] Fig. 3 compares the three videoconferencing systems in terms of how their clients interact with the service endpoints for content streaming, which we discover from their traffic traces. On Zoom and Webex, a single service endpoint is designated for each meeting session, and all meeting participants send or receive streaming data via this endpoint. On Meet, each client connects to a separate (geographically close-by) endpoint, and meeting sessions are relayed among these multiple distinct endpoints.

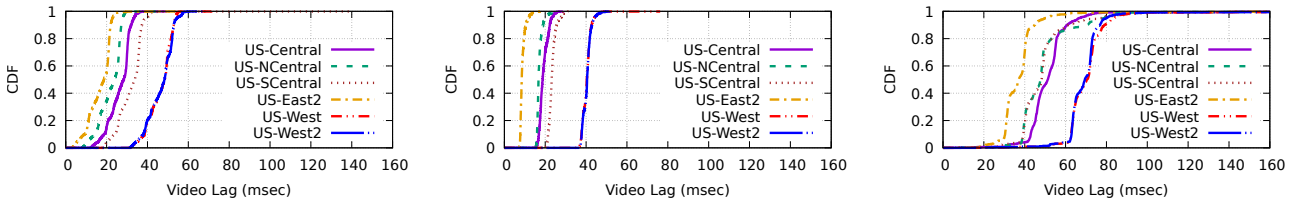
The number of distinct service endpoints encountered by a client varies greatly across different platforms. For example, out of 20 videoconferencing sessions, a client on Zoom, Webex and Meet encounters, on average, 20, 19.5 and 1.8 endpoints, respectively. On Zoom and Webex, service endpoints almost always change (with different IP addresses) across different sessions, while, on Meet, a client tends to stick with one or two endpoints across sessions.

Figs. 4–7 plot the CDFs of streaming lag experienced by clients in four different scenarios. In Figs. 4 and 5, we consider videoconferencing sessions among seven US-based clients (including a meeting host), where the host is located in either US-east or US-west. In Figs. 6 and 7, similarly we set up videoconferencing sessions among seven clients in Europe, with a meeting host in either UK or Switzerland. We make the following observations from the results.

4.2.1 US-based Videoconferencing. When sessions are created from US-east (Fig. 4), across all three platforms, streaming lag experienced by clients increases as they are further away from US-east, with the US-west clients experiencing the most lags (about 30 ms higher than the US-east client). This implies that streaming is relayed via the servers in US-east, where the meeting host resides. This is in fact confirmed by Fig. 8, where we plot RTTs between clients and service endpoints they are connected to. In the figure, RTTs measured by different clients are indicated with distinct dots. Each dot represents an average RTT (over 100 measurements) in a particular session. On Zoom and Webex, RTTs measured by US-east users are much lower than those by US-west users. On Meet, RTTs are uniform across clients due to its distributed service endpoint architecture as shown in Fig. 3.

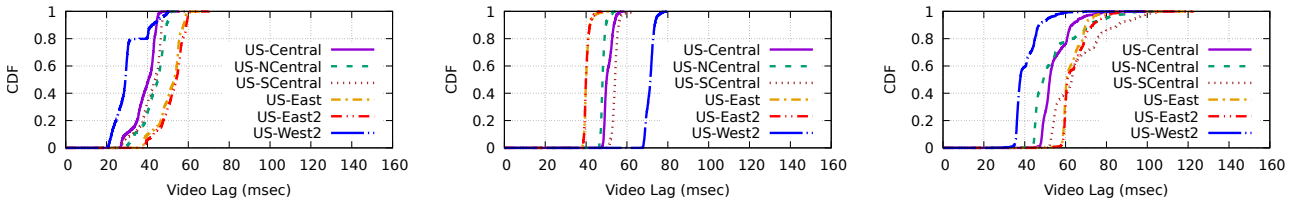
When a meeting host is in US-west (Fig. 5), geographic locality plays a similar role with Zoom and Meet, where the most far-away clients in US-east experience the worst lags. In case of Webex, however, the worst streaming lag is actually experienced by another user in US-west. According to the RTTs collected in this scenario for Webex (Fig. 9b), its service endpoints seem to be provisioned on the *east*-side of the US even when sessions are created in US-west, causing the streaming between US-west users to be detoured via US-east. Due to the geographically-skewed infrastructure, the lag distributions for US-west-based sessions are simply shifted by 30 ms from the US-east-based counterparts (Figs. 4b and 5b). One unexpected observation is that Meet sessions exhibit the worst lag despite having the lowest RTTs. This might be due to the fact that Meet sessions are relayed via *multiple* service endpoints, unlike Zoom and Webex (Fig. 3). In addition, although Meet's videoconferencing infrastructure appears to be distributed over wider locations (with lower RTTs), the total aggregate server capacity at each location may be smaller, hence leading to more load variation.

[2] One exceptional case we observe is that, on Zoom, if there are only two users in a session, peer-to-peer streaming is activated, where they stream to each other *directly* on an ephemeral port without going through an intermediary service endpoint.



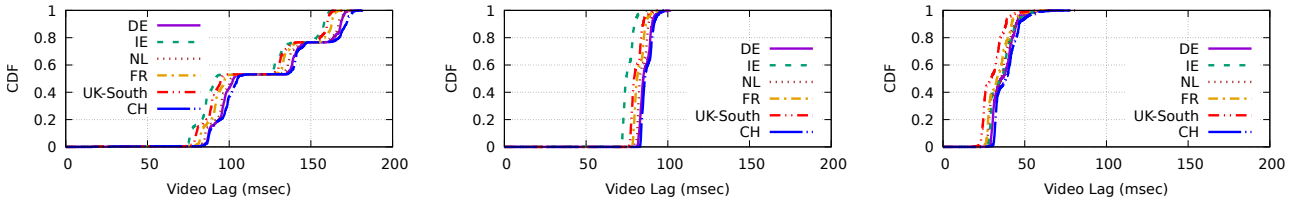
(a) Zoom (b) Webex (c) Meet

Figure 4: CDF of streaming lag: meeting host in US-east.



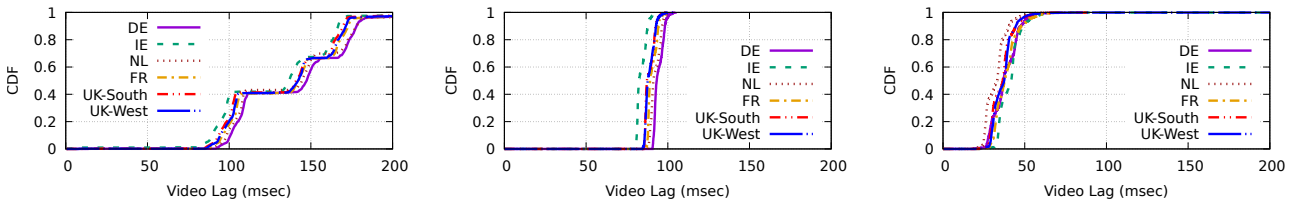
(a) Zoom (b) Webex (c) Meet

Figure 5: CDF of streaming lag: meeting host in US-west.



(a) Zoom (b) Webex (c) Meet

Figure 6: CDF of streaming lag: meeting host in UK-west.



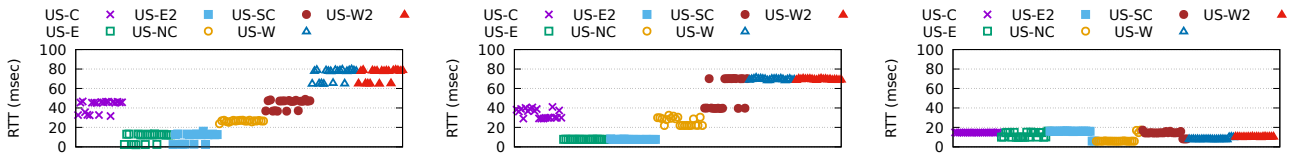
(a) Zoom (b) Webex (c) Meet

Figure 7: CDF of streaming lag: meeting host in Switzerland.

4.2.2 Non-US-based videoconferencing. According to Figs. 6–7, when sessions are set up among clients in Europe, Zoom/Webex clients experience much higher lags than Meet users. When compared to the Zoom/Webex sessions created in US-east, clients in Europe experience 55–75 ms and 45–65 ms higher median lags on Zoom and Webex, respectively. The reported RTTs (Figs. 10 and 11) show that the clients closer to the east-coast of US (e.g., UK and Ireland) have lower RTTs than those located further into central Europe (e.g., Germany and Switzerland). These observations suggest that the service infrastructures used are located somewhere in US.[3]

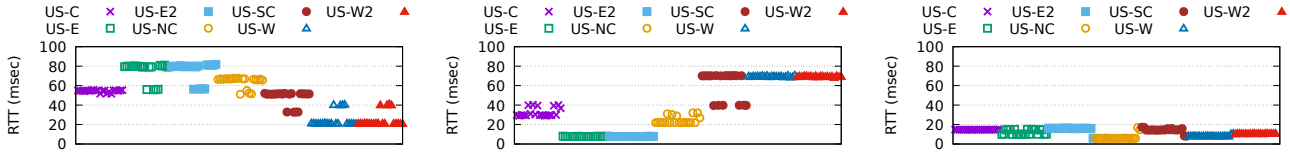
Comparing Zoom and Webex, one can see that RTTs to service endpoints on Zoom vary much more widely across different sessions. In fact, as shown in Figs. 10a and 11a, RTTs tend to spread across three distinct ranges that are 20 ms and 40 ms apart, which causes step-wise lag distributions in Figs. 6a and 7a. This suggests that Zoom may be employing *regional load balancing* within the US when serving non-US sessions. Whereas in Webex, RTTs to service endpoints consistently remain close to the trans-Atlantic RTTs [15], indicating that non-US sessions are relayed via its infrastructure in US-east. In case of Meet, its distributed service endpoints allow clients in Europe to enjoy stream lags that are comparable to US-based counterparts without any artificial detour. The reason why its streaming lag is lower

[3]We are not able to pinpoint the locations of the infrastructures because `traceroute`/`tcptraceroute`-probings are all blocked.



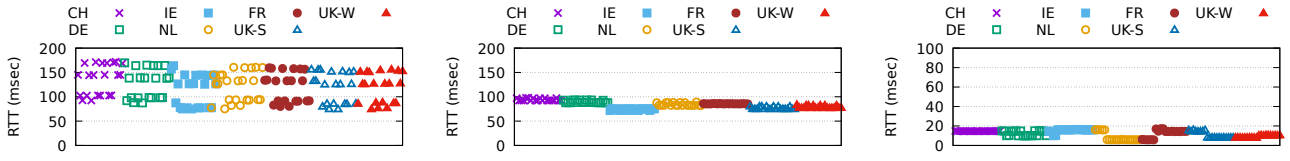
(a) Zoom (b) Webex (c) Meet

Figure 8: Service proximity: meeting host in US-east.



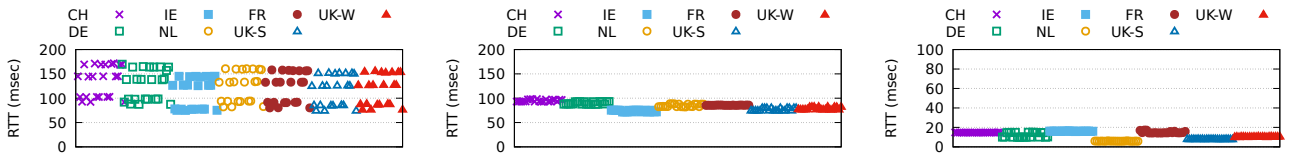
(a) Zoom (b) Webex (c) Meet

Figure 9: Service proximity: meeting host in US-west.



(a) Zoom (b) Webex (c) Meet

Figure 10: Service proximity: meeting host in UK-west.



(a) Zoom (b) Webex (c) Meet

Figure 11: Service proximity: meeting host in Switzerland.

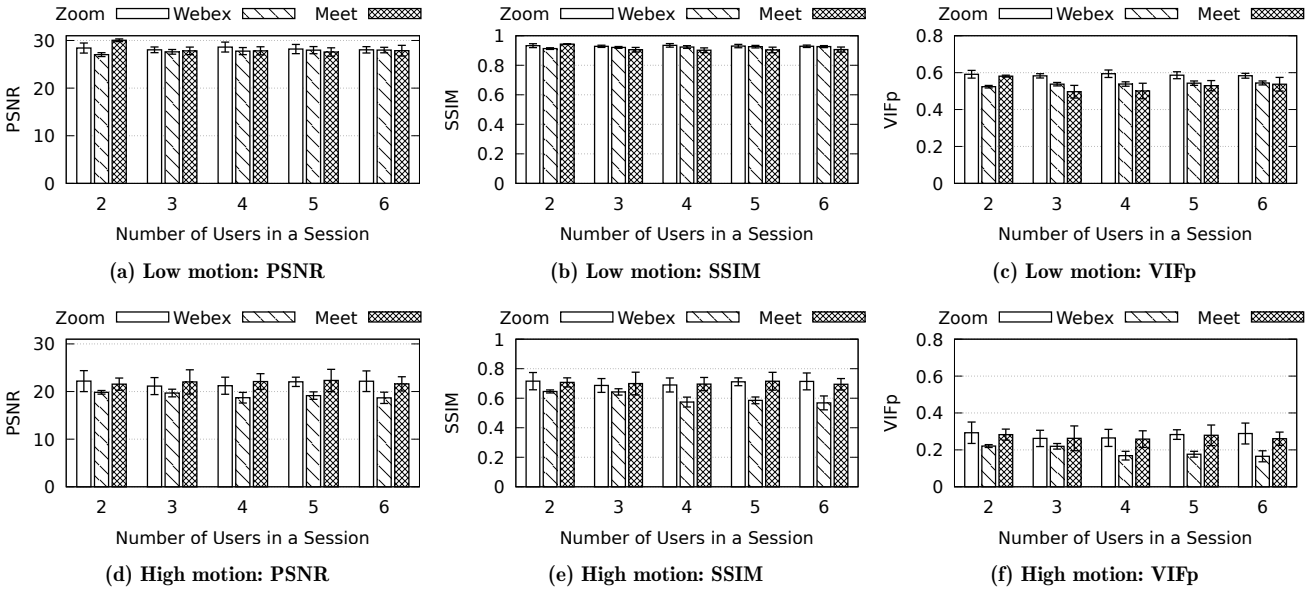
in Europe than in the US may be because the end-to-end latency among the clients (connected via service endpoints) in Europe may be smaller than that in the US. Average RTT within Europe is indeed smaller than that in the US [15].

4.3 User-Perceived Video Quality

Next we shift our focus to user-perceived quality of videoconferencing. A videoconferencing client typically captures a single person view against a stationary background, but it is also possible to have high-motion features in streamed content if a participant is joining a session from a mobile device, sharing a video playback with dynamic scenes, or showing a media-rich presentation, etc. In general, however, little is known about how different videoconferencing systems measure up to one another in terms of user-perceived quality under different conditions.

For this evaluation we prepare two distinct video feeds with 640×480 resolution: (i) a low-motion feed capturing the upper body of a single person talking with occasional hand gestures in an indoor environment, and (ii) a high-motion tour guide feed with dynamically moving objects and scene changes. On each videoconferencing system, we use a designated meeting host VM to create 10 five-minute long sessions, and inject the low-/high-motion videos into the sessions in an alternating fashion (hence two sets of five sessions). In each session, we let N clients join the session and render the received video feed in *full screen* mode while their desktop screen is recorded locally. For desktop recording, we use `PulseAudio` as audio backend, and set the video/audio codec to `H.264` (30 fps) and `AAC` (128 Kbps), respectively. We repeat the whole experiment as we vary N from one to five.[4]

[4] A typical number of users in a videoconferencing session is less than five [22].



(a) Low motion: PSNR (b) Low motion: SSIM (c) Low motion: VIFp

(d) High motion: PSNR (e) High motion: SSIM (f) High motion: VIFp

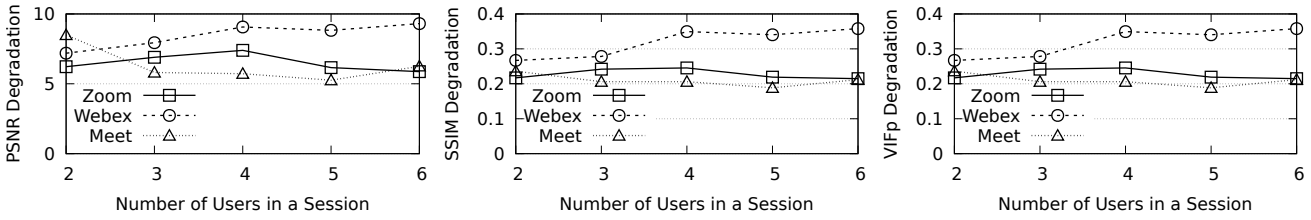
Figure 12: Video QoE metrics comparison (US).

Figure 13: Video screen with padding.

We compare the originally injected videos and recorded videos in terms of their quality with `VQMT` [6]. The `VQMT` tool computes a range of well-known objective QoE metrics, including PSNR (Peak Signal-to-Noise Ratio), SSIM (Structural Similarity Index Measure) [39] and VIFp (Pixel Visual Information Fidelity) [35]. Each of these metrics produces *frame-by-frame similarity* between injected/recorded videos. We take an average over all frames as a QoE value. One issue that complicates accurate quality comparison is the fact that the video screen rendered by a client is partially blocked by client-specific UI widgets (e.g., buttons, user thumbnails, etc.), even in full screen mode. To avoid such partial occlusion inside the video viewing area, we prepare video feeds with enough padding (Fig. 13). When recorded videos are obtained, we perform the following post-processing on them before analysis. We first crop out the surrounding padding and resize video frames to match the content layout and resolution of the injected videos. On top of that, we synchronize the start/end time of original/recorded videos with millisecond-level precision by trimming them in a way that per-frame SSIM similarity is maximized.

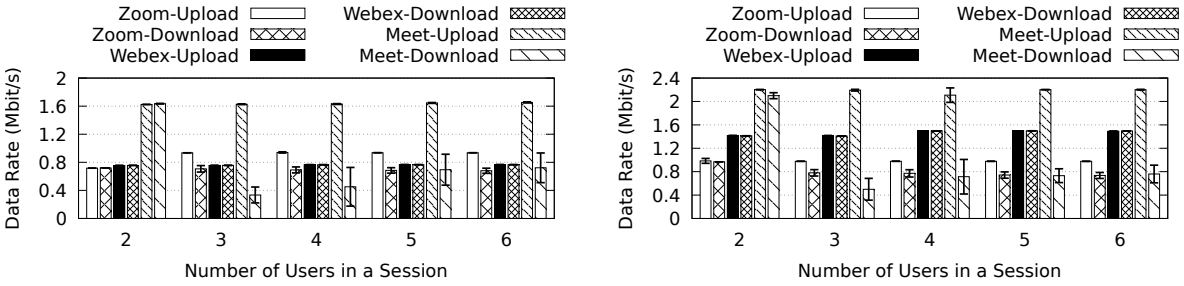
4.3.1 US-based Videoconferencing. We use one cloud VM in US-east designated as a meeting host which broadcasts a stream, and up to five other VMs in US-west and US-east receiving the stream as passive participants. Fig. 12 compares the quality of video streaming for these VMs in terms of three QoE metrics (PSNR, SSIM & VIFp) as the number of users in a session (N) increases. The height of bars indicates average QoE values across all sessions, with the errorbars being standard deviations. For easy comparison between low-motion and high-motion feeds, Fig. 14 shows the amount of QoE reduction with high-motion feeds (compared to low-motion feeds). Figs. 15a and 15b show the corresponding data rates for these sessions.

We make the following observations from the figures. Comparing Figs. 12a–12c against Figs. 12d–12f, one can find that, across all three platforms, low-motion sessions experience less quality degradation than high-motion sessions because their video feeds contain largely static background. The amount of decrease in QoE values between low-motion/high-motion sessions (Fig. 14) is significant enough to downgrade mean opinion score (MOS) ratings by one level [30]. On Webex, QoE degradation in high-motion scenario tends to become more severe with more users. Whereas no such consistent pattern is observed in Zoom and Meet. The QoE results from low-motion sessions (Figs. 12a–12c) show that there is a non-negligible QoE drop between N=2 and N>2 on Meet. We find that, on Meet, the data rate for two-user sessions (1.6–2.0 Mbps) is significantly higher than other multi-user sessions (0.4–0.6 Mbps) (Fig. 15). Such higher traffic rate with N=2 helps with the QoE of low-motion sessions, but does not contribute much to the QoE of high-motion sessions. Among the three, Webex exhibits the most *stable* QoE across



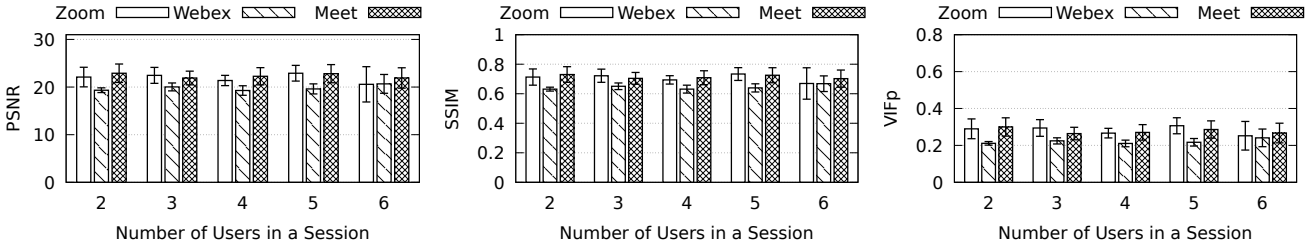
(a) PSNR (b) SSIM (c) VIFp

Figure 14: Video QoE reduction when video feeds are changed from low-motion to high-motion (US).



(a) US: low-motion (b) US: high-motion

Figure 15: Upload/download data rates (US). The data rate is computed from Layer-7 payload length in pcap traces. "X-Upload" shows the average upload rate of a meeting host (using Zoom, Webex, or Meet) who is broadcasting a video feed, while "X-Download" indicates the average download rate of the clients who are receiving the feed. All uploads/downloads occur via cloud-based relay, except for Zoom with N=2 which uses peer-to-peer traffic.



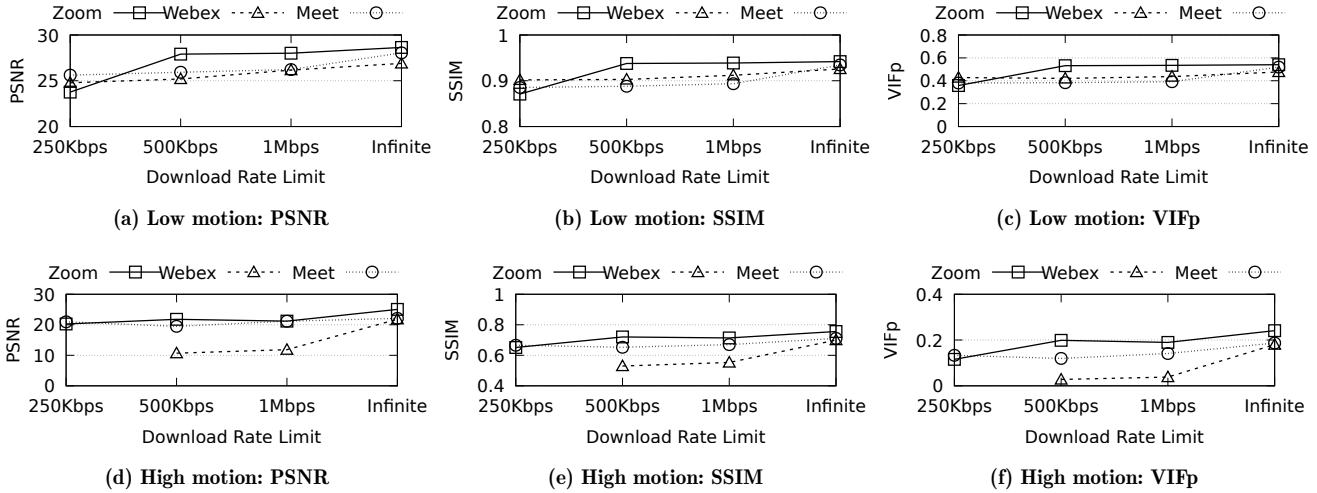
(a) High motion: PSNR (b) High motion: SSIM (c) High motion: VIFp

Figure 16: Video QoE metrics comparison (Europe).

different sessions. A similar behavior is observed in Figs. 4–7, where Webex shows the least variance in streaming lag as well.

Traffic-wise (Fig. 15), we focus on two aspects: (1) data rate difference between low-motion and high-motion feeds, and (2) data rate variation across multiple sessions with the same feed. All three systems send out a low-motion video feed in a lower rate than a high-motion counterpart as the former is more compressible. The rate reduction in low-motion streams is the highest in Webex, where its low-motion sessions almost halve the required downstream bandwidth. With a given video feed, Webex shows virtually no fluctuation in data rate across multiple sessions. On the other hand, Meet reduces its data rate in low-motion sessions roughly by 20% compared to high-motion sessions, but with much more *dynamic* rate fluctuation across different sessions than Zoom/Webex. Zoom exhibits the least difference (5–10%) in data rate between low-motion and high-motion sessions. Its downstream data rate is slightly higher with peer-to-peer streaming (∼1 Mbps; N=2) than with cloud-based relay (∼0.7 Mbps; N>2). When QoE metrics and data rates are considered together, Zoom appears to deliver the best QoE in the most bandwidth-efficient fashion, at least in the US.

4.3.2 Non-US-based videoconferencing. We repeat the QoE analysis experiment using a set of VMs created in Europe, one VM in Switzerland designated as a meeting host, and up to five other VMs (in France, Germany, Ireland, UK) joining the session created by the host. Fig. 16 shows QoE values on three systems with high-motion sessions. The results from



(a) Low motion: PSNR (b) Low motion: SSIM (c) Low motion: VIFp

(d) High motion: PSNR (e) High motion: SSIM (f) High motion: VIFp

Figure 17: Effect of bandwidth constraints on video quality.

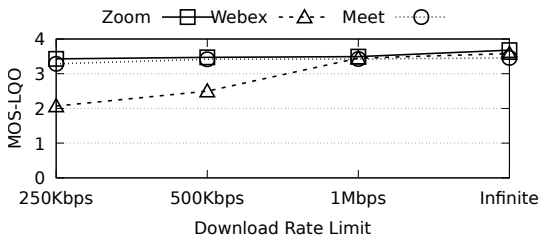


Figure 18: Audio quality under bandwidth constraints. MOS-LQO is computed in speech mode for low-motion sessions which contain only human voice.

low-motion sessions are similar to those collected in the US, and thus are omitted. When compared side-by-side, Meet maintains a slight edge in QoE metrics among three systems, potentially due to its European presence. In case of Zoom, although its average QoE appears to be similar to that of Meet, its QoE variation across different sessions is higher than Meet for high N. This observation is aligned with our earlier finding in Fig. 6a, where we show that its regional load balancing causes more variable streaming lag in Europe.

4.4 Streaming under Bandwidth Constraints

The experiments presented so far are conducted in an *unlimited* bandwidth environment. The cloud VMs used have a bidirectional bandwidth of multi-Gbps [3], which far exceeds the measured data rates of 1–2 Mbps (Fig. 15). In the next set of experiments, we apply artificial bandwidth caps on our cloud VM and examine its effect on QoE. We use Linux `tc/ifb` modules to enable traffic shaping on incoming traffic. Here we present QoE metric analysis not just for video but also for audio. We extract video and audio data separately from recorded videoconferencing sessions. For audio QoE analysis, we perform the following processing on extracted audio. First, we normalize audio volume in the recorded audio (with EBU R128 loudness normalization), and then synchronize the beginning/ending of the audio in reference to the originally injected audio. We use the `audio-offset-finder` tool for this. Finally, we use the `ViSQOL` tool [19] with the original/recorded audio data to compute the MOS-LQO (Mean Opinion Score - Listening Quality Objective) score, which ranges from 1 (worst) to 5 (best).

Figs. 17 and 18 show how video/audio QoE metrics change under various rate-limiting conditions. Each dot in the figures represents the average QoE values of five 5-minute long sessions.

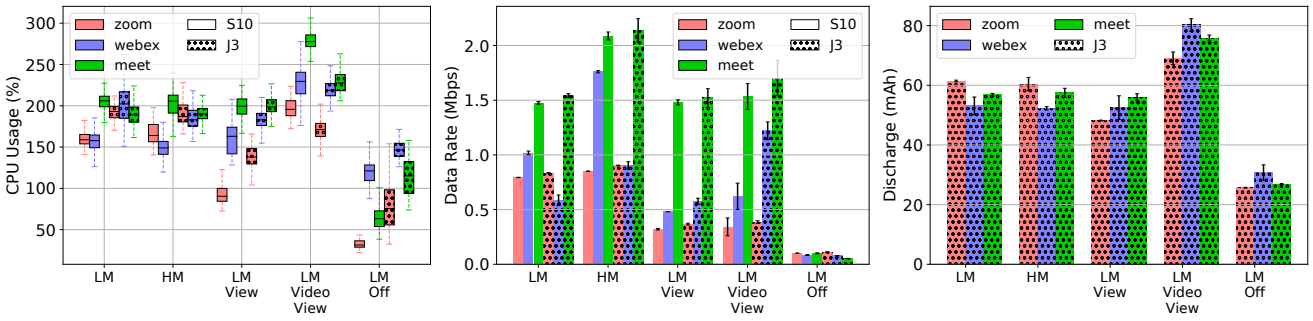
Video QoE. Overall, Zoom tends to maintain the best QoE with decreasing bandwidth limits, although there is sudden drop in QoE with a bandwidth cap of 250 Kbps. Meet maintains more graceful QoE degradation across all scenarios. Webex suffers from the most significant QoE drops with smaller bandwidth caps. With bandwidth $\leq$ 1 Mbps, video frequently stalls and even completely disappears and reappears on Webex.

Audio QoE. Compared to non-negligible QoE drops in video, audio QoE levels remain virtually constant on Zoom and Meet, probably due to the low data rate of audio (90 Kbps for Zoom and 40 Kbps for Meet).[5] However, voice quality on Webex, even with its low rate (45 Kbps), is relatively sensitive to bandwidth limits, starting to deteriorate noticeably (e.g., manifested as distorted/paused sound) with a limit of 500 Kbps or less.

5 RESOURCE CONSUMPTION

After having investigated user-perceived QoE offered by three major videoconferencing systems, we shift our attention to their client-side *resource consumption*, such as CPU, bandwidth and battery usages. For this analysis, we resort to

[5]We measure their audio rates separately using audio-only streams.



(a) CPU usage (b) Data rate (c) Battery discharge (J3 only)

Figure 19: Resource consumption evaluation. Samsung S10 and J3 (Android).

two Android devices: S10 (high-end) and J3 (low-end) as described in Table 2. In addition to these devices, each experiment includes one cloud VM designated as a meeting host. Since the Android devices are located in a residential access network of the east-coast of US, we run the meeting host in a US-east VM. We set the videoconference duration to five minutes, and repeat each experiment five times. The meeting host streams the two previously introduced video feeds: low-motion (`LM`) and high-motion (`HM`).

At the Android devices, we consider several device/UI settings which can affect videoconferencing sessions. Unless otherwise noted by the label "`Video`", each device's camera is turned off to minimize noise, and the incoming video feed is displayed in full screen. Given the low-motion video, we: 1) change the videoconferencing client's view into a gallery-view (`LM-View`), which assigns to each videoconference participant an equal portion of the screen,[6] 2) turn on the cameras and the gallery-view (`LM-Video-View`), and 3) turn off both camera and screen (`LM-Off`), simulating a driving scenario.

CPU usage. Fig. 19a shows boxplots of CPU usage sampled every three seconds for the experiment duration across all devices and scenarios. Each boxplot accounts for CPU samples collected across five repetitions of an experiment. We report CPU usage in absolute numbers, e.g., 200% implies full utilization of two cores. If we focus on the `LM` and `HM` scenarios for S10 (high-end), the figure shows that Zoom and Webex have comparable CPU usage (median of 150–175%), while Meet adds an extra 50%. When we focus on J3 (low-end device), CPU usage among the three clients is instead comparable (median around 200%). This indicates a dynamic behavior of the Meet client which only grabs more resources if available.

Zoom is the only client which benefits from the gallery view (both when the device's camera is on or off), reducing its CPU usage by 50% on both devices. Meet sees no benefit from this setting, which is expected given that it has no direct support for it, *i.e.,* the meeting host's video still occupies about 80% of the screen. Surprisingly, Webex does not benefit from its gallery view, even causing a slight CPU increase on S10. Irrespective of the videoconferencing client, activating the device's camera (`LM-Video-View`) adds an extra 100% and 50% of CPU usage on S10 and J3, respectively. The higher CPU usage on S10 is due to a better camera with twice as many megapixels (10M), HDR support, etc.

Finally, CPU usage is minimized when the device screen is off. However, while Zoom and Meet's CPU usage is reduced to 25–50% (S10), Webex still requires about 125%. This result, coupled with the lack of benefit of Webex's gallery setting, indicates some room for Webex to improve their Android client with more careful detection of user settings, as achieved by its competitors.

Data rate. We now focus on the Layer-7 download data rate, computed directly from pcap traces. Fig. 19b shows the average download data rate per client, device, and setting, with errorbars reporting the standard deviation. If we focus on the high-end device (S10) and the `LM`/`HM` scenarios, the figure shows a trend similar to Fig. 19a: Zoom uses the lowest data rate while Meet the highest, and the HM video causes a significant data rate increase, with the exception of Zoom. When focusing on the low-end device (J3), we notice that only Webex shows a "truly" adaptive behavior, *i.e.,* lower data rate for both LM and the low-end device. Zoom instead sticks to a somehow default data rate (750 Kbps), while Meet only takes into account the quality of the video, and not the target device. As previously observed for CPU usage, Meet's data rate is not impacted by the client-side gallery view, while it drops Zoom's data rate by 50%, both with a device's video on and off. Webex's gallery view is instead less data efficient, particularly when a device's video is on. In this case, J3 reports a significant increase in the data rate (more than doubled compared to the `LM` scenario) due to the video sent by S10. In comparison, S10 reports a much lower video rate (700 Kbps vs. 1.2 Mbps) due to the J3's lower quality camera, as well as lack of light due to its location in the lab. Finally, the `LM-Off` scenarios confirm that no video is streamed when the screen is off, and just 100–200 Kbps are needed for audio.

[6] Meet has no support for this feature. We approximate it by "zooming out", *i.e.,* revealing UI controls like mute or hang up.

Table 4: Data rate and CPU usage with various videoconference sizes (N). Each cell reports statistics for S10/J3.

N	Client	Full screen		Gallery	
		Data rate (Mbps)	CPU (%)	Data rate (Mbps)	CPU (%)
3	Zoom	0.85/0.9	164/186	0.33/0.37	102/148
	Webex	1.76/0.9	148/183	0.57/0.59	149/186
	Meet	2.08/2.13	205/190	2.08/2.11	209/200
6	Zoom	0.92/0.94	189/212	0.71/0.73	101/152
	Webex	1.75/0.9	140/195	0.43/0.47	155/184
	Meet	2.25/2.33	257/211	2.15/2.24	235/219
11	Zoom	0.91/0.96	191/211	0.73/0.75	100/150
	Webex	1.76/0.89	141/194	0.48/0.43	154/182
	Meet	2.24/2.36	258/210	2.16/2.26	236/220

Battery usage. Next, we report on the battery consumption associated with videoconferencing. In this case, we only focus on J3, whose (removable) battery is connected to a Monsoon power meter [31]. Fig. 19c confirms that videoconferencing is an expensive task on mobile, draining up to 40% of its 2600mAh battery during an one-hour conference with camera on. Overall, the figure shows no dramatic difference among the three clients, whose battery usage is within 10% of each other. Zoom is the most energy efficient client with gallery view (`LM-View`), which provides a 20% reduction compared with `LM`. Gallery view does not provide benefits to both Webex and Meet, similar to what we reported for both CPU usage and data rate. Finally, turning off the screen and relying on audio only saves up to 50% of a user's battery.

Videoconference size. Finally, we investigate the impact of the number of conference participants on client-side resource utilization. To do so, in addition to the meeting host, we introduce up to eight cloud-VMs as participants. To further stress the devices under test, we configure the host as well as the extra eight cloud VMs to stream a high-motion video simultaneously. We consider two UI settings in the Android clients: *full screen* and *gallery*.

Table 4 summarizes the results; we report on average data rate and median CPU utilization per device (S10/J3, in each cell) and scenario. Note that $N = 3$ is the scenario we have evaluated so far, *i.e.,* one cloud VM as a meeting host plus two Android devices. In full screen mode, Meet incurs a significant increase in both data rate (10%) and CPU usage (50%). This is because, even in full screen, Meet still shows a small preview of the video of the other two participants (plus the device's video, if on). Conversely, Zoom and Webex appear visually equivalent in full screen regardless of N. Still, for Zoom we observe a small increases in its data rate (5%) and CPU (12%) suggesting that some additional video streams are being buffered in the background with the goal to minimize latency in case a user decides to change the view into *gallery*. Although we could not capture such latency, we visually confirmed that both Zoom and Meet allow us to rapidly switch between participants, while Webex incurs some buffering time in the order of seconds.

If we focus on the gallery view, we see that the extra participants cause a twofold data rate increase for Zoom, but no additional CPU usage. A similar trend is observed for Webex CPU-wise, but in this case we also observe a counter-intuitive data rate reduction (from 600 Kbps down to 450 Kbps). Upon visual inspection, we find that this reduction is associated with a significant quality degradation in the video stream. Further increasing the participants to 11 does not lead to additional resource consumption. This happens because the gallery view of Zoom and Webex – as well as the only Meet's setting – show videos for up to four concurrent participants.

6 LIMITATIONS

We conclude the paper by discussing several limitations of our study and future works we plan to explore.

Effect of last mile. Our cloud-based vantage points may not represent the realistic last-mile network environments (e.g., broadband, wireless) of typical videoconferencing users from two perspectives. First, the upstream connectivity of cloud-emulated clients (with multi-Gbps available bandwidth) is too idealistic. While our experiments with bandwidth emulation (Section 4.4) and a mobile testbed (Section 5) are intended to address this limitation, a more realistic QoE analysis would consider dynamic bandwidth variation and jitter as well. Another caveat is that all our emulated clients are created inside a *single* provider network (i.e., Azure network). Thus any particular connectivity/peering of the Azure network might influence our cloud experiments. Ideally, the testbed should be deployed across multiple cloud providers to mitigate any artifact of a single provider, or even encompass distributed edge-based platforms provisioned across heterogeneous access networks (e.g., residential [20, 38], campus [2] and enterprise networks [11, 12]). In fact, moving the evaluation platform to the edge would allow us to extend the list of target videoconferencing systems to study.[7]

Free-tier vs. paid subscription. The validity of our findings is limited to the free-tier services. Given the prevalent multi-CDN strategies [9] and differentiated product offerings, user's QoE can change under different subscription plans or any special license arrangements. For example, Zoom is known to allow users with a university license to connect to AWS in Europe [34], which we do not observe with its free-tier/paid subscription plans. In case of Webex, we confirm that, with a paid subscription, its clients in US-west and Europe can stream from geographically close-by Webex servers (with RTTs < 20 ms). Our methodology allows us to easily extend the scope of our study beyond the free-tier services.

Videoconferencing scalability. Our QoE analysis targets small-size videoconferencing sessions (with up to 11 participants). An interesting question is how well user's QoE on each system scales as a videoconferencing session is joined by a moderate/large number of participants. One possible approach would use a mix of crowd-sourced human users (who would generate varied-size videoconferencing sessions) and cloud VMs (which would be under our control for detailed QoE analysis of such sessions).

[7] The use of Azure cloud prevents us from including Microsoft Team in our evaluation list.

Black-box testing. Our measurement study is a case of *black-box testing*, where we do not have any knowledge on inner workings of individual systems we study. As such we are severely limited in our ability to *explain* some of the observations we are making. That said, we still argue that our study presents a valuable contribution to the community. For one, our platform-agnostic methodology is general enough for any arbitrary videoconferencing systems. Also, the proposed evaluation scenarios can be a useful input to videoconferencing operators for enhancing their infrastructures and clients.

Videoconferencing client. Our emulated client runs on Linux only (via in-kernel virtual devices). We consider that the modern Linux environment is representative enough for videoconferencing due to the good Linux support [13] or the use of cross-platform web clients by the existing systems. For completeness, one can extend the client emulation beyond Linux, at least in a desktop environment, using similar device emulation and workflow automation tools on Windows [5, 17] and MacOS [4, 10]. The QoE analysis could then be extended to cover different mobile and desktop environments in a more comprehensive fashion.

REFERENCES

[1] 2020. AWS and Zoom Extend Strategic Relationship. https://press.aboutamazon.com/news-releases/news-release-details/aws-and-zoom-extend-strategic-relationship/.
[2] 2020. CloudLab. https://www.cloudlab.us.
[3] 2020. Fsv2-series – Azure Virtual Machines. https://docs.microsoft.com/en-us/azure/virtual-machines/fsv2-series.
[4] 2020. OBS (macOS) Virtual Camera. https://github.com/johnboiles/obs-mac-virtualcam.
[5] 2020. OBS-VirtualCam. https://github.com/CatxFish/obs-virtual-cam.
[6] 2020. VQMT: Video Quality Measurement Tool. https://www.epfl.ch/labs/mmspg/downloads/vqmt/.
[7] 2021. https://support.zoom.us/hc/en-us/articles/201362023-System-Requirements-for-PC-Mac-and-Linux.
[8] 2021. https://help.webex.com/en-us/WBX22158/What-are-the-Minimum-Bandwidth-Requirements-for-Sending-and-Receiving-Video-in-Cisco-Webex-Meetings.
[9] 2021. 2020 Pandemic Network Performance. Broadband Internet Technical Advisory Group. https://www.bitag.org/documents/bitag_report.pdf.
[10] 2021. Automator User Guide for Mac. https://support.apple.com/guide/automator/.
[11] 2021. AWS Snow Family. https://aws.amazon.com/snow/.
[12] 2021. Azure Stack Edge. https://azure.microsoft.com/en-us/products/azure-stack/edge/.
[13] 2021. Desktop client, mobile app, and web client comparison. https://support.zoom.us/hc/en-us/articles/360027397692-Desktop-client-mobile-app-and-web-client-comparison.
[14] 2021. Google Meet hardware requirements. https://support.google.com/meethardware/answer/4541234.
[15] 2021. IP Latency Statistics by Verizon. https://www.verizon.com/business/terms/latency/.
[16] 2021. Microsoft Azure. https://azure.microsoft.com.
[17] 2021. Power Automate - Microsoft Power Platform. https://powerautomate.microsoft.com.
[18] 2021. Time sync for Linux VMs in Azure. https://docs.microsoft.com/en-us/azure/virtual-machines/linux/time-sync.
[19] 2021. ViSQOL: Perceptual Quality Estimator for speech and audio. https://github.com/google/visqol.
[20] Hyunseok Chang, Adiseshu Hari, Sarit Mukherjee, and T. V. Lakshman. 2014. Bringing the Cloud to the Edge. In *Proc. IEEE Workshop on Mobile Cloud Computing*.
[21] Ana-Paula Correia, Chenxi Liu, and Fan Xu. 2020. Evaluating Videoconferencing Systems for the Quality of the Educational Experience. *Distance Education* 41, 4 (2020), 429–452.
[22] Augusto Espin and Christian Rojas. 2021. The Impact of the COVID-19 Pandemic on the Use of Remote Meeting Technologies. SSRN. https://dx.doi.org/10.2139/ssrn.3766889.
[23] Anja Feldmann, Oliver Gasser, Franziska Lichtblau, Enric Pujol, Ingmar Poese, Christoph Dietzel, Daniel Wagner, Matthias Wichtlhuber, Juan Tapiador, Narseo Vallina-Rodriguez, Oliver Hohlfeld, and Georgios Smaragdakis. 2020. The Lockdown Effect: Implications of the COVID-19 Pandemic on Internet Traffic. In *Proc. ACM Internet Measurement Conference (IMC)*.
[24] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S. Wahby, and Keith Winstein. 2018. Salsify: Low-Latency Network Video through Tighter Integration between a Video Codec and a Transport Protocol. In *Proc. NSDI*.
[25] Pan Hu, Rakesh Misra, and Sachin Katti. 2019. Dejavu: Enhancing Videoconferencing with Prior Knowledge. In *Proc. the 20th International Workshop on Mobile Computing Systems and Applications*. 63–68.
[26] Randall Hunt. 2017. Keeping Time With Amazon Time Sync Service. https://aws.amazon.com/blogs/aws/keeping-time-with-amazon-time-sync-service/.
[27] Lisa M. Koonin, Brooke Hoots, Clarisse A. Tsang, Zanie Leroy, Kevin Farris, Brandon Jolly, Peter Antall, Bridget McCabe, Cynthia B.R. Zelis, Ian Tong, and Aaron M. Harris. 2020. Trends in the Use of Telehealth During the Emergence of the COVID-19 Pandemic—United States, January–March 2020. *Morbidity and Mortality Weekly Report* 69, 43 (Oct. 2020).
[28] Craig Labovitz. 2020. Network traffic insights in the time of COVID-19: March 23-29 update. https://www.nokia.com/blog/network-traffic-insights-time-covid-19-march-23-29-update/.
[29] Kyle MacMillan, Tarun Mangla, James Saxon, and Nick Feamster. 2021. Measuring the Performance and Network Utilization of Popular Video Conferencing Applications. *arXiv preprint arXiv:2105.13478* (2021).
[30] Arghir-Nicolae Moldovan and Cristina Hava Muntean. 2017. QoE-aware Video Resolution Thresholds Computation for Adaptive Multimedia. In *Proc. IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*.
[31] Monsoon Solutions Inc. 2021. High Voltage Power Monitor. https://www.msoon.com.
[32] Adam Ozimek. 2020. *Economist Report: Future Workforce*. Technical Report. ClearlyRated. https://www.upwork.com/press/releases/economist-report-future-workforce.
[33] Otto Parra and Maria Fernanda Granda. 2021. Evaluating the Meeting Solutions Used for Virtual Classes in Higher Education during the COVID-19 Pandemic. In *Proc. the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*.
[34] Constantin Sander, Ike Kunze, Klaus Wehrle, and Jan Rüth. 2021. Video Conferencing and Flow-Rate Fairness: A First Look at Zoom and the Impact of Flow-Queuing AQM. In *Proc. International Conference on Passive and Active Measurement*.
[35] Hamid Rahim Sheikh and Alan C. Bovik. 2006. Image Information and Visual Quality. *IEEE Transactions on Image Processing* 15, 2 (2006).
[36] Ravinder Singh and Soumya Awasthi. 2020. Updated Comparative Analysis on Video Conferencing Platforms-Zoom, Google Meet, Microsoft Teams, WebEx Teams and GoToMeetings. *EasyChair Preprint no. 4026* (2020).
[37] Sri Srinivasan. 2020. Cisco Webex: Supporting customers during this unprecedented time. https://blog.webex.com/video-conferencing/cisco-webex-supporting-customers-during-this-unprecedented-time/.
[38] Matteo Varvello, Kleomenis Katevas, Mihai Plesa, Hamed Haddadi, and Benjamin Livshits. 2019. BatteryLab: A Distributed Power Monitoring Platform For Mobile Devices. In *Proc. HotNets '19*.
[39] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing* 13, 4 (2004).
[40] Cui Zou, Wangchuchu Zhao, and Keng Siau. 2020. COVID-19 Pandemic: A Usability Study on Platforms to Support eLearning. In *Proc. International Conference on Human-Computer Interaction*. Springer, 333–340.