Measuring the Performance and Network Utilization of Popular Video Conferencing Applications

Kyle MacMillan, Tarun Mangla, James Saxon, Nick Feamster
Department of Computer Science, University of Chicago
Chicago, IL, USA
{macmillan,tmangla,jsaxon,feamster}@uchicago.edu

ABSTRACT

Video conferencing applications (VCAs) have become a critical Internet application during the COVID-19 pandemic, as users worldwide now rely on them for work, school, and telehealth. It is thus increasingly important to understand the resource requirements of different VCAs and how they perform under different network conditions, including: how do application-layer performance metrics (e.g., resolution or frames per second) vary under different link capacity; how VCAs perform under temporary reductions in available capacity; how they compete with themselves, with each other, and with other applications; and how usage modality (e.g., gallery vs. speaker mode) affects utilization. We study three modern VCAs: Zoom, Google Meet, and Microsoft Teams. Answers to these questions differ substantially depending on VCA. First, the average utilization on an unconstrained link varies between 0.8 Mbps and 1.9 Mbps. Given temporary reduction of capacity, some VCAs can take as long as 50 seconds to recover to steady state. Differences in proprietary congestion control algorithms also result in unfair bandwidth allocations: in constrained bandwidth settings, one Zoom video conference can consume more than 75% of the available bandwidth when competing with another VCA (e.g., Meet, Teams). For some VCAs, client utilization can decrease as the number of participants increases, due to the reduced video resolution of each participant's video stream given a larger number of participants. Finally, one participant's viewing mode (e.g., pinning a speaker) can affect the upstream utilization of other participants.

CCS CONCEPTS

• **Networks** → **Network measurement**; • **Information systems** → **Multimedia streaming**.

KEYWORDS

Video conferencing, network measurement, real-time streaming

ACM Reference Format:
Kyle MacMillan, Tarun Mangla, James Saxon, Nick Feamster. 2021. Measuring the Performance and Network Utilization of Popular Video Conferencing Applications. In *Internet Measurement Conference (IMC '21), November 2–4, 2021, Virtual Event.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3487552.3487842

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

ACM ISBN 978-1-4503-9129-0/21/11…$TBA
https://doi.org/10.1145/3487552.3487842

1 INTRODUCTION

Internet users around the world have become increasingly dependent on video conferencing applications over the past several years, particularly as a result of the COVID-19 pandemic, which has caused many users to rely almost exclusively on these applications for remote work, education, healthcare, social connections, and many other activities. Some ISPs have reported as much as a three-fold increase in video conferencing traffic during an eight-month period in 2020 during the pandemic [5]. Our increased reliance on these video conferencing applications (VCAs) has highlighted certain disparities, especially given that nearly 15 million public school students across the United States during the pandemic lacked Internet access for remote education [23].

As cities and countries around the world moved to close this gap, many asked a series of simple questions geared towards understanding the level of connectivity that they needed to provide citizens to guarantee reliable, high-quality Internet experience: *What is the baseline level of Internet performance needed to support common video conferencing applications for the activities that people commonly use them for (e.g., education, remote work, telehealth)?* This was the question, in fact, that our own city officials asked us which motivated us to pursue this question in this paper. The question has been brought into focus even more over recent months as the United Stated Federal Communications Commission (FCC) continues to debate the levels of Internet speed that should be classified as a "broadband" service offering. The current standard is 25 Mbps downstream/3 Mbps upstream. With the rise of video conferencing applications, however, consumer Internet connections saw an increase in upstream traffic utilization. Some policy advocacy papers have claimed (without measurements) that the FCC should change its definition of "broadband" to a symmetric 20 Mbps connection on account of these trends. Moreover, because many users, especially in underserved regions, experience frequent connectivity disruptions, a comparative analysis of VCAs under dynamic (and degraded) network conditions can also shed light on the design practices of VCAs and help identify best design practices.

In light of these discussions and questions from municipal, state, and federal policymakers, we were motivated to explore the answers to questions concerning how much network resources video conferencing applications required, how they responded to network degradations and connectivity interruptions, how they compete with each other and with other applications, and how these questions vary depending on the modalities of use (e.g., gallery mode vs. speaker mode). Similar questions have of course been studied in the

past [2, 6, 7, 12, 36], but the vast majority of these studies are now at least a decade old, during which time both the VCAs themselves and how we use them has changed dramatically. New VCAs such as Google Meet, Microsoft Teams, and Zoom have emerged in the last few years alone. In light of these new VCAs and recent drastic shifts in usage pattern and our dependence on these applications, a re-appraisal of these questions is timely.

We study the following questions for three popular, modern video conferencing applications:

(1) What is the network utilization of common video conferencing applications?
(2) How do VCA performance metrics (e.g., resolution) vary under different link capacities?
(3) How do VCAs respond to temporary disruptions to connectivity? and how quickly do they recover when connectivity is restored?
(4) How do VCAs respond in the presence of competing traffic? How do they compete with themselves, with other VCAs, and with other applications?
(5) How do different usage modalities (number of participants, speaker vs. gallery viewing) affect network utilization?

We explore these questions by performing controlled laboratory experiments with emulated network conditions across a collection of clients, collecting application performance data using provided application APIs. We collect application data under a variety of network conditions and call modalities, including different numbers of participants and viewing modes. Many of the answers to these questions surprised us; as we will discover, sometimes the answers depend quite a lot on the particular video conferencing application. Table 1 summarizes our main findings and where they can be found in the paper. In particular, we found significant differences in network utilization, encoding strategy, and recovery time after network disruption. We also discovered that VCAs share available network resources with other applications differently. We confirm earlier findings [38], that one participant's choice of how to interact in the video conference can affect the network utilization of *other* participants.

Our findings have broad implications, for network management, as well as for policy. For network management, our findings concerning network utilization and performance under various network conditions have implications for network provisioning, as access ISPs seek to provision home networks to achieve good performance for consumers. From a policy perspective, our results are particularly important: questions about the throughput needed to support quality video conferencing, especially in multi-user households, was a question from city government officials that motivated this paper in the first place. Looking ahead, as city, state, and federal governments both subsidize broadband connectivity and build out new infrastructure, understanding how these increasingly popular video conferencing applications consume and share network resources is of critical importance.

2 BACKGROUND & EXPERIMENT SETUP

We provide a brief background on video conferencing transport, technology, and architecture followed by a description of our experiment setup.

The average network utilization on an unconstrained link ranges from 0.8 Mbps to 1.9 Mbps (§3.1).
Despite using the same WebRTC API, Meet and Teams browser clients differ significantly in how they encode video (e.g., frames per second, resolution) under reduced link capacity (§3.2).
Recovery times following transient drops in bandwidth are quite different and depend on both VCA's congestion control mechanism and the direction of the drop, i.e. uplink or downlink. However, all VCAs take at least 20 seconds to recover from severe uplink drops to 0.25 Mbps (§4).
Differences in proprietary congestion control also create unfair bandwidth allocations in constrained bandwidth settings. Zoom and Meet can consume over 75% of the available downlink bandwidth when competing with Teams or a TCP flow (§5).
Each participant's video layout impacts its own and others' network utilization. Pinning a user's video to the screen (speaker mode) leads to an increase in the user's uplink utilization (§6).

Table 1: *Main results, and where they can be found in the paper.*

2.1 Video Conferencing Applications

Most modern VCAs use Real-time Transport Protocol (RTP) [30, 31] or its variants [3, 18] to transmit media content. The audio and video data is generally transmitted over separate RTP connections. RTP also uses two other protocols in conjunction, Session Initiation Protocol (SIP) [28] to establish connection between clients and RTP Control Protocol (RTCP) [30] to share performance statistics and control information during a call. Despite using well-known protocols, VCAs can differ from each other significantly across the following dimensions:

- **Network mechanisms**: RTP and associated protocols are implemented in the application-layer. Thus, the specific implementation of the protocols can vary across VCAs. For instance, Zoom reportedly uses a custom extension of RTP [18]). Furthermore, the key transport-layer functions, i.e., congestion control and error recovery, can also be different and proprietary.
- **Application-layer parameters**: These include media encoding standards and default bitrates. More recent video codecs (e.g., VP9, H.264) can encode at the same video quality with fewer bytes when compared to older codec (e.g., VP8), albeit with a higher compute overhead [4].
- **Streaming architecture**: VCAs can either choose to use direct connections or use relay servers. Centralized servers are almost always used for multi-point calls to combine data from multiple users. VCAs can differ in the exact strategies of data combination as well as the geographic footprint of their servers. For instance, Zoom rapidly expanded its server infrastructure to support increased call volume because of COVID-19 [20].

Differences across one or more of these factors can lead to different VCA network utilization and performance. In this paper, we aim to dig deeper into some of these differences for a subset of VCAs.

2.2 Experiment Setup

Video Conferencing Applications: We study three popular VCAs—Zoom, Google Meet, and Microsoft Teams. These VCAs have been used extensively worldwide over the past year, especially in enterprises and educational institutions [34]. Teams and Zoom provide desktop applications as well as browser clients, whereas Meet is "native" in Chrome. Most of our tests are conducted using the desktop applications for Teams and Zoom, and using the Google Chrome browser for Meet. In-browser tests for Teams and Zoom are specified by *Teams-Chrome* and *Zoom-Chrome*, respectively. We use Chrome (Meet) version 89.0.4389, Zoom client version 5.6.1, and Teams client version 1.4.00.7556. Note that we use the enterprise versions of all three VCAs provided by our University accounts. Moreover, the associated Zoom account used for the experiments had HD enabled, while default settings are used for Meet and Teams.

Laboratory Environment: We conduct experiments in a controlled environment. We begin by describing our experimental setup for a 2-party call. We use two identical laptops, referred to as V1 and V2, representing the two VCA clients. Each laptop is a Dell Latitude 3300 with a screen resolution of 1366 × 768 pixels and running Ubuntu 20.04.1. The laptops have a wired connection to a Turris Omnia 2020 router and access a dedicated 1 Gbps symmetric link to the Internet. Each experiment consists of a call between V1 and V2 under a pre-specified network bandwidth profile and VCA. Traffic Control (`tc`), more specifically the hierarchical token bucket queuing discipline, is used at the router to emulate the network bandwidth profile. A pre-recorded talking-head video[1] with a resolution of 1280 × 720 is used as the video source for the call, using `ffmpeg`. This is done to both replicate a real video call and ensure consistency across experiments. All experiments are conducted with the laptop lid open and the application window maximized.

In all cases, the participants join calls that are initiated in real-time by one of the participants. For two-person calls, this means one participant launches the meeting and the other joins shortly thereafter. Such ad-hoc calls are different from scheduled calls wherein VCAs can perform few optimizations including pre-allocation of relay servers. While choice of relay servers can significantly impact end-user experience, especially for geographically distributed clients [16], we assume that the WAN connection is not the bottleneck in our experiments for three reasons: i) all of our participants are located within the University network with well-provisioned access link and upstream transit connections, ii) the response time to the relay server (Google) and the last responsive hop (Microsoft and Zoom) are within 25ms, and iii) our experiments are conducted over the course of many weeks so any temporary congestion that may have occurred will not influence our results. As a result, it is likely that there is well-provisioned infrastructure that supports Teams, Meet, and Zoom and provide good service. Finally, we conduct experiments throughout the day and do not control for any time of day effects on server load.

Automating Experiments: To conduct experiments at scale, we automated the entire call process. We take several steps to enable automated calls. We use the Python PyAutoGUI package [27] to automate joining and leaving calls. The package allows us to programmatically control the keyboard and the mouse by specifying coordinates or visual elements on the screen. For Zoom-Chrome, we encountered CAPTCHA before joining a call on the default browser. Using the Selenium-based Chrome browser [32], however, enabled us to bypass the CAPTCHA. Note that the experiments using Selenium are run exactly as they would be in default Chrome browser. The workflow is controlled from V1, with TCP sockets used to coordinate between V1 and V2. We modify this setup slightly for subsequent experiments (e.g., multi-party calls); those slight modifications are described in the respective sections.

[1]It would be inappropriate to use the device webcam as the video, because without movement, VCAs compress the video and ultimately send at a much lower rate than during a normal call.

VCA	Utilization (Mbps)	
	Upstream	Downstream
Meet	0.95	0.84
Teams	1.40	1.86
Zoom	0.78	0.95

Table 2: *Unconstrained network utilization.*

3 STATIC NETWORK CONDITIONS

In this section, we study the effect of network capacity on VCA performance.

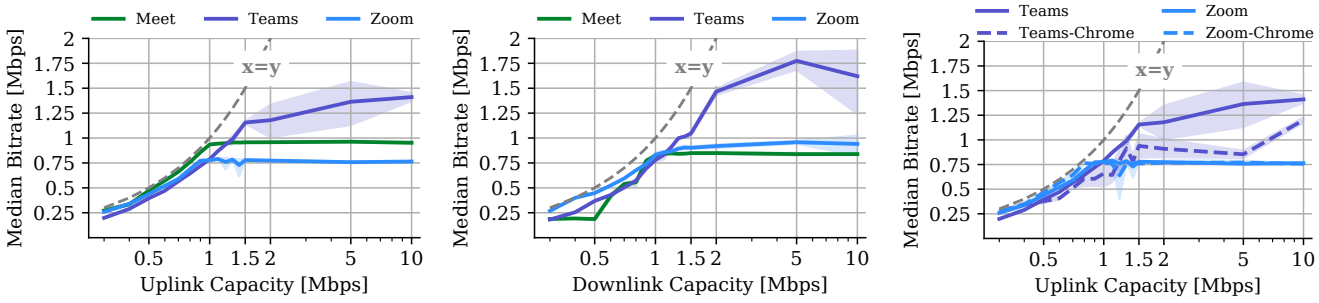
Method: We conduct a series of experiments, each consisting of a 2.5-minute call between two clients, V1 and V2 (as described in Section 2.2), under a specific shaping level. We conduct two sets of experiments, shaping first the uplink and then the downlink. We constrain throughput to {0.3, 0.4, . . . , 1.4, 1.5, 2, 5, 10} Mbps (in both the upstream and downstream directions). For each condition, we perform five 2.5-minute experiments; our data show that due to relatively low variance in most observations, this number of experiments is sufficient to achieve statistically significant results. For Zoom and Teams clients, we perform measurements both for native clients as well as for browser-based clients, referred to as Zoom-Chrome and Teams-Chrome, respectively.

3.1 Network Utilization

Constrained upstream utilization: Figure 1a shows the median sent network bitrate for different uplink capacities. We observe differences in upstream network utilization among VCAs given the same available uplink network capacity. In the case of a 10 Mbps uplink, for example, the average upstream utilization for Teams-native is 1.44 Mbps whereas it is only 0.95 for Meet and 0.77 Mbps for Zoom. All three VCAs utilize the uplink efficiently (above 85%) when that link is severely constrained (0.8 Mbps or lower), although Meet's bitrate is slightly higher than the other VCAs.

Constrained downstream throughput: We explored the effect of constrained downstream capacity on VCAs' network utilization. Figure 1b shows this result. As with a constrained uplink, the VCAs differ in terms of their downlink utilization under unconstrained link. The unconstrained downlink utilization differs from unconstrained upstream utilization, as shown in Table 2. To better understand this phenomenon, we analyzed the traffic captured from both clients.

In the case of Teams, we found that the total sent traffic from V1 is almost the same size as the total received traffic at V2, and vice versa. We suspect the small differences in upstream and downstream utilization may largely be due to the variability in utilization across



(a) *Uplink bandwidth vs network bitrate* **(b)** *Downlink bandwidth vs network bitrate* **(c)** *Impact of VCA platform*

Figure 1: *Network utilization under different shaping levels. The bands represent 90% confidence intervals.*

experiments in Teams, as is also evident in the larger confidence intervals as compared to Zoom and Meet.

In contrast, Zoom's utilization patterns were asymmetric: we found an asymmetry in sent and received data rates at both clients. For instance, in a single instance with 10 Mbps downlink shaping, V2 sent a median 0.85 Mbps and V1 received a median 1.10 Mbps. Investigating this asymmetry further, we discovered an interesting phenomenon: We found that Zoom uses a relay server instead of direct communication. Additionally, related work by Nistico et al. [26] suggests that Zoom uses Forward Error Correction (FEC) for error recovery. This is further supported by a related patent from Zoom itself, which talks about a methodology to generate FEC data at the server [19]. We suspect that the extra data may thus correspond to FEC added by the relay server, leading to asymmetric upstream and downstream utilization.

In addition to the asymmetric unconstrained utilization, Meet exhibits markedly different behavior with a constrained downlink. Specifically, the network utilization with constrained downstream throughput (< 0.8 Mbps) is only 39–70% (Figure 1b), while it is more than 90% in the case of a constrained uplink (Figure 1a). Upon further exploration, we discovered that Meet also uses a relay server, as well as *simulcast*, wherein the sender (V2) transmits multiple copies of the video to the server, each at a different quality level [26]. The server then relays one of the quality streams to V1 depending on the inferred available capacity of the server-V1 link. We observe two simultaneous video streams in our experiments, one at 320x180 and other at 640x360 resolution. When downstream throughput is constrained, the relay server cannot switch to a higher quality video and keeps sending at low quality bitrate. This explains why Meet's network utilization at 0.5 Mbps is only 0.19 Mbps, almost similar to its utilization at 0.3 Mbps. The use of simulcast also explains the higher upstream utilization as compared to downstream utilization.

Browser vs. native client utilization: Figure 1c compares the upstream utilization of Zoom and Teams between their respective native and Chrome clients. Zoom's utilization is similar across the native and browser-based platforms; in contrast, we find significant difference between the Teams native and browser client. When uplink capacity is shaped to 1 Mbps, the native client uses 0.84 Mbps, whereas the browser version uses only 0.61 Mbps. We found a similar difference between the native and browser versions when downstream throughput is constrained.

3.2 Application Performance

Next, we explore how video conference application performance varies depending on network capacity. To do so, we use the WebRTC stats API available in Google Chrome and obtain application metrics for Teams-Chrome and Meet [35]. We could not obtain the same statistics for Zoom-Chrome as it uses DataChannels instead of RTP MediaStream in WebRTC to transmit media. DataChannels statistics lack any video quality metrics and mostly contain data volume information. Obtaining application metrics is challenging for native clients. The Zoom Web API [40] does provide limited application performance (e.g., video resolution, FPS) at a per-minute granularity for the native client, but this granularity is insufficient to observe short-term quality fluctuations.[2] We thus limit our analysis in this section to only Meet and Teams-Chrome. We focus on a subset of metrics available from WebRTC Stats API that relate to video *quality* and *freezes*. These metrics are available at a per-second granularity during each call.

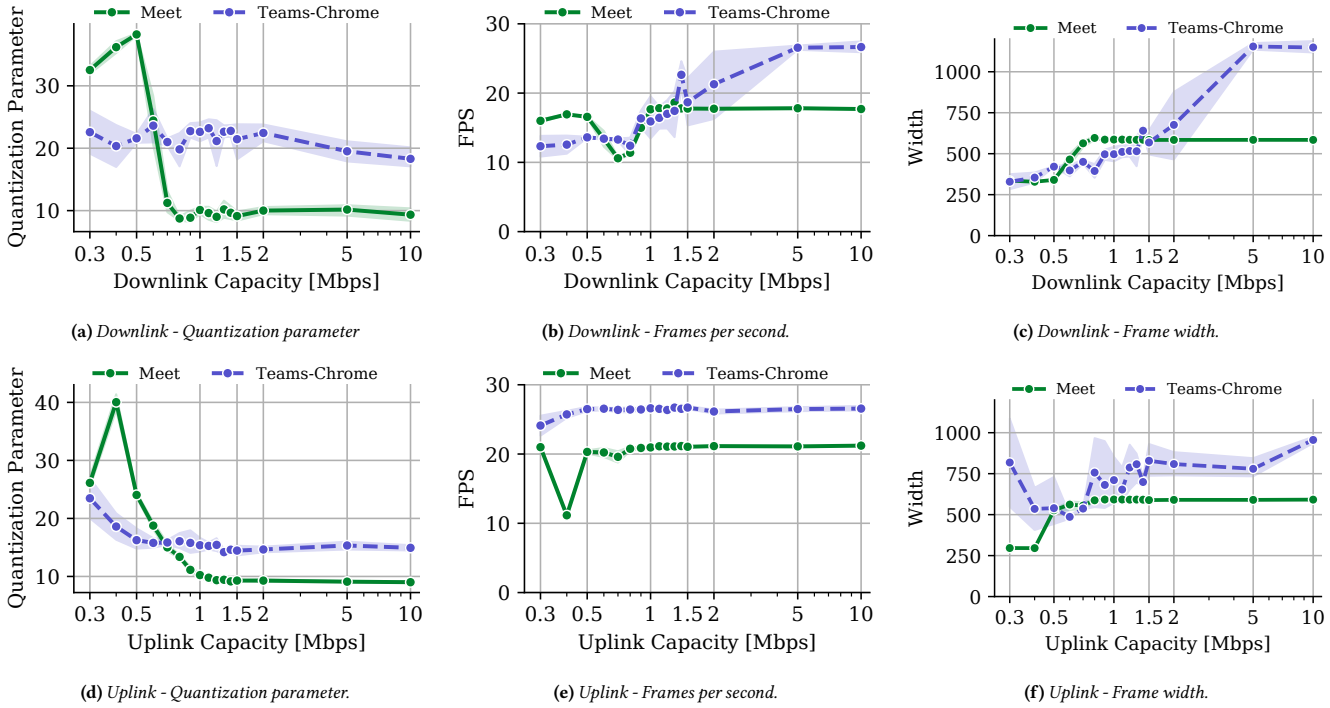
Video quality: VCAs adapt the video quality by adjusting the encoding parameters to achieve a target bitrate estimate provided by the transport. Ideally, VCAs can adjust one or more of the following three parameters:

- *frames per second* (FPS),
- *quantization parameter* used in video compression. It regulates the spatial detail saved while encoding the frames. A lower value implies better picture quality. Note that the values of quantization parameter are not comparable across different encoding standards, and
- *video resolution* indicated as the number of pixels in each dimension. In our analysis, we present a single dimension of the resolution, namely, its *width*.

We can obtain all three parameters from the WebRTC stats.

We first examine these parameters for different downstream capacities (see Figures 2a to 2c). Teams-Chrome simultaneously degrades all three video quality parameters as downstream capacity decreases. In the case of Teams-Chrome, we also observe variable behavior across multiple experiments under the same network capacity, as shown by the 90% confidence interval bands in the plot. Meet, on the other hand, behaves in a more consistent fashion: it

[2]The Zoom native client also provides a statistics tab with fine-grained measures of the same metrics which can be captured using image-based analysis. We consider this as a part of future work.



(a) *Downlink - Quantization parameter*

(b) *Downlink - Frames per second.*

(c) *Downlink - Frame width.*

(d) *Uplink - Quantization parameter.*

(e) *Uplink - Frames per second.*

(f) *Uplink - Frame width.*

Figure 2: *Video encoding parameters with 90% confidence intervals under downstream and upstream throughput constraints.*

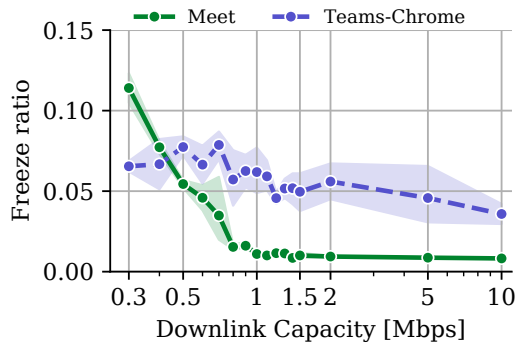
adjusts parameters differently based on capacity. Within the 0.7–1 Mbps range, Meet adapts the bitrate primarily by adapting frames per second, while keeping the quantization parameter and frame width similar to the quality levels in the absence of degradation. As capacity is further constrained, however, from 0.5–0.7 Mbps, both the frame width and quantization parameter degrade, whereas there is a simultaneous *increase* in the frames per second. Upon closer inspection of the per-second statistics in this operating range, we find that the relay server may switch to a lower quality copy of the stream it received via simulcast. Meet does not appear to reduce the bitrate any further by reducing the FPS for the low-quality stream—specifically, we observed a consistent frame rate even at bitrates of less than 0.5 Mbps It is not clear why the quantization parameter reduces from 38 to 33 at 0.3 Mbps in Meet.

We next analyze the effect of uplink capacity constraints on encoding parameters (see Figures 2d to 2f). Teams adapts to constrained throughput settings mainly by increasing the quantization parameter and reducing the frame width, while keeping the FPS almost constant. To our surprise, we found that the frame width, *increases* as uplink capacity is reduced to 0.3 Mbps. There seems to be no particularly good explanation for this effect, and (as we discuss in the next paragraph) it also results in video freezes, suggesting a poor design decision or implementation bug. Meet follows a similar trend by mostly increasing the quantization parameter until the upstream bandwidth decreases to 0.5 Mbps. At 0.4 Mbps, it also reduces frame width and the FPS.

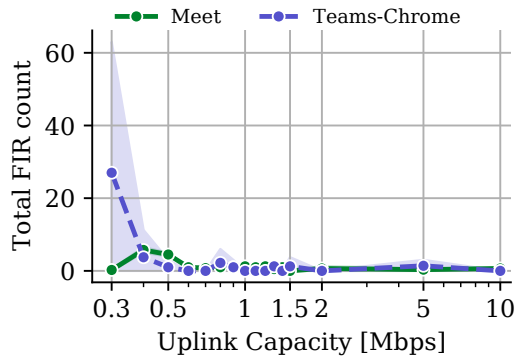
Video freezes: We also analyze the effect of constrained settings on video freezes. When constraining downlink capacity, we directly obtain the freeze duration from WebRTC stats. The WebRTC API assumes the video froze if the frame inter-arrival time > max (3δ, $\delta + 150ms$), where δ is the average frame duration. We normalize the freeze duration with the total call duration to obtain freeze ratio. Figure 3a shows the freeze ratio under different downstream capacities. The freeze ratio increases as the downlink bandwidth degrades. Meet has higher freeze ratio than Teams-Chrome with 10% freeze ratio at 0.3 Mbps. Interestingly, Teams-Chrome incurs freezes (a 3.6% freeze ratio) *even in the absence of throughput constraints*, suggesting again implementation problems or poor design choices.

When the uplink is constrained, we could not obtain freeze statistics from V1's logs, because the WebRTC stats provide freeze statistics only for the received video stream. Instead, we analyze the total count of Full Intra Requests (FIR) for the upstream video data. An FIR is sent if the receiver cannot decode the video or falls behind, likely due to frame losses. A low FIR count does not rule out freezes on the receiver, but a high count does indicate that video freezes are occurring. Figure 3b shows that the FIR count is particularly high for Teams-Chrome at uplink capacity below 0.5 Mbps. A high FIR count in Teams-Chrome may be triggered due to the sender sending high-resolution video, as shown in Figure 2f.

Takeaways: VCA requirements and performance under the same network conditions vary among tested VCAs. Further, the VCAs' average upstream utilization on an unconstrained link has implications for broadband policy. The FCC currently recommends a 25/3 Mbps minimum connection. The uplink in such a connection can be saturated by just two simultaneous 2-person Teams calls, potentially leading to degradation in the end-user experience.

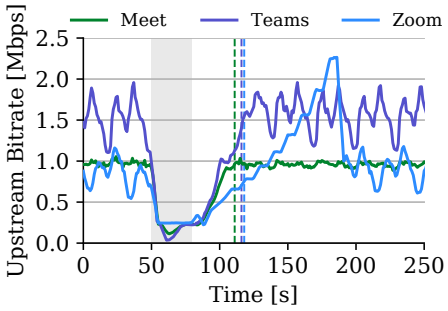


(a) *Downstream: Freeze ratio.*

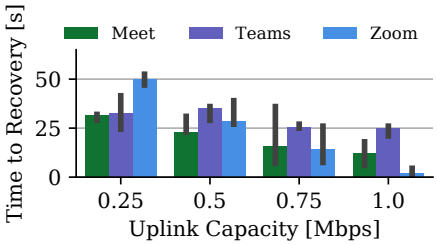


(b) *Full Intra Request (FIR) count.*

Figure 3: *Video freeze under downstream and upstream throughput constraints. The bands represent 90% confidence intervals.*



(a) *Average upstream bitrate over time. Grey region indicates period where uplink capacity is constrained to 0.25 Mbps. Vertical dotted lines indicate when the upstream bitrate has returned to the average bitrate before disruption.*



(b) *Time to recover from disruption to a given uplink capacity*

Figure 4: *VCA response to a 30-second reduction in uplink capacity.*

4 NETWORK DISRUPTIONS

Although all VCAs must handle network disruptions, they may do so in different ways. Designing applications to cope with disruptions, such as interruptions to network connectivity, has become even more pertinent over the past year as users have increasingly relied on broadband Internet access to use VCAs. In addition to experiencing periodic outages, home networks are susceptible to disruptions caused by temporary congestion along the end-to-end path, In this section, we aim to understand how VCAs respond to the types of disruptions that home Internet users might sometimes experience. We do this by studying the network utilization of the VCAs during and following disruptions in a controlled setting. We mainly focus on the time and the path the VCA takes to recover back to normal utilization, i.e., the utilization under unconstrained network. While this does not provide significant insight into the user QoE during disruption, using normal utilization as the baseline can help in understanding how fast a VCA recovers back to its nominal experience and the patterns of recovery.

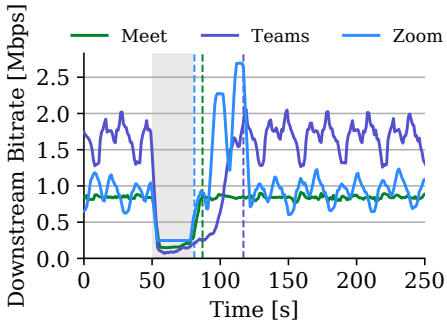
Method: We analyze how VCAs respond to temporary network disruptions during the call by introducing transient capacity reductions. Using the same setup as in the previous experiment, we initiate a five-minute VCA session between two clients, V1 and V2, both of which are connected to the Internet via a 1 Gbps link. One minute after initiating the call, we reduce the capacity between V1 and the router for 30 seconds, before reverting back to the unconstrained 1 Gbps link. We repeat each experiment four times. We conduct two sets of experiments: first disrupting the uplink, and then disrupting the downlink. We reduce capacity to the following levels: 0.25, 0.5, 0.75, 1.0 Mbps. We do not consider disruptions in capacity to levels of more than 1 Mbps because both Zoom and Meet's average utilization is below 1 Mbps.

4.1 Uplink Disruptions

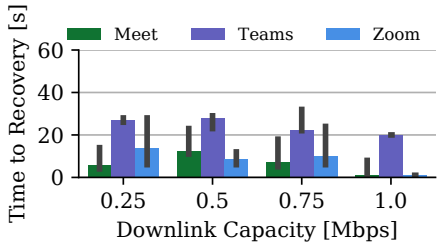
Figure 4a shows VCA upstream bitrate over the course of a call. Following a disruption, both the time to recovery and the characteristics of that recovery differ across VCAs. We quantify the time for VCA utilization to return to normal by defining a *time to recovery* (TTR) metric. We define TTR as the time between when the interruption ends and when the five-second rolling median bitrate reaches the median bitrate before interruption, also referred to as nominal bitrate.

Time to Recovery: Figure 4b shows how the extent of the disruption to uplink connectivity affects each VCA's time to recovery. The more severe the capacity reduction, the longer the VCAs need to recover.

Teams takes longer to recover even at less severe disruptions for two reasons: (1) the nominal bitrate of Teams is higher than Meet and Teams, (2) Teams initially increases the upstream bitrate slowly immediately after the interruption before increasing quickly back to normal (as shown in Figure 4a). Meet also observes a similar recovery pattern when the disruption drops capacity to 0.25 Mbps. However,



(a) *Average downlink bitrate over time. Grey region indicates period where downlink capacity is constrained to 0.25 Mbps. Vertical dotted lines indicate when the downstream bitrate has returned to the average bitrate before disruption.*



(b) *Time to recover from disruption to a given downlink capacity.*

Figure 5: *Response to a 30-second reduction in downlink capacity.*

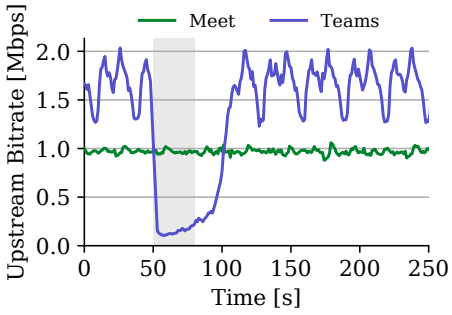


Figure 6: *Client 2 (V2) upstream bitrate. Grey region indicates when V1's downlink capacity is reduced to 0.25 Mbps*

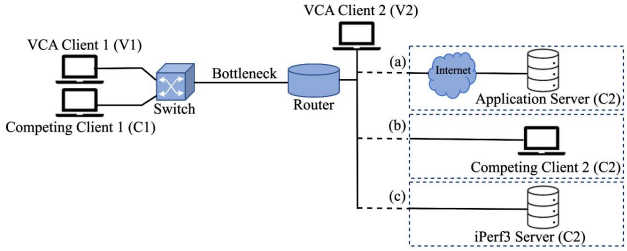


Figure 7: *Setup for competition experiments. When C1 is streaming Netflix or Youtube, it follows path (a). If C1 is a VCA, it follows path (b). When C1 run iPerf3, it follows path (c), which is within the University network.*

it recovers much faster for the case of less severe disruptions, mostly because its nominal bitrate is around 0.96 Mbps.

Recovery Patterns: While Meet and Teams follow a more rapid trend, Zoom's recovery is different: It takes the longest time to recover under severe disruptions. According to Figure 4a, Zoom follows a stepwise recovery, with an almost-linear increase immediately after interruption. It then enters a periodic-probing phase where it increases the sending rate, stays at it for sometime before increasing it again. The probing phase continues well above its nominal bitrate, before finally reducing the bitrate. Zoom does not return to a steady-state sending pattern until two minutes after the disruption, in the meantime sending at much higher rates. At first glance, such probing might appear to be bad design as it could introduce additional packet loss and delay harming both Zoom's own performance and the performance of other applications. We suspect, however, that Zoom may be using redundant FEC packets to gauge capacity similar to the FBRA congestion control proposed by Nagy et al. [25]. Thus, even if such behavior induces packet loss, the user performance may not suffer. Nevertheless, this inefficient use of the uplink, however, could disturb other applications on the same link, leading to a poor quality of experience for competing applications.

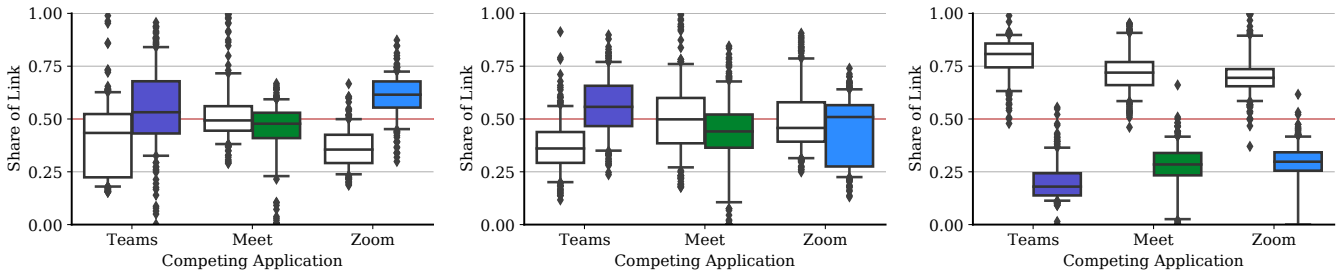
4.2 Downlink Disruptions

Figure 5b shows that Teams recovers more slowly than Meet and Zoom, always taking at least 20 seconds longer to return to the average rate, regardless of the magnitude of the interruption. Furthermore, Meet and Zoom recover much more quickly in this case compared to uplink interruptions. This can be explained by the way each VCA sends video. For all three VCAs, the clients communicate through an intermediary server. Thus, the recovery times depend on the congestion control behavior at the server, as well.

As mentioned in Section 3, Meet uses an encoding technique called simulcast, where the client encodes the same video at different quality levels and sends the encoded videos to an intermediate server. The server then sends the appropriate version based on the perceived downlink capacity at the receiver. The server can quickly switch between versions based on downstream capacity. This quick recovery is shown in Figure 5: Meet returns to its average rate in under ten seconds, regardless the severity of the interruption.

Similarly, Zoom uses *scalable video coding* when transmitting video [39]. Instead of sending several versions of varying quality, Zoom sends hierarchical encoding layers that, when combined, produce a high quality video. This allows V2 to send uninterrupted even when V1's downlink capacity decreases. The server can then recover faster by sending additional layers once the network conditions improve.

While the intermediary server for Zoom and Meet performs congestion control, in the case of Teams, this server acts only as a relay. During a Teams call, V2 will recognize V1 has limited downstream capacity and send at only the bitrate it knows V1 can receive. Once V1 has more available bandwidth, V2 must discover this: it must first probe the connection before returning to its average sending rate. Figure 6 illustrates how V2's sending rate to the intermediary server does not change during a Meet call, but drops below V2's downlink capacity during a Teams call, leading to the slow recovery.



(a) *Incumbent: Meet* **(b)** *Incumbent: Teams* **(c)** *Incumbent: Zoom*

Figure 8: *Share of uplink capacity used by VCAs in competition with VCAs on a 0.5 Mbps symmetric link. White box is the incumbent application*

Zoom's downstream and upstream bitrates remain at the level of available capacity during the disruption; Meet and Teams suffer more serious degradation. Zoom's efficient network utilization and response to disruption may be due to its encoding mechanisms, which allows it to effectively control the encoding parameters (i.e., SVC-encoded layers, FPS, resolution etc.) to match the target bitrate.

Takeaways: Modern VCAs are slow to recover from reductions in uplink capacity, with all three requiring more than 25 seconds to recover from severe interruptions to 0.25 Mbps. Only Teams is consistently slow to recover from disruptions to downlink capacity, even showing degradation in response to more moderate capacity reductions (e.g., to 1 Mbps). This result can be attributed to each VCA's media sending mechanism and how congestion control is implemented at the intermediate server for each VCA.

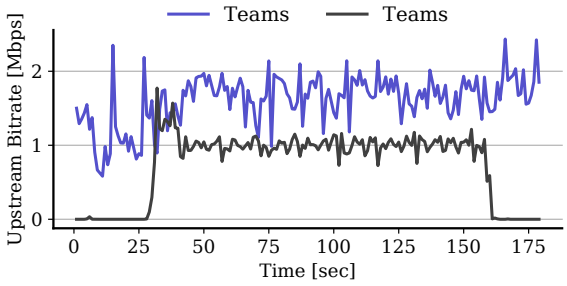
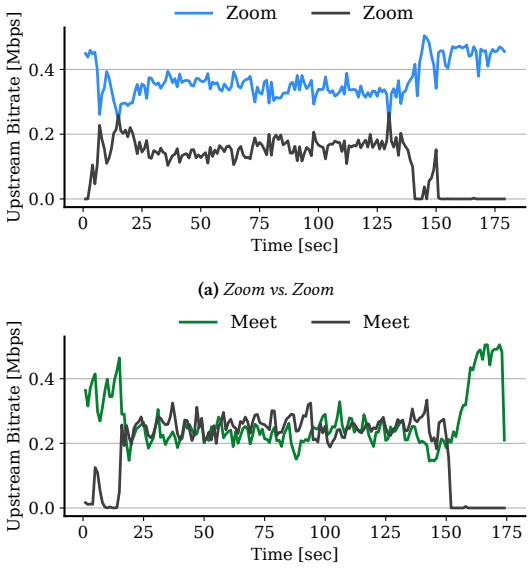


Figure 9: *A second Teams call does not reach its nominal uplink utilization when competing with an incumbent Teams call on a 3 Mbps uplink capacity. The black line is the competing application. The competing client's bitrate drops to 0 Mbps because the connection is terminated.*

5 COMPETING APPLICATIONS

With work and school continuing online, it is even more common to have multiple applications simultaneously using the home network, potentially leading to competition between any combination of VCAs, video streaming applications, or other popular applications. In this section, we measure how VCAs perform in the presence of other applications sharing the same bottleneck link. We focus on



(a) *Zoom vs. Zoom*

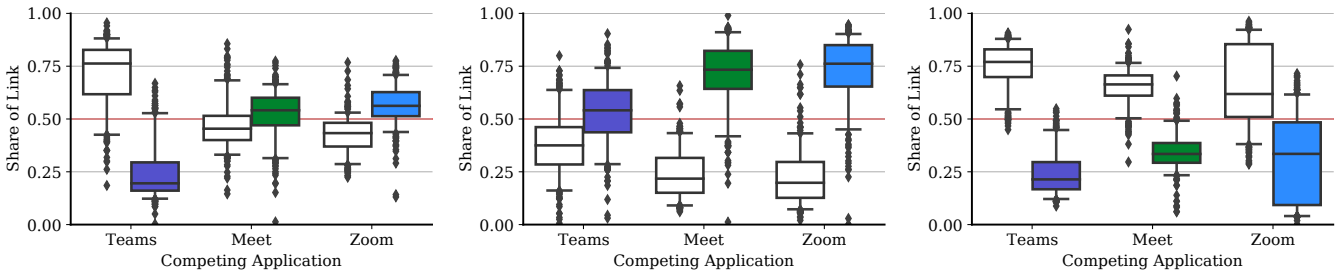
(b) *Meet vs. Meet*

Figure 10: *Upstream bitrate of VCAs in competition on a 0.5 Mbps capacity link. The black line is the competing application. The competing client's bitrate drops to 0 Mbps because the connection is terminated.*

link sharing with other VCAs, a single TCP flow (iPerf3), and two popular video streaming applications, Netflix and YouTube (which uses QUIC).

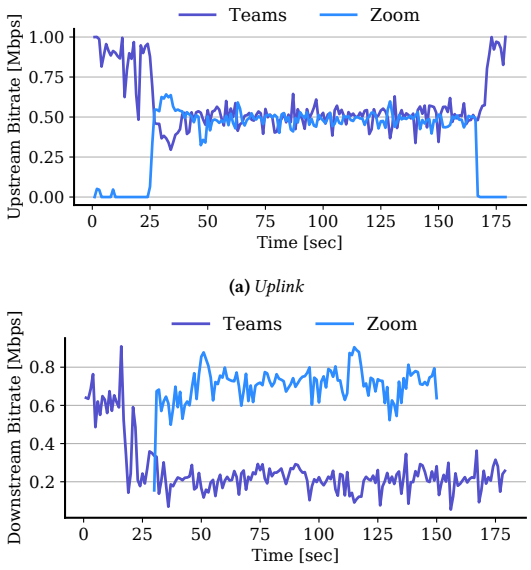
Method: As illustrated in Figure 7, the setup for these experiments differs slightly from earlier ones. Instead of connecting the client directly to the router, the two matched clients, V1 and C1, are connected to the router via a switch. The link capacity between the switch and the router is set on the router. For each test, V1 (the incumbent application) first establishes a VCA call with V2. Approximately 30 seconds later, C1 initiates the competing application, which lasts for two minutes. C1's counter-party, C2, depends on the type of competing application. If the competing application is a VCA call, then C2 is another laptop. If the application is an iPerf3[3] (TCP) flow, then C2 is a server on the same network. If C1 is Netflix or Youtube (launched on Chrome), then C2 is a server for the respective

[3]The iPerf3 server uses TCP CUBIC and is within the University network (average RTT 2ms).



(a) *Incumbent: Meet* **(b)** *Incumbent: Teams* **(c)** *Incumbent: Zoom*

Figure 11: *Share of downlink capacity used by VCAs in competition with other VCAs on a 0.5 Mbps symmetric link. White box is the incumbent application.*



(a) *Uplink*

(b) *Downlink*

Figure 12: *Comparison of Teams competition behavior for uplink and downlink on a 1 Mbps symmetric link. Teams is the incumbent application. In Figure 12a, the Zoom bitrate goes to 0 Mbps because the connection is terminated.*

service. After the competing application terminates, the incumbent application continues for an additional minute. We repeat each experiment 3 times, with link capacity varied symmetrically at {0.5, 1, 2, 3, 4, 5} Mbps.

5.1 VCA vs. VCA

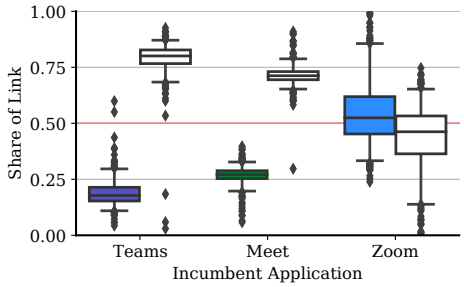
VCAs can achieve their nominal bitrate when the link capacity is 4 Mbps or greater, which is expected because the link capacity exceeds the sum of nominal bitrates of each VCAs. Figure A.1 demonstrate that most VCAs, except two competing Teams clients, can achieve their nominal uplink usage on the minimum uplink capacity recommended by the FCC (3 Mbps). In this section, we focus on the operating regime when the link capacity is lower, leading to competition between the VCAs. With only two competing clients, we use the proportion of the link shared as a metric to assess fairness. We call a VCA "aggressive" if it uses more than half of the link capacity under competition, and "passive" otherwise.

Considering the upstream direction, we observe differences in how each VCA shares the link. Figure 8 shows a box plot of the upstream share of an incumbent VCA and a competing VCA when the uplink capacity is 0.5 Mbps. We find that an incumbent Meet client shares the link fairly with a new Meet or Teams client but backs off when a Zoom client joins (see Figure 8a). The results are similar for Teams except it receives a slightly higher share while competing with Zoom compared to Meet. Interestingly, Zoom is highly aggressive, both as an incumbent and a competing application, using at least 75% of the link capacity in the case it is an incumbent client (see Figure 8c). In fact, Zoom's congestion control is not even fair to itself.

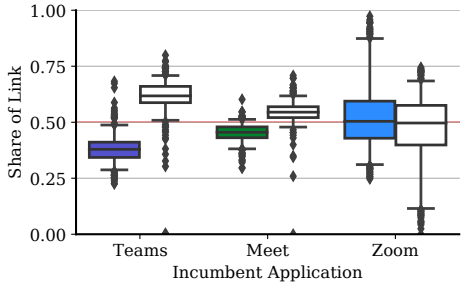
Figure 10a illustrates this effect more clearly, with the link sharing between two Zoom clients for a single experiment under 0.5 Mbps uplink capacity. We contrast this with the link sharing between two Meet clients in Figure 10b, where both sessions converge to their fair share of 0.25 Mbps. The results are similar for other uplink capacities, with aggressive applications leaving more room for new clients if they achieve their nominal bitrate.

We find similar results for Zoom and Meet when downstream capacity is constrained. However, we find that Teams is passive when sharing downstream capacity, backing off to all other VCAs, including other Teams flows. Figure 11b shows the link share of an incumbent Teams client compared to the competing VCA under 0.5 Mbps downlink capacity. Teams achieves only 20% of the link when sharing with Meet and Zoom. We observe similar behavior for other downlink capacities. Figure 12b illustrates how an incumbent Teams client shares the link with a Zoom client at 1 Mbps downlink capacity. Clearly the Teams client backs off to 0.2 Mbps. In contrast in the upstream direction, both Teams and Zoom converge to a near fair share of the link, as shown in Figure 12a. Additionally, the downstream throughput in Figure 12b for Teams degrades even before the Zoom call starts, likely because the competing client opens the Zoom landing page to initiate a call. This can lead to competing TCP traffic on the link and as we find in the next subsection, Teams is extremely passive when competing against TCP. The competing client's bitrate drops to 0 around $t = 168s$ in Figure 12a because the connection is terminated.

Finally, we observe that the competing Teams client does not reach its nominal utilization rate when competing with an incumbent Teams client on a 3 Mbps capacity uplink. Figure 9 shows that the second client has a far lower network utilization rate than the incumbent. We found in Section 3 that Teams has a nominal uplink



(a) *Downlink.*



(b) *Uplink.*

Figure 13: *iPerf3 link sharing with VCAs on a 2 Mbps capacity link.*

usage of 1.4 Mbps, whereas in this case the competing client uses only 1 Mbps for the majority of the call. The FCC definition of broadband internet is 25/3 Mbps but it is clear that a 3 Mbps uplink capacity is not sufficient to enable two Teams calls to reach their nominal utilization.

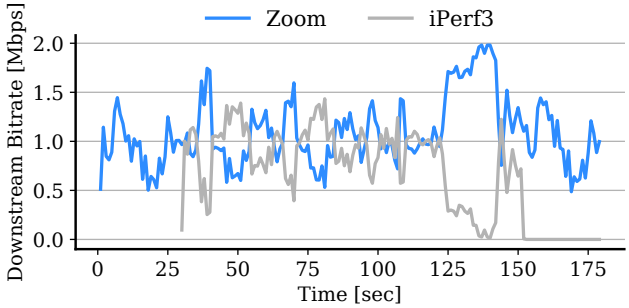
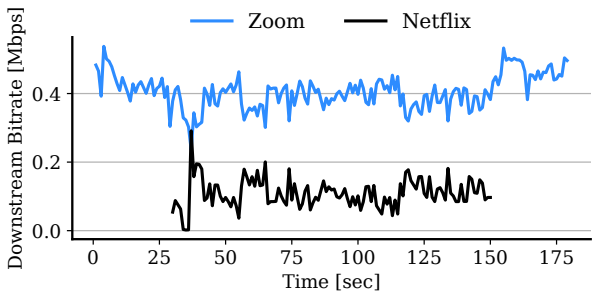


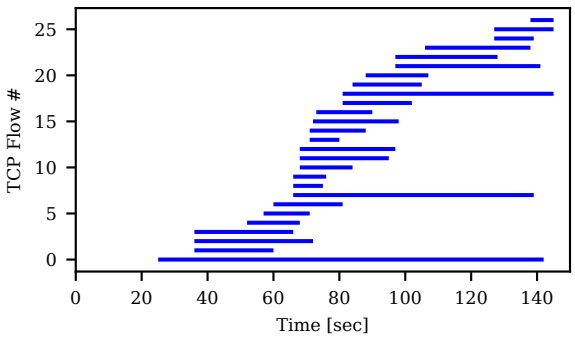
Figure 14: *Zoom probing can adversely affect competing applications.*

5.2 VCA vs. TCP

We now compare how the three VCAs compete with a 120 seconds long TCP flow from iPerf3. In a home network, a long TCP flow can be created by file download or upload. We find that Zoom is highly aggressive, especially at low uplink and downlink capacity, consuming more than 75% of the bandwidth under a symmetric 0.5 Mbps link. Meet is TCP-friendly in the uplink direction but not in the downlink, consuming 75% of the bandwidth at 0.5 Mbps downlink (figures omitted due to lack of space). Teams, on the other hand, is passive and backs-off against a TCP flow in both upstream and



(a) *Downstream bitrate for Zoom and Netflix.*



(b) *TCP connections opened by Netflix.*

Figure 15: *Netflix and Zoom competition on a 0.5 Mbps capacity link.*

downstream even at high link capacity. Figure 13a and 13b show how each VCA shares a 2 Mbps downlink and uplink with iPerf3, respectively. Teams is able to use 37% in uplink and only 20% in the downlink even at 2 Mbps. The low upstream throughput for Teams with a TCP flow is particularly surprising as it was able to achieve its fair-share when competing against (more aggressive) Meet and Zoom, as shown in Figure 8b. At 2 Mbps, both Meet and Teams achieve their nominal bitrate allowing iPerf3 to use rest of the link. Clearly, all the VCAs are not TCP CUBIC-friendly with Meet and Zoom being highly aggressive and Teams being highly passive.

Anomalous Zoom Bursts: In the previous section, Figures 4a and 5a showed how Zoom can send bursts of data for an extended period of time following a network disruption. Figure 14 shows how this behavior is also exhibited when in competition with iPerf3. At about 125 seconds, Zoom's increased sending rate causes iPerf3 to abruptly lower its utilization. This confirms that the temporary bursts in Zoom, likely for inferring available bandwidth, can adversely affect other applications on a scarce link.

5.3 VCA vs. Video Streaming

We also compared each VCA's link share against two video streaming applications, Netflix and YouTube, which consume significant downstream bandwidth. YouTube uses QUIC, a UDP-based transport protocol, which can be TCP-friendly depending on some configuration values [10]. Both Meet and Zoom are very aggressive when competing against video streaming applications, using over 75% of the link capacity. In contrast, Teams uses less than 25% while competing with YouTube and Netflix at 0.5 Mbps. Figure 15a shows

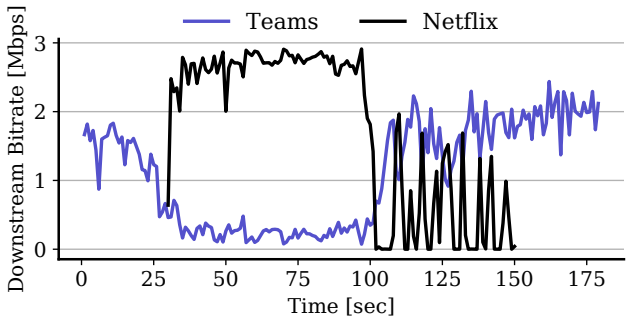


Figure 16: *Teams in competition with Netflix on 3Mbps downlink.*

this effect, when a Netflix client competes with an incumbent Zoom client at 0.5 Mbps downstream capacity. Zoom achieves an average throughput around 0.4 Mbps while Netflix struggles to reach more than 0.1 Mbps. We observe this effect despite the fact that Netflix is known to use multiple TCP connections, especially when capacity is limited. Figure 15b shows the TCP connections opened by Netflix during the 120-second experiment. In total, Netflix opens 27 TCP connections, each with more than than 100 Kbits of data; at one point, it opens 11 parallel TCP connections. Despite this behavior, opening multiple TCP connections does not improve link sharing for Netflix while competing with Zoom.

Figure 16 illustrates how Teams's utilization is affected by the steady state behavior of a competing Netflix stream. When the Netflix stream is initiated, it uses the available bandwidth to fill its buffer before switching to a steady-state ON-OFF regime. During the *buffering* phase of Netflix, the network utilization of Teams drops significantly. Once Netflix reaches steady state around $t = 100s$, Teams's average utilization increases but it is still unstable. When Netflix increases utilization during the ON phase, Teams's utilization drops. As Netflix switches to OFF phase, Teams's utilization increases slowly and drops again when Netflix switches back to ON phase. As a result, Teams does not return to stable utilization until after the Netflix stream ends.

6 CALL MODALITIES

People now increasingly rely on video conferencing for remote work, school, and even large virtual conferences (like IMC '21!), particularly during the COVID-19 pandemic, which has shifted many in-person activities to virtual forums. These settings involve many participants, which raises questions concerning performance and network utilization of VCAs in settings involving multiple users. We study network utilization under two prominent call modalities:

(1) the number of participants in a call and
(2) the viewing mode.

We consider two viewing modes that are common across all three VCAs: *speaker* mode wherein a specific user's video is pinned on the call and *gallery* mode, in which all participants' video are shown on the screen. Each experiment consists of a 2-minute call with n users and specific viewing mode. We vary the number of clients in the call from two to eight, across both gallery and speaker modes. We perform five experiment for each (number, viewing mode) combination; we log the network utilization of Client C1 for each call. Although it is difficult to evaluate these applications for hundreds of simultaneous users, we can nevertheless explore trends in utilization and performance for a smaller number of users and observe various utilization trends. Larger-scale experiments could be a possible area for future work.

6.1 Number of Users

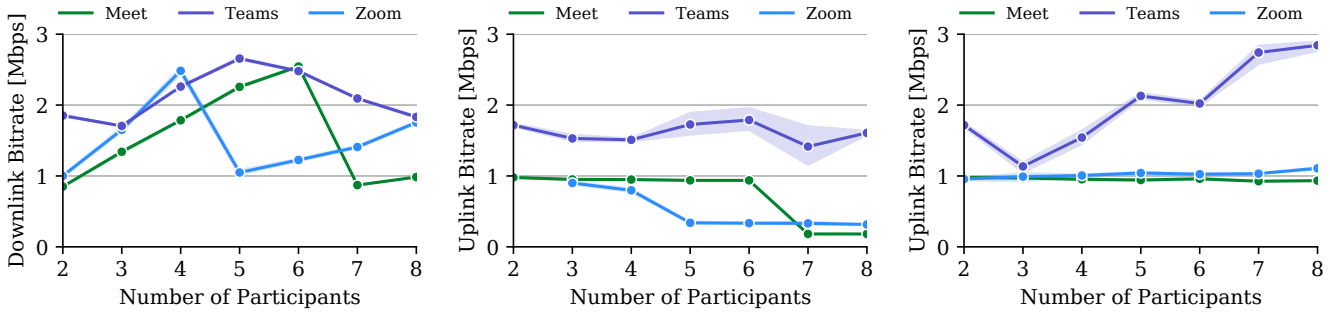
To explore the effect of the number of users on utilization, we fix the viewing mode to *gallery*, which is the default viewing mode in all of the VCAs. Figures 17a and 17b show the average downstream and upstream network utilization respectively for different numbers of participants. Total downstream utilization depends on both the number of video streams and the data rate of each stream. Utilization in both directions typically *decreases* as the number of participants increases in a Meet or Zoom call. In the case of Zoom, the uplink utilization drops from 0.8 Mbps to 0.4 Mbps as number of participants increases from 4 to 5. For Meet, the reduction happens at n = 7, from 1 Mbps to 0.2 Mbps. Both Meet and Zoom have tiled-screen display with each user displayed in a separate tile. As the number of users increases, the tile size shrinks to accommodate all the users on the fixed size screen. For instance, Zoom uses a 2×2 grid for 4 participants; switching to 5 participants creates a third row of video feeds making each individual video feed smaller. The sender uses this opportunity to reduce the resolution of transmitted video, leading to reduction in upstream utilization.

We observe a similar reduction in downstream utilization at 5 and 7 participants for Zoom and Meet, respectively (Figure 17a). However, there are also notable differences when compared to the uplink utilization. For instance, the downlink utilization for Google Meet increases from 1.25 Mbps to 2.5 Mbps when the number of participants increases from 3 to 6, while upstream utilization stays mostly constant. The trend is similar for Zoom as the number of participants increases beyond than 5.

Teams does not exhibit these trends: upstream utilization remains almost constant as the number of participants changes. Downstream utilization increases until five participants and drops as more participants join the call. Teams has a fixed 4-tile layout on Linux. It thus displays only a *subset* of participants if a call has more than four participants which may explain why the sending rate does not change, particularly as the number of videos and the frame size may not change significantly with more users. It is, however, not clear why the downstream utilization decreases in calls with more than participants; this phenomenon deserves more exploration.

6.2 Viewing Mode

For all three VCAs, viewing a user's video in speaker mode leads to greater uplink consumption on *that* user's network as compared to gallery mode. Putting C1 on the speaker mode enlarges its tile size on other users' screen. The sender streams the video at a high resolution to provide a high better user experience, thus leading to increase in uplink utilization compared to when C1 was not pinned by other users. Zoom and Meet consistently send at 1 Mbps when all clients pin C1's video, regardless of the number of participants. Note



(a) *Downlink traffic of client whose video is viewed in gallery mode.*

(b) *Uplink traffic of client whose video is viewed in gallery mode.*

(c) *Uplink traffic of client whose video is pinned by all other participants (speaker mode).*

Figure 17: *Network utilization in different viewing modes. The bands represent 90% confidence intervals.*

that only one client needed to put C1 on speaker for this behavior. Each client can decide to pin any client independently of others.

The behavior of Teams differs from Meet and Zoom in this regard, as well. C1's uplink utilization continues to increase from 1.25 Mbps with 3 participants to 2.9 Mbps participants when it is put on speaker mode. We checked if the increase could be attributed to Teams communicating with multiple destinations (e.g., with each user separately). However, we observe that all of the traffic was directed to a single server. It is not clear what contributes to the increase in traffic for Teams but this clearly leads to inefficient network utilization, especially when compared to Zoom and Meet.

Takeaways: Each participant's video layout affects client's network utilization. Calls with more participants can *decrease* upstream and downstream utilization for each participant, depending on how the VCA displays participant video. The settings of one participant (e.g., pinning a video, using speaker mode vs. gallery mode) can also affect the upstream utilization of *other* participants.

7 RELATED WORK

VCA measurement: Some of the early VCA measurement work has focused on uncovering the design of the Skype focusing on its streaming protocols [2], architecture [12], traffic characterization [6], and application performance [14]. More recent work has included other VCAs and streaming contexts [1, 36, 37]. Xu et al. [36] use controlled experiments to study the design and performance of Google+, iChat, and Skype. The work is further extended to include performance of the three services on mobile video calls [37].

Closest to our work is work by Jansen et al. [15], Nistico et. al [26], and Chang et al. Jansen et al. evaluate WebRTC performance using their custom VCA under controlled network conditions [15]. Emulating similar network conditions, we consider performance of commercial and more recent VCAs that widely used for education and work. Even between tested VCAs using WebRTC, namely Meet and Teams-Chrome, we find significant performance differences, likely due to different design parameters (e.g., codecs, default bitrates). Nistico et al. [26] consider a wider range of recent VCAs, focusing on their design differences including protocols and architecture. Our work provides a complementary performance analysis for a subset of the VCAs studied by them. We use the insights from their work to explain the differences among VCAs' network utilization and performance under similar streaming contexts. Finally, Chang et al. [9] study the effects of geography, client device, and number of users on streaming experience in Meet, Teams, and Webex by emulating calls on the cloud. Our controlled experiments provide a complimentary method to evaluate the application performance. We also focus on response to disruptions and fairness of the VCAs.

VCA congestion control: Several congestion control algorithms have been proposed for VCAs. These algorithms rely on a variety of signals such as loss [13], delay [8], and even VCA performance metrics [33] for rate control. For instance, Google Congestion Control [8], also implemented in WebRTC, uses one-way delay gradient for adjusting the sender rate while SCReAM [17] relies on both loss and delay along with TCP-like self-clocking. While the VCAs may use one or more of these variants, the exact implementation of the algorithm and parameter values vary and is proprietary. In this work, we study the efficacy of the VCA congestion control in the case of transient interruptions and background applications. A recent study by Sander et al. evaluates Zoom's congestion control along these dimensions [29]. Our work observes similar results for Zoom and also analyses more VCAs, including their fairness to each other and other popular internet applications, namely YouTube (QUIC-based) and Netflix (TCP-based).

8 LIMITATIONS AND FUTURE WORK

Generalizability to other VCA contexts: This paper focuses on understanding the performance and network utilization of three popular video conferencing applications mostly over a Linux-based setup in a controlled environment. Clearly, real-world VCA performance is impacted by a variety of factors including end-user location, device platform, access network technology, the VCA backbone infrastructure, and streaming mechanisms (e.g., encoding or bitrate adaptation). In addition, Figure 1c shows that Teams has higher utilization in the native application than in browser. This indicates that VCA performance in web clients may not generalize to the native applications. While this study is a first step in isolating the

impact of some of these individual factors on VCA performance, a continued evaluation over a variety of streaming contexts is required for a thorough understanding, especially as the VCAs evolve over time. We believe the methods in this paper could be extended to other VCAs and across different device platforms. For instance, our experiment automation framework using PyAutoGUI can be easily extended to other VCAs and other desktop device platforms including *macOS* and *Microsoft Windows*. We demonstrate this by repeating the experiments in Section 3 for macOS on Meet, Zoom, and Teams-Chrome (see Figures A.3 and A.2). We have made our experiment code publicly available so that others can extend this work to these contexts [21].

Finally, VCAs are increasingly used on mobile devices which differ from laptop or desktop clients in terms of operating systems, constrained resources, and even network conditions in some cases (e.g., while using cellular network). We believe similar automation frameworks need to be developed for mobile devices to draw conclusions for these platforms.

Application performance metrics: Analyzing application performance statistics, especially the audio and the video metrics individually, can shed more light on the VCA behavior and user quality of experience. While we demonstrate the impact of network impairments on video performance metrics through a subset of our results using WebRTC statistics in Section 3.2, it is challenging to obtain application performance metrics for all VCAs, especially native clients. Future research could explore the methods from related work, such as using annotated audio and video [36] or network traffic captures [11, 22] to infer application metrics.

Performance under other network conditions: Our work mainly focuses on understanding the impact of throughput-based impairments on VCA performance. We have conducted experiments in-lab, where we can control for other network factors such as latency, jitter, and packet loss. With this step complete, it is important to study the impact of these other network factors. For instance, future work could examine the impact of variations in the packet loss or latency over the congestion control for different VCAs.

Finally, we acknowledge that real-world networks are more complex and dynamic than in-lab networks. Our results, therefore, are not necessarily representative in all cases. At the same time, it can be challenging to emulate real-world networks in a controlled environment. A future study could collect VCA performance metrics and network metrics from real-world users. Alternatively, these analyses could be performed through aggregate traffic on enterprise networks (e.g., university networks). These organizations could also leverage the consoles or APIs provided with most enterprise VCA subscriptions.

Further exploration of VCA design: Some of our results reveal unusual VCA behavior that we found difficult to explain and deserve further exploration. Examples include: (1) Some VCAs exhibit different behavior depending on if the competition exists in the upstream or downstream direction, (2) Teams' network utilization does not follow a clear pattern, with respect to the number of participants or their video layout; and (3) Meet shows unusual encoding behavior (specifically, increased frame rates) when network capacity is very low.

9 CONCLUSION

We analyzed the network utilization and performance of three major VCAs under diverse network conditions and usage modalities. Our results show that VCAs vary significantly both in terms of their resource utilization and performance, especially when capacity is constrained. Differences in behavior can be attributed to different VCA design parameter values, congestion control algorithms, and media transport mechanisms. The VCA response also varies depending on whether constraints exist on the uplink or downlink. Different behavior often arises due to both differences in encoding strategies, as well as how the VCAs rely on an intermediate server to deliver video streams and adapt transmission to changing conditions. Finally, the network utilization of each VCA can vary significantly depending on the call modality. Somewhat counterintuitively, calls with more participants can actually reduce any one participant's upstream utilization, because changes in the video layout on the user screen ultimately lead to changes in the sent video resolutions. Future work could extend this analysis to more VCAs, device platforms, and network impairments. In general, however, performance can begin to degrade at upstream capacities below 1.5 Mbps, and some VCAs do not compete well with other TCP streams under constrained settings, suggesting the possible need for the FCC to reconsider its 25/3 Mbps standard for defining "broadband".

Acknowledgments. This research is funded in part by National Science Foundation awards CNS-2124393 and CNS-1704077. We thank the reviewers and our shepherd, Dave Choffnes, for comments that improved the paper.

REFERENCES

[1] A. Azfar, K.-K. R. Choo, and L. Liu. Android mobile voip apps: a survey and examination of their security and privacy. *Electronic Commerce Research*, 16(1):73–111, 2016.

[2] S. A. Baset and H. Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. *arXiv preprint cs/0412017*, 2004.

[3] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman. The secure real-time transport protocol (srtp), 2004.

[4] J. Bienik, M. Uhrina, M. Kuba, and M. Vaculik. Performance of h. 264, h. 265, vp8 and vp9 compression standards for high resolutions. In *2016 19th International Conference on Network-Based Information Systems (NBiS)*. IEEE, 2016.

[5] Bitag. 2020 pandemic network performance. https://bitag.org/documents/bitag_report.pdf, 2021.

[6] D. Bonfiglio, M. Mellia, M. Meo, N. Ritacca, and D. Rossi. Tracking down skype traffic. In *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*. IEEE, 2008.

[7] D. Bonfiglio, M. Mellia, M. Meo, and D. Rossi. Detailed analysis of skype traffic. *IEEE Transactions on Multimedia*, 2008.

[8] 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*, pages 1–12, 2016.

[9] H. Chang, M. Varvello, F. Hao, and S. Mukherjee. Can you see me now? a measurement study of zoom, webex, and meet. In *Proc. of ACM IMC*, 2021.

[10] R. Corbel, S. Tuffin, A. Gravey, X. Marjou, and A. Braud. Assessing the impact of quic on network fairness. In *2nd International Conference on Communication and Network Protocol (ICCNP 2019)*, 2019.

[11] M. Dasari, S. Sanadhya, C. Vlachou, K.-H. Kim, and S. R. Das. Scalable ground-truth annotation for video qoe modeling in enterprise wifi. acm. In *IEEE IWQoS*, 2018.

[12] S. Guha and N. Daswani. An experimental study of the skype peer-to-peer voip system. Technical report, Cornell University, 2005.

[13] M. Handley, S. Floyd, J. Padhye, and J. Widmer. Tcp friendly rate control (tfrc): Protocol specification. Technical report, RFC, 2003.

[14] T. Hoßfeld and A. Binzenhöfer. Analysis of skype voip traffic in umts: End-to-end qos and qoe measurements. *Computer Networks*, 2008.

[15] B. Jansen, T. Goodwin, V. Gupta, F. Kuipers, and G. Zussman. Performance evaluation of webrtc-based video conferencing. *ACM SIGMETRICS Performance Evaluation Review*, 2018.

[16] J. Jiang, R. Das, G. Ananthanarayanan, P. A. Chou, V. Padmanabhan, V. Sekar, E. Dominique, M. Goliszewski, D. Kukoleca, R. Vafin, et al. Via: Improving internet telephony call quality using predictive relay selection. In *ACM SIGCOMM*, 2016.

[17] I. Johansson and Z. Sarker. Self-clocked rate adaptation for multimedia. *draft-johansson-rmcat-scream-cc-05 (work in progress)*, 2015.

[18] C. Lab. Zoom uses a bespoke extension of RTP. https://citizenlab.ca/2020/04/move-fast-roll-your-own-crypto-a-quick-look-at-the-confidentiality-of-zoom-meetings/, 2020.

[19] Q. Liu, Z. Jia, K. Jin, J. Wu, and H. Zhang. Error resilience for interactive real-time multimedia application, 2019. US Patent 10,348,454.

[20] S. Liu, P. Schmitt, F. Bronzino, and N. Feamster. Characterizing service provider response to the covid-19 pandemic in the united states. *arXiv preprint arXiv:2011.00419*, 2020.

[21] K. MacMillan, T. Mangla, J. Saxon, and N. Feamster. Automating VCA Calls. https://github.com/kyle-macmillan/vca-imc-21, 2021.

[22] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura. eMIMIC: Estimating HTTP-based Video QoE Metrics from Encrypted Network Traffic. In *Proc. of IEEE/IFIP TMA*, 2018.

[23] C. S. Media. LOOKING BACK, LOOKING FORWARD: What it will take to permanently close the K–12 digital divide. https://www.commonsensemedia.org/about-us/news/press-releases/the-us-k-12-digital-divide-has-narrowed-but-must-close-to-eliminate, 2021.

[24] Microsoft. Virtual camera does not work on macOS Teams. https://docs.microsoft.com/en-us/microsoftteams/troubleshoot/teams-on-mac/virtual-camera-doesnt-work-macos, 2021.

[25] M. Nagy, V. Singh, J. Ott, and L. Eggert. Congestion control using fec for conversational multimedia communication. In *Proceedings of the 5th ACM Multimedia Systems Conference*, pages 191–202, 2014.

[26] A. Nistico, D. Markudova, M. Trevisan, M. Meo, and G. Carofiglio. A comparative study of rtc applications. In *2020 IEEE International Symposium on Multimedia (ISM)*. IEEE, 2020.

[27] ReadTheDocs. PyAutoGUI: Application automation. https://pyautogui.readthedocs.io/en/latest/, 2020.

[28] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, et al. Sip: session initiation protocol, 2002.

[29] C. Sander, I. Kunze, K. Wehrle, and J. Rüth. Video conferencing and flow-rate fairness: A first look at zoom and the impact of flow-queuing aqm, 2021.

[30] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. Rfc3550: Rtp: A transport protocol for real-time applications, 2003.

[31] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, et al. Rtp: A transport protocol for real-time applications, 1996.

[32] Selenium. Selenium: Web browser automation. http://www.seleniumhq.org, 2017.

[33] V. Singh, J. Ott, and I. D. Curcio. Rate-control for conversational video communication in heterogeneous networks. In *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–7. IEEE, 2012.

[34] E. T. Tester. Video conferencing market share. https://www.emailtooltester.com/en/blog/video-conferencing-market-share/, 2021.

[35] W3. WebRTC Stats. https://www.w3.org/TR/webrtc-stats/, 2020.

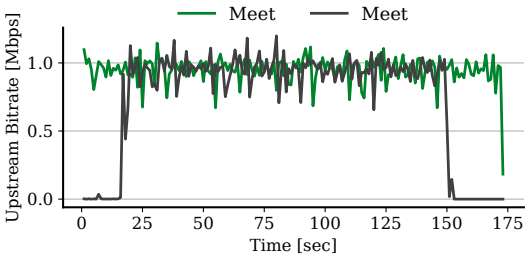
[36] Y. Xu, C. Yu, J. Li, and Y. Liu. Video telephony for end-consumers: Measurement study of google+, ichat, and skype. In *IMC*. ACM, 2012.

[37] C. Yu, Y. Xu, B. Liu, and Y. Liu. "can you see me now?" a measurement study of mobile video calls. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 1456–1464. IEEE, 2014.

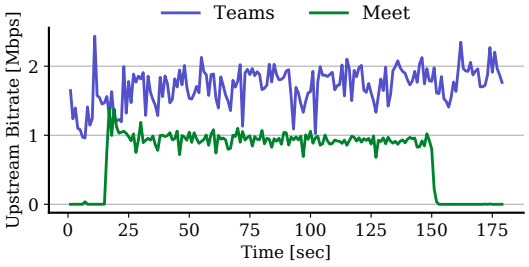
[38] J. Zhu, D. Jones, and S. Webster. Testing Bandwidth Usage of Popular Video Conferencing Applications, November 2020.

[39] Zoom. Here's How Zoom Provides Industry-Leading Video Capacity. https://blog.zoom.us/zoom-can-provide-increase-industry-leading-video-capacity/, 2019.

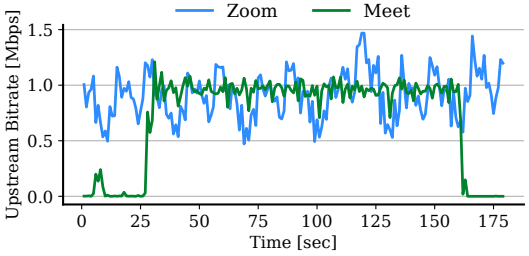
[40] Zoom. Zoom API. https://marketplace.zoom.us/docs/api-reference/zoom-api/, 2021.



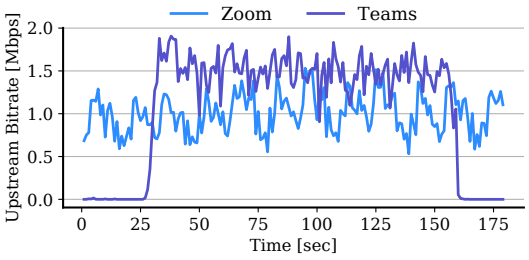
(a) *Meet vs. Meet*



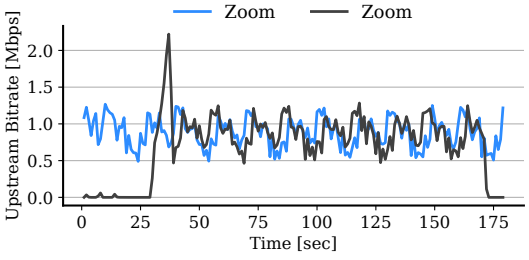
(b) *Teams vs. Meet*



(c) *Zoom vs. Meet*

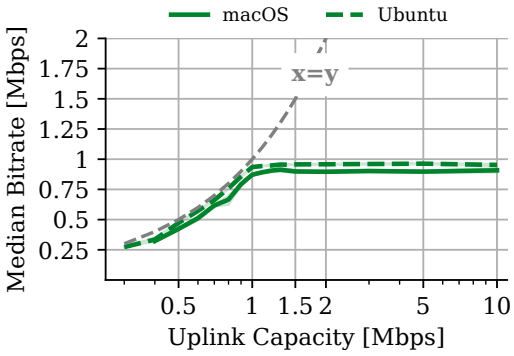


(d) *Zoom vs. Teams*

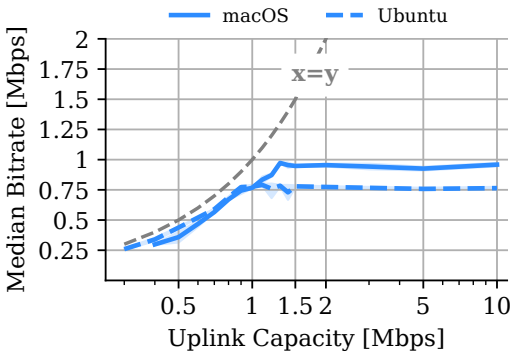


(e) *Zoom vs. Zoom*

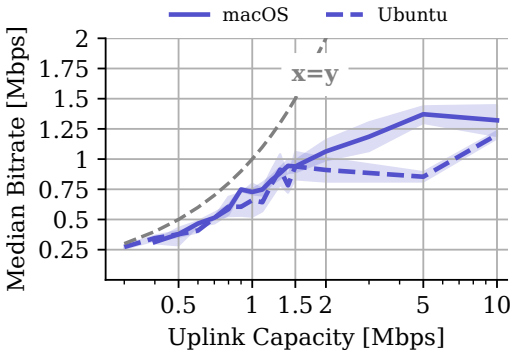
Figure A.1: *VCA vs. VCA on a 3 Mbps symmetric connection.*



(a) *Meet*

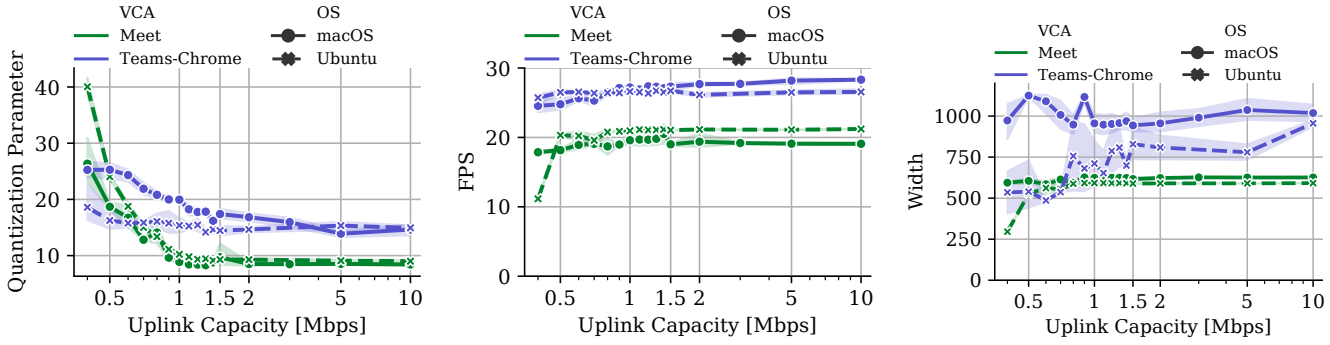


(b) *Zoom*



(c) *Teams-Chrome*

Figure A.2: *Comparison of uplink network utilization between macOS and Ubuntu under different shaping levels. The bands represent 90% confidence intervals.*



(a) *Uplink - Quantization parameter.* **(b)** *Uplink - Frames per second.* **(c)** *Uplink - Frame width.*

Figure A.3: *Comparison of video encoding parameters between macOS and Ubuntu with 90% confidence intervals under upstream throughput constraints.*

A VCA VS. VCA COMPETITION

We include the following figures to demonstrate that most VCAs can achieve their nominal uplink utilization rate when in competition with the other VCAs on a 3 Mbps uplink capacity, the minimum uplink capacity recommended by the FCC (25/3). The only exception is two competing Teams calls, which we discuss in Section 5.

B STATIC EXPERIMENTS - REPETITION ON MAC OPERATING SYSTEM (MACOS)

Here, we demonstrate the extensibility of our automation framework for macOS. More specifically, we repeat the experiments from Section 3 with the uplink shaped on a device running macOS. We use a 2017 MacBook Air with 8GB 1600 Mhz DDR3 memory, 1.8 GHz Dual-Core Intel Core i5 processor, and Intel HD Graphics 6000 1536 MB Graphics. The laptop is running macOS Version 10.15.7. The MacBook Air initiates a call with a Linux client with the same specifications as outlined in Section 2.2. We use Zoom Version 5.7.6 and Chrome Version 92.0.4515.159. We test the browser version of Teams and the native client of Zoom and Meet.

There were also some platform-specific differences while trying to automate the data collection. The macOS does not support v4l2loopback; instead we use Open Broadcaster Software or OBS (Version 27.0.1) to feed in the same talking-head video used throughout our experiments. In addition, we only include the Teams-Chrome client for Teams and omit the native client. This is because the current version of the Teams client on macOS does not support virtual camera [24].

Results: Figure A.2 shows the uplink network utilization on the macOS device with uplink network utilization on Ubuntu shown as a baseline. The overall trends are similar as on Ubuntu. There are a few differences between the two platforms: Zoom has a higher nominal utilization on macOS (0.96 Mbps) than Ubuntu (0.78 Mbps). Teams-Chrome also has a higher nominal utilization on macOS than Ubuntu. Meet has a similar uplink utilization, using 0.89 Mbps on Mac and 0.95 Mbps on Ubuntu. Some of these differences could be due to updated VCA version itself, while others could be attributed to the differences in the operating system.

We next report application performance metrics (see Figure A.3) by logging the WebRTC stats API as in Section 3.2. The Ubuntu statistics under the same shaping levels are also reported as a baseline. We find that Meet adapts to the constrained throughput setting mainly by increasing the quantization parameter. On the other hand, Teams-Chrome simultaneously adapts both the sent frames per second and the quantization parameter of the encoded video. Interestingly, there are some minor differences compared to Ubuntu. For instance, the sent frame width as well as the FPS for Meet does not change on macOS at low capacity region (0.4-0.5 Mbps), while it decreased in the case of Ubuntu. This is despite the fact that the browser-based Meet can ideally use similar codebase across operating systems. With the current experiment setup, it is not clear, however, if the root cause of the difference can be attributed to the operating system. This is because Meet may have updated in between the two set of experiments were conducted. In future work, we plan to repeat the experiments around similar time to understand any operating system-specific differences.