Performance and Network Utilization of Popular Video Conferencing Applications

Authors

Abstract

Video conferencing applications (VCAs) have become a critical Internet application, particularly as users worldwide now rely on them for work, school, recreation, and telehealth. The COVID-19 pandemic has spurred unprecedented growth in the use of VCAs, with some Internet service providers seeing a three-fold increase in video conferencing traffic over an eight-month period in 2020. It is important to understand the resource requirements of different VCAs and how these VCAs perform under different network conditions, including: how much "speed" (specifically, upstream and downstream throughput) a VCA needs to support high quality of experience, 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., number of participants) affects utilization. We study these questions for three modern (and increasingly popular) VCAs: Zoom, Google Meet, and Microsoft Teams. Answers to these questions differ substantially depending on VCA. First, the steady-state 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 45 seconds to recover to steady state; recovery time differs across VCAs. Differences in proprietary congestion control also create unfair bandwidth allocations: in constrained bandwidth settings, one Zoom video conference can consume more than 75% of the available bandwidth when competing with another video conference application (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.

1 Introduction

COVID-19 has made video conferencing applications (VCAs) both common and necessary. People now regularly use VCAs for work, school, and telehealth, as activities have shifted from in-person to online. The success of remote communication indicates that VCAs will continue to play a critical role during and after the pandemic. However, VCAs are unique among internet applications and their increased use has altered the makeup of internet traffic.

VCAs differ from other popular types of internet activity in several ways. Most notably, VCAs have far greater uplink utilization. Video streaming and internet browsing, two dominant sources of internet traffic, [receive the bulk of their traffic and send very little]. VCAs, however, send a video stream on the uplink. Further, VCAs do not benefit from network optimizations used in video streaming. Video streaming applications can "buffer" by fetching video ahead of time, then sporadically downloading when necessary. VCAs can not download video ahead of time because the video occurs in real-time. As a result, VCAs require a consistent and low-latency connection.

The earliest studies on VCAs focused on Skype, the dominant application in the VCA landscape at the time. These works sought to understand Skype from both a design and performance perspective. As the VCA landscape evolved over time, later work shifted their attention to newer VCAs and the protocols they used.

While significant research and measurement studies have been conducted on VCAs in the past, both the VCAs and how we use them has since changed. Several new VCAs (e.g. Microsoft Teams, Google Meet) have been introduced, while existing VCAs (e.g. Zoom) have undergone a renewal. In light of these developments, it is important to investigate how differing VCA design choices impact application performance and network consumption.

The continuing reliance on VCAs makes understanding these network requirements precisely crucial. In this paper, we conduct a comparative analysis of three popular VCAs: Google Meet, Microsoft Teams, and Zoom. We can determine these requirements by measuring VCA performance and consumption under realistic network conditions, including low bandwidth availability and other activity on the same network. Our goal is to understand the following:

1. How VCAs performance varies under different static network conditions.
2. How VCAs respond to temporary drops in bandwidth availability.

3. How VCAs interact with competing applications on a constrained link.

4. How usage modalities (number of participants, speaker vs. gallery viewing) affect network consumption.

Answering the first question gives us insight into the minimum network requirements for each VCAs.

2 Background and Approach Overview

This section provides a background on video conferencing applications. We also describe our basic measurement methodology.

2.1 Background

Most video conferencing applications use Real-time Transport Protocol [6] or its variants [**srtp**, **zoom**]. RTP typically runs over UDP. It uses two other protocols in conjunction, Session Initiation Protocol (SIP) to establish connection between clients and RTP Control Protocol (RTCP) to share performance statistics and control information during a call. While VCAs are similar in terms of their basic architecture, they differ across the following dimensions:

use P2P or a relay server. STUN servers used to by pass NAT. TURN servers are used for multi-party calls or when a direct connection cannot be established. In some cases, using a relay server can also improve performance [**via**].

Networking mechanisms: Congestion control. Error correction mechanisms.

Application-layer parameters: Encoding standards. The audio data is encoded using constant-bitrate. Video is encoded using standard codecs such as VP9 and H.264 with the bitrate controlled based on the underlying network conditions.

Video conferencing architecture: Direct vs relay-based. Also differ in the capabilities of the relaying server.

Thus, the logic for congestion control is implemented in the application-layer. Audio and video data are usually transmitted over separate RTP channels. T RTCP is the control channel that provides end-to-end feedback about the network.

2.2 VCA selection and performance metrics

Our goal in the paper is to do a comparative analysis of performance of VCAs under different network conditions and calling context. While there are many VCAs available, this study focuses on three popular VCAs, namely Zoom, Google Meet, Microsoft Teams. These VCAs have been used extensively over the past year for work and education purposes []. Therefore, it becomes critical to understand the performance and network utilization of the VCAs. Our experiment methodology, however, can be easily extended to other VCAs. We plan to make the code and the associated data public.

Performance metrics: We mainly focus on the following application metrics.

- **Network bitrate**
- Video resolution
- Frames per second
- Freeze count
- Freeze duration
- Jitter-buffer delay

Most of our analysis is using network bitrate. We consider other application performance metrics when they can be extracted from Google Chrome[1]. Our analysis shows that there is a strong correlation between network bitrate and application performance. This is intuitive as real-time streaming is characterized by low latency and thus any network interruptions usually lead to degradation in application performance.

2.3 Measurement Methodology

The measurement setup consists of several components: Hardware

- matched laptops for nominal flows
  - All Linux; caveats – may not be the most-maintained apps
  - What laptops, for the many-laptop tests?
- turris router for control of single shaped link
- private iperf server on university network
- traffic control scripts
- autogui; selenium for browser
  - Important that we have the screen up – see e.g., the tweet, but maybe chase the paper, about the struggle of getting the machines to actually render flows being important.
  - client vs browser
  - sw versions?
- pcap, iperf logs

[1]chrome://webrtc-internals

- webrtc
  - difference between meet and zoom
- Zoom API access (& limitations?)
- control over sockets?

3 Approach Overview

The experiments are conducted in a lab environment but aim to simulate realistic VCA usage. To that end, we use the following setup throughout the experiments.

Hardware: All data is collected on two identical Dell Latitude 3300 laptops running 20.04.1 Ubuntu. The laptops have a resolution of the laptops is 1366 × 768 pixels. The laptops have a wired connection to a Turris Omnia 2020 router and access an unconstrained gigabit link on the University of Chicago network.

Automating Calls: It is crucial that the automated calls are as close to an actual call as possible, as any deviation from real call use may warp results. We take several steps to recreate the in-call process.

All VCAs are tested in their native client unless otherwise indicated. Note, the application's "native client" refers to the Chrome browser for Meet and the desktop applications for Zoom and Teams. In the first set of experiments, we compare browser vs. client utilization for Teams and Zoom. In-browser tests will be specified by *Teams-chrome* or *Zoom-chrome*.

We use the PyAutoGui and AutoGui Python packages to automate joining and leaving calls. We also use Selenium browser to bypass the CAPTCHA before joining a Zoom call in Chrome browser. In this case, Selenium is not run headless, but is run exactly as it would be in Chrome browser. All experiments are conducted with the laptop lid open and the application window maximized. Run otherwise, the applications may adapt their behavior. For example, if the application window is minimized or the lid is closed, the download bitrate may decrease.

Finally, we use a 1280 × 720 pre-recorded talking-head video as the video source during calls to both replicate a real video call and ensure consistency across experiments. Using the webcam feed is a non-starter because without movement, the VCAs compress the video and ultimately send at a much lower rate than during a normal call.

Network Shaping: We simulate network conditions by configuring traffic control using `tc` on the laptop and router for uplink and downlink shaping, respectively. During the competition experiments, all shaping occurs at the router.

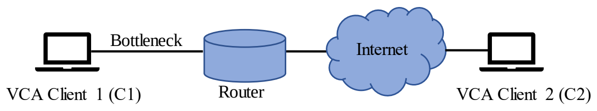


Figure 1: Setup for static experiments

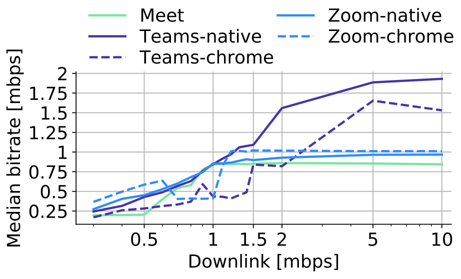


Figure 2: Downlink bandwidth vs network bitrate

Inter-Laptop Communication: The workflow is controlled with by a wrapper script on laptop 1. Laptop 1 establishes a socket connection with laptop 2 to control automation on laptop 2.

Competing Flows:

4 Static Network Conditions

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

Methodology: We conduct two set of experiments with downlink shaped in the first set and uplink shaped in the second set. We consider the following shaping levels: {0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 2, 5, 10} Mbps. Each experiment consists of a 2.5-minute call between C1 and C2 under a specific shaping level. We perform 5 repetitions for each shaping level. We also include the browser clients for Zoom and Teams, referred to as Zoom-browser and Teams-

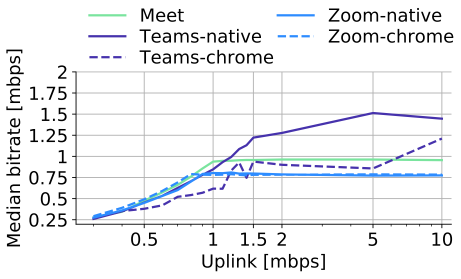


Figure 3: Uplink bandwidth vs network bitrate

browser, respectively. This is done to understand if there are any platform-related differences.

4.1 Link capacity and network performance

Figure 2 shows the median received network bitrate averaged over 5 runs for different downlink shaping levels. We find differences among VCAs in terms of their bitrate configuration and bitrate adaptation leading to different performance under same network conditions. For instance, the peak utilization for Teams-native is 1.9 Mbps whereas it is only 0.96 for Zoom-native and 0.8 Mbps for Meet. The received bitrate also reflects how different VCAs utilize the network especially under constrained links. Interestingly, we find that Zoom-native has the highest network utilization while Meet has the lowest (only 50% at 0.5 Mbps) for downlink capacity lower than 0.75 Mbps. Finally, we also observe differences between Teams-browser and Teams-native under same network conditions with the former having lower bitrates than the latter. For instance, at 1 Mbps shaping level, the received bitrate is only 0.5 Mbps for Teams-browser and 0.8 Mbps for Teams-native. This suggests that implementation of the VCA can differ across platforms leading to different network and application behavior.

We next analyze the median sent network bitrate for different uplink capacity (see Figure 3). As expected, we find similar differences in terms of the maximum bitrate among the VCAs and between browser and native clients for Teams as observed in the downlink shaping experiment. However, the network utilization is higher, especially for Meet, when the upstream link is constrained as compared to downlink. Thus, there seems to be a discrepancy in bitrate adaptation when the upstream is constrained. We explore this discrepancy in more detail in Section 5.

4.2 Link capacity and application performance

Here we describe how the link shaping level impacts application performance metrics. However, obtaining application metrics can be challenging for VCAs. We rely on WebRTC stats available on Google Chrome to obtain statistics for Teams-browser and Meet. We could not obtain the same statistics for Zoom-browser as it used different transport channels to transmit media, i.e., datachannels instead of the RTP MediaStream [**zoom_uses_datachannel**, **webrtc_stats**] as it may provides more flexibility (e.g., flexible encoding standards) to Zoom. However, video quality statistics are not exposed anymore through the WebRTC stats API. Thus, we limit this analysis to only Meet and Teams-browser for this section.

Application metrics and bitrate adaptation: VCAs can ideally adapt the video bitrate by adjusting one or more of the following three parameters: i) frames per second (*FPS*), ii) *quantization parameter* used in image compression, and iii) *video resolution* indicating the number of pixels in each dimension. We indicate resolution with one of the dimension, i.e., *frame width* in our analysis. Figure 4 shows the above parameters for Meet and Teams-browser under different downlink shaping levels.

We find that Teams-browser simultaneously degrades all three parameters as the downlink becomes more constrained. There is also significant variance across multiple repetitions under the same network bandwidth as shown by the 90% confidence interval bands in the plot. Meet, on the other hand, follows a more consistent trend. In the 0.7-1 Mbps region, the bitrate is controlled mostly by adapting the FPS while keeping the quantization parameter and frame width similar to the nominal levels. However, after 0.7 Mbps, both *resolution* and *quantization parameter* degrade whereas there is an increase in the FPS. Below 0.5 Mbps, the frame width and the FPS stabilize while the quantization parameter actually reduces. It is not clear as to why Meet chose to reduce the quantization parameter after 0.5 Mbps and increase the FPS after 0.8 Mbps.

We also plot the statistics related to video freezes during the call as obtained from the WebRTC stats. A freeze is assumed to occur if the frame inter-arrival is either greater than $3 \times$ average frame duration or $150ms$ + average frame duration. Figure 4d and 4e show the freeze ratio and number of freezes per second for Meet and Teams-browser. We find that both freeze ratio and freezes per second increase as the downlink bandwidth degrades. For Meet, the degradation is more severe than Teams-browser with 10% freeze ratio at 0.3 Mbps.

Figure **??**

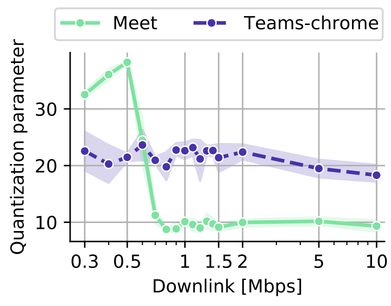
Takeaway: VCA performance under same network conditions varies.

5 Temporary Interruptions

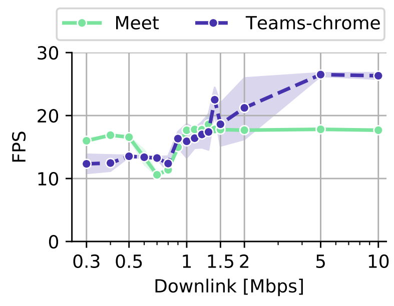
In this section, we discuss how VCAs respond to temporary bandwidth drops during the call.

Methodology: Using the same setup as Figure 1, a VCA call is initiated between two clients, C1 and C2, both of which sit behind unconstrained links. Once the VCA reaches its nominal consumption, we reduce the available bandwidth between C1 and the router for 30 seconds, before reverting back to an unconstrained link.

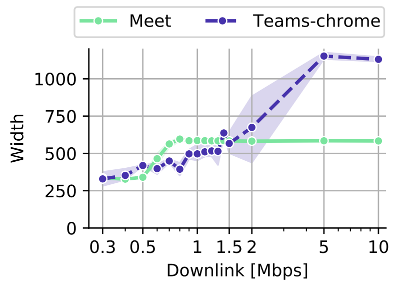
We introduce the metric *time to recovery* (TTR) in order to compare how the different VCAs respond to the interruption. We define TTR as the time between when the interruption ends and when the 5 second rolling median bitrate



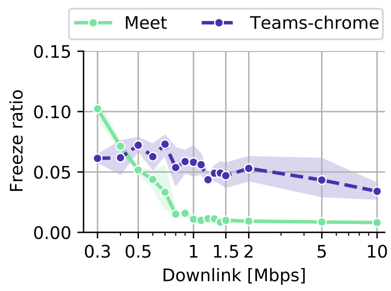
(a) Quantization parameter



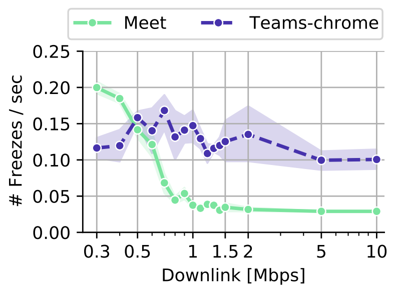
(b) Frames per second



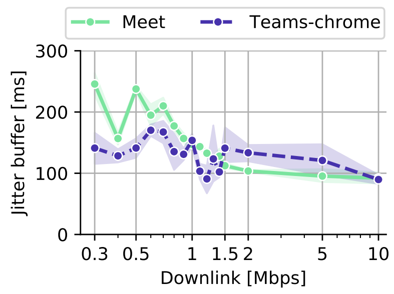
(c) Frame width



(d) Freeze ratio

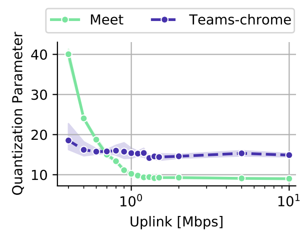


(e) Number of freezes per second

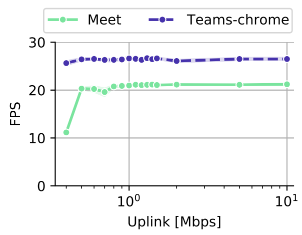


(f) Jitter buffer delay

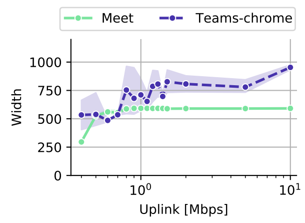
Figure 4: Downlink bandwidth and application performance metrics



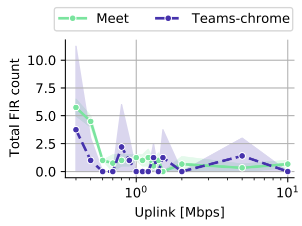
(a) Quantization parameter



(b) Frames per second



(c) Frame width



(d) FIR Count

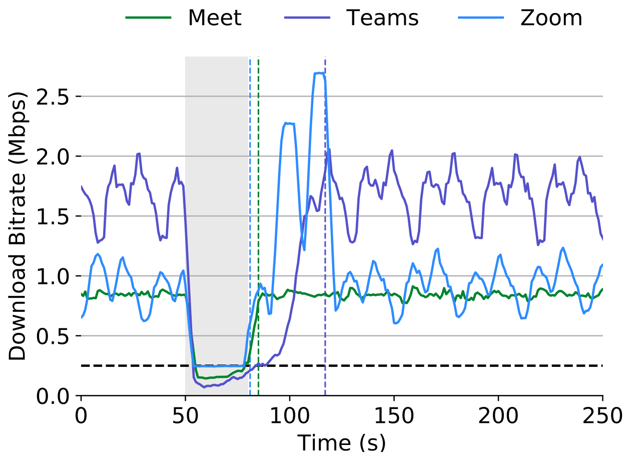
Figure 5: Uplink bandwidth and application performance metrics

reaches the default bitrate. In other words, how long it takes the VCA performance to return to normal following the interruption.

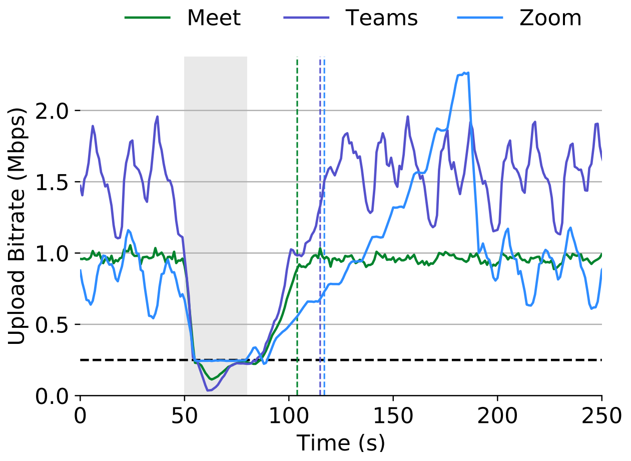
Results: The VCAs differ greatly in their response to interruptions, with some recovering much faster than others. Focusing first on download shaping, it is clear that Teams recovers much slower than Meet and Zoom, always taking at least 20 seconds longer to return to the nominal rate, regardless of the magnitude of the interruption. This can be explained by the way each VCA sends video.

In all three VCAs, C1 and C2 communicate through an intermediary server. Meet uses an encoding technique called *simulcast*, where the client encodes two versions . The server then sends the appropriate version based on the perceived downlink capacity at the receiver. This way, a sender transmitting on an unconstrained link does not have to adjust its sending and can rely on the server to send the appropriate version. The server can then quickly switch between which version to send to the receiver based on the feedback it receives. This quick recovery is clearly illustrated in 6a, in which Meet returns to its nominal rate in under 10 seconds, depending on the severity of the interruption.

Similarly, Zoom uses *multi-bitrate encoding* when transmitting video. Instead of sending several versions of varying



(a) While Meet and Zoom recover quickly, Teams probes the connection before slowly returning to the nominal rate



(b) All 3 VCAs require at least 20 seconds to recover from the interruption

Figure 6: Throughput over time. Gray region indicates with the link was lowered to 0.25 Mbps

quality, Zoom sends many ”layers” that, when superimposed, produce a high quality video.

While the intermediary server for Zoom and Meet does congestion control, it only acts as a relay for Teams. So during a Teams call, C2 will recognize C1 has limited downlink capacity and adjust its behavior, sending at only the bitrate it knows C1 can handle. Once C1 has more available bandwidth, however, C2 must first probe the connection before returning to its nominal sending rate. Figure **??** illustrates how C2's sending rate does not change during a Meet call, but drops below the shaping threshold during a Teams call, leading to the slow recovery.

While Teams has the highest nominal bitrate out of the three VCAs, its recovery to Zoom's (0.75Mbps) and Meet's (0.8Mbps) nominal bitrate, is still the slowest. Looking at Figure 6a, Teams's recovery follows a seemingly cubic trajectory; slowly probing the connection then rising back.

Turning now to upload shaping, we observe longer recovery times for all three VCAs. There is a clear trend that the more severe the upload shaping, the longer the VCAs need to recover. While Meet and Teams have a fairly direct recovery trend, Figure 6b shows how Zoom slowly increases the sending bitrate, going well above the nominal rate, before finally dropping back to steady state.

6 Competing Flows

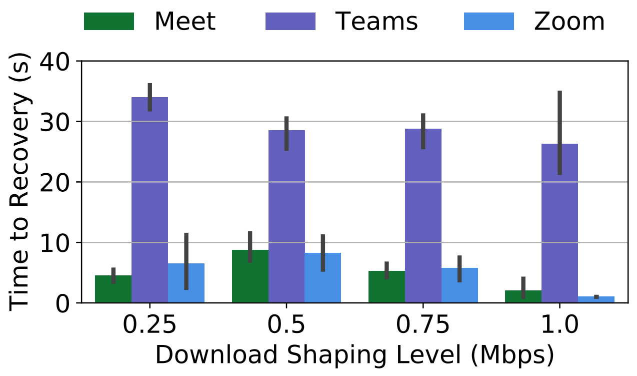
To test how VCAs handle fairness on a constrained link, we launch a competing application on the same network. We define three sets of competing applications: VCAs, video streaming, and file transfer. We use Meet, Teams, and Zoom for the VCAs, Youtube and Netflix for video streaming, and iPerf3 TCP for file transfer. The competing VCAs are initiated as described above. We play the same gaming video and the same movie on Youtube and Netflix, respectively, for every experiment using `xdg-open`. Finally, iPerf3 is started as usual on the command line. Two fundamental questions:

1. are the apps fair WRT each other, WRT traditional flows (TCP), and other applications (YouTube, NetFlix)
2. for two flows, what link capacity is required for full performance?

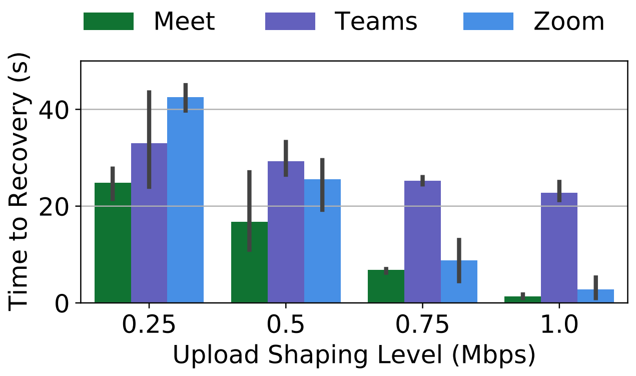
[*Jamie: part of what feels weird to me, in using download flows, is that these are SO MUCH lower than what most people by, that they feel irrelevant. If we had upstream, these same numbers would not be unreasonable, for the "popular discourse" on broadband thresholds.*]

Methodology.

The experiment is built within the framework already described. The distinguishing feature of this measurement is that the constrained link is between a switch and an Open-WRT Router (which constrains the link). The router has a gigabit link to the Internet. The competing flows are initiated from two matched consumer laptops, connected to the switch over unconstrained, gigabit links. We consider three forms of competitive flows: a standard TCP flow through



(a) Meet and Zoom recover quickly from downlink interruptions, while Teams takes consistently longer.



(b) Time to recovery increases with the severity of the uplink interruption

Figure 7: The time to recovery is the time needed to return to the nominal rate following a drop in bandwidth

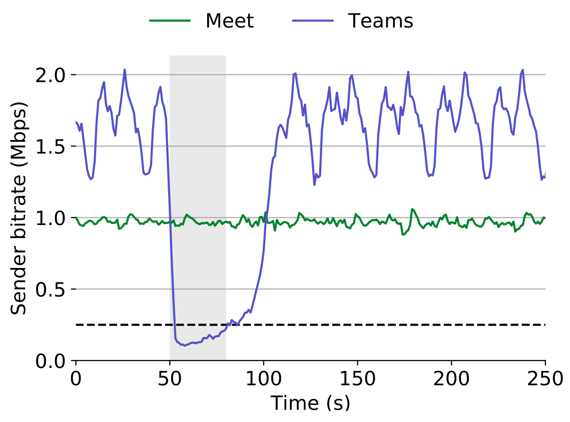


Figure 8: C2 sending rate.

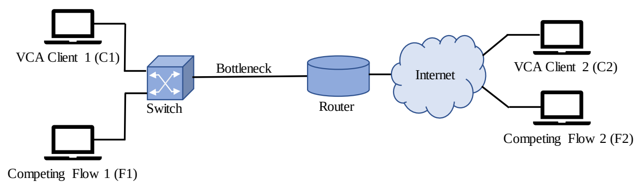


Figure 9: Setup for competition experiments

iperf3 with cubic congestion control, other video conferencing applications, and common video streaming services. The "counter-parties" for these flows are, respectively, a dedicated server on the university network, consumer laptops, and industry servers. In contrast to previous experiments, we focus on the native versions of Zoom and Teams clients. [*Jamie: Awkward: Meet is native in Chrome.*]

Each single experiment lasts for 4 minutes. The "nominal flow" is established first, and and the competing flow follow 30 seconds later. The competing flow lasts for 2.5 minutes. The experiment concludes with 1 minute with the nominal flow alone. This setup is illustrated in Figure 10, with the time series for a six experiments between Zoom and Netflix. Each configuration of nominal and competing flows and link constraint is repeated five times. The capacity constraints are to 0.5, 1, 2, 3, 4, and 5 Mbps. Bitrates are measured from packet captures.

From these flows we construct two variables to reflect fairness and performance. Fairness is represented as the nominal flow's share of the constrained link. Performance is the flow's share of its bitrate as measured on a fully unconstrained call (see Table **??**). [*Jamie: The fully-unconstrained values are important benchmarks that we should measure carefully. I know Tarun quoted these. But we should establish these and put them in a table early on.*] Figures 12-14 display results for Teams, Meet, and Zoom respectively.

Each application in "competition" with itself, uses half the link. Competition is in evidence primarily at the low end of the domain, with the most severe constraints. With weaker constraints, most applications reach their nominal

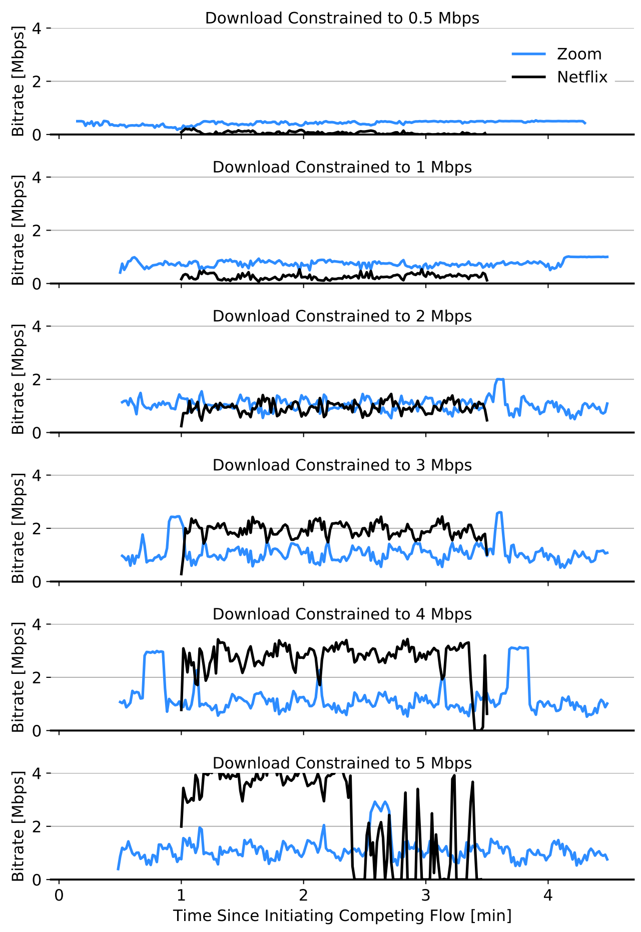


Figure 10: Time series of bitrate for Zoom in competition with a Netflix flow, at different link capacity. [*Jamie: timing offset bug*]

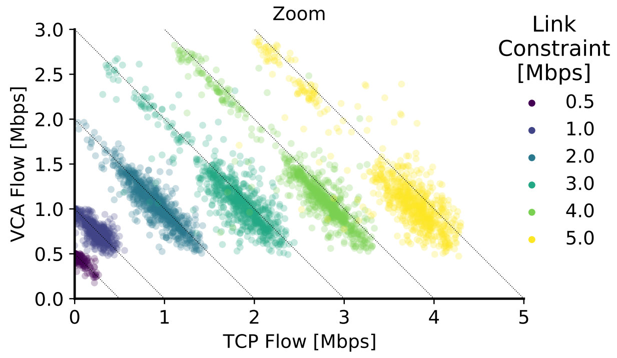


Figure 11: Competition between Zoom and an iperf3 TCP flow. [*Jamie: kind of pretty, but not obviously useful*]

bitrates and the ratios in the upper panels simply show the ratio of their bandwidth demands. For example, the low VCA shares with respect to iperf at weak constraint simply illustrates the "inexhaustible" demand of a TCP flow, whereas link share's of Zoom vs Meet or Meet vs Zoom at link capacity of 5 Mbps (0.4 or 0.6) simply reflect the nominal bandwidths.

[*Jamie: ambiguity: NetFlix or YouTube buffering, vs slow-start from teams*]

At the low end, Zoom competes aggressively and effectively with all other flows: it consumes nearly the entire link against Meet (95%), and 80% of the link against Teams. Both Teams and Meet fare worse on constrained links.

The lower panels of the figures show the link bandwidth required for the VCAs' consumption to converge to their nominal levels. Again, Zoom is quick out of the gate, and achieves over 90% of nominal link bitrate, for constraints weaker than 2 Mbps. On the other hand, Teams does not achieve full bitrate until [*Jamie: we gotta run higher..*] Depending on the flow, Meet achieves full performance when the shared link has capacity greater than 3 Mbps, except for the (unlimited) TCP flow.

[*Jamie: Zoom competing with meet nominal is flat, whereas the inverse is not. These aren't exactly the same – there's an "incumbency" advantage – but worth noting.*]

[*Jamie: Add notes on performance, e.g., through Meet, though I think that the preceding is most of the story.*]

7 Call parameters and consumption

In this section, we analyze the impact of different usage modalities such as the number of persons in the call, viewing mode, and device type on the network consumption.

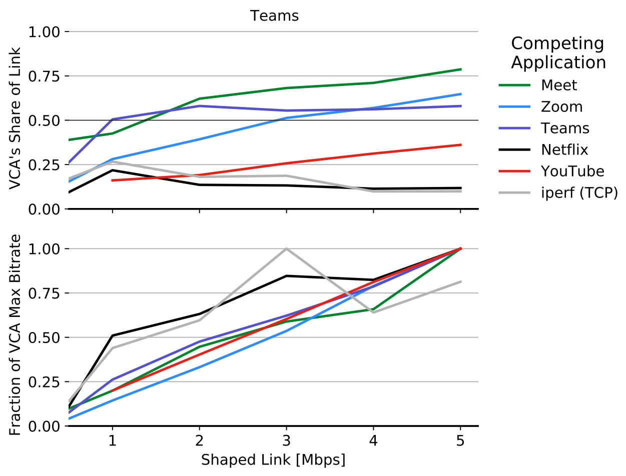


Figure 12: Link share and share of nominal bitrate for Teams, in competition with other flows, as a function of downlink bitrate cap. [*Jamie: Combine this and following figures into one full-width / 6-panel figure. Increase font sizes, etc.*]

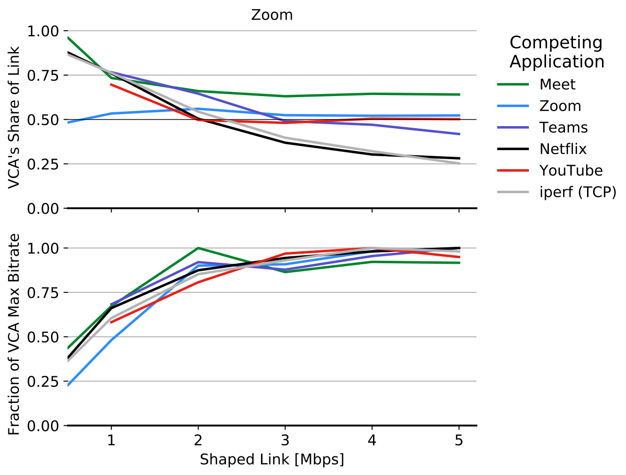


Figure 14: Link share and share of nominal bitrate for Zoom, in competition with other flows, as a function of downlink bitrate cap.

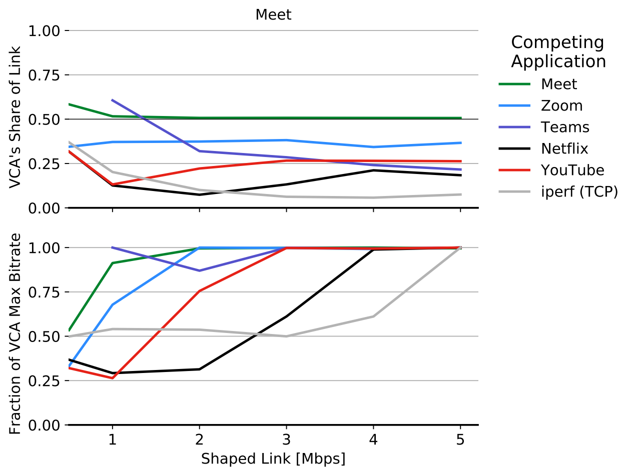


Figure 13: Link share and share of nominal bitrate for Meet, in competition with other flows, as a function of downlink bitrate cap.

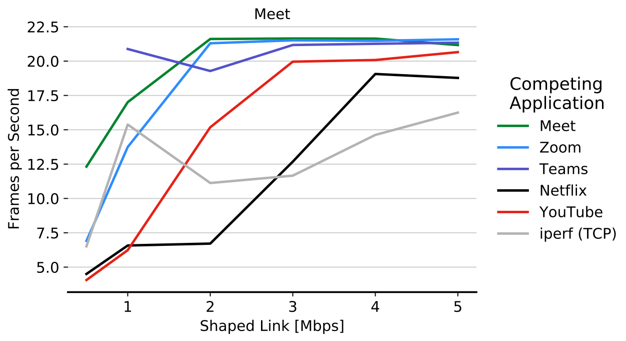


Figure 15: Meet frames per second, in competition with other flows and as a function of link capacity.

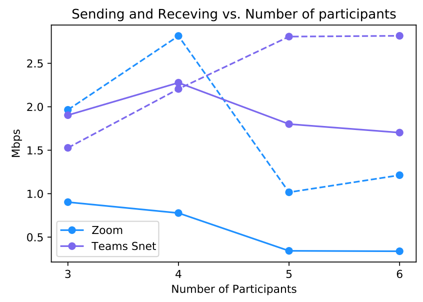


Figure 16: Sending and Receiving Rate vs. Num Participants

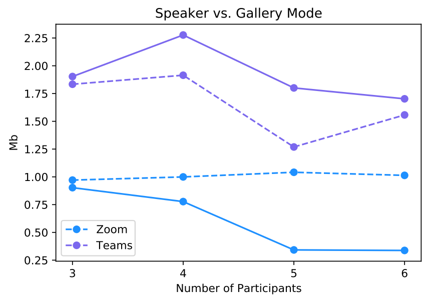


Figure 17: Speaker vs. Gallery Mode

7.1 Number of Users

We fix the viewing mode in gallery, where no one user's video is pinned. We find that increasing the number of participants may reduce the network utilization. In particular, the sending rate decreases as the number of participants increases. One reason for this is that increasing the number of participants in a call reduces the size of their video on the receiver's screen. As a result, the sender's video resolution can decrease. Similarly, we see how the received bitrate drops dramatically at 5 and 7 participants for Zoom and Meet, respectively. We observe that Teams is also exhibiting a downward trend at 8 participants.

7.2 Viewing Mode

We find that viewing a user's video in speaker mode leads to greater network consumption on the user's network. When C1's video is pinned, all other clients view a higher resolution from video. Thus, C1 must continue sending at a higher resolution. Zoom consistently sends at 1 Mbps when all clients pin C1's video, regardless of the number of participants.

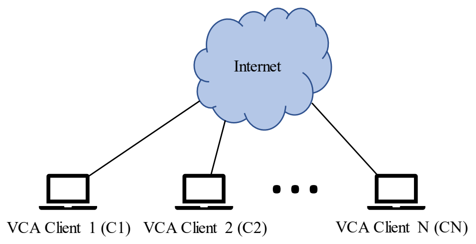


Figure 18: Modality Setup

8 Related Work

Performance of Internet applications There is also related work on measuring other applications over the Internet including video streaming [], web browsing [], and online gaming []. Our work focuses on VCAs which are characterized by their strict latency requirements and upstream link utilization.

VCA measurement studies Early VCA measurement work has focused on uncovering the design of the Skype focusing on the streaming protocols [4] and architecture [1–3]. More recent work including other streaming applications [5, 7] highlighting the differences in design of these applications. In this paper, we take a fresh take on the measurement focusing on the application performance and impact of new application modalities on network utilization.

Application fairness There is also work on measuring application fairness including fairness among TCP variants, or fairness of TCP vs other UDP-based protocols (e.g., QUIC, WebRTC), or applications. Our work focuses on how VCAs share bandwidth with other VCAs, standard TCP implementations, and also applications with their additional bandwidth adaptation over TCP.

9 Conclusion

We analyse three major video conferencing applications.

Future work: - generalizability to other VCAs ? how do we obtain ground truth metrics for other VCAs? can we infer application performance metrics from network data?

References

[1] Salman A Baset and Henning Schulzrinne. "An analysis of the skype peer-to-peer internet telephony protocol." In: *arXiv preprint cs/0412017* (2004).

[2] Dario Bonfiglio, Marco Mellia, Michela Meo, Nicolo Ritacca, and Dario Rossi. "Tracking down skype traffic." In: *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*. IEEE. 2008.

[3] Dario Bonfiglio, Marco Mellia, Michela Meo, and Dario Rossi. "Detailed analysis of skype traffic." In: *IEEE Transactions on Multimedia* (2008).

[4] Saikat Guha and Neil Daswani. *An experimental study of the skype peer-to-peer voip system*. Tech. rep. Cornell University, 2005.

[5] Antonio Nistico, Dena Markudova, Martino Trevisan, Michela Meo, and Giovanna Carofiglio. "A comparative study of RTC applications." In: *2020 IEEE International Symposium on Multimedia (ISM)*. IEEE. 2020.

[6] Henning Schulzrinne, Stephen Casner, Ron Frederick, Van Jacobson, et al. *RTP: A transport protocol for real-time applications*. 1996.

[7] Yang Xu, Chenguang Yu, Jingjiang Li, and Yong Liu. "Video telephony for end-consumers: Measurement study of Google+, iChat, and Skype." In: *IMC*. ACM. 2012.