Packet-Level Analysis of Zoom Performance Anomalies

Mehdi Karamollahi
mehdi.karamollahi@ucalgary.ca
University of Calgary
Calgary, Alberta, Canada

Carey Williamson
cwill@ucalgary.ca
University of Calgary
Calgary, Alberta, Canada

Martin Arlitt
marlitt@ucalgary.ca
University of Calgary
Calgary, Alberta, Canada

ABSTRACT

In this paper, we use Wireshark packet-level traces to study the performance of the Zoom network application. Our work is motivated by several anecdotal reports of Zoom performance problems on our campus network during the Fall 2021 semester. Through the collection and analysis of Wireshark traces from different vantage points, we are able to pinpoint the root cause of the Zoom performance problems, which is a congested external Internet link for our campus network. We also identify several characteristics of the Zoom application that exacerbate its performance issues on congested and lossy networks, due to multi-layer protocol interactions.

CCS CONCEPTS

• **Networks** → **Network measurement**; **Network performance analysis**.

KEYWORDS

Zoom videoconferencing, network traffic measurement, network debugging, performance evaluation, Quality of Experience (QoE)

ACM Reference Format:
Mehdi Karamollahi, Carey Williamson, and Martin Arlitt. 2023. Packet-Level Analysis of Zoom Performance Anomalies. In *Proceedings of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23), April 15–19, 2023, Coimbra, Portugal.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3578244.3583725

1 INTRODUCTION

Zoom is a videoconferencing network application that is widely used for remote work and learning. Due to Zoom's availability and ease of use, it became the solution of choice for many organizations during the COVID-19 pandemic when hosting virtual conferences, remote meetings, and online lectures. Zoom's unprecedented growth in usage made it one of the most prominent videoconferencing applications during the pandemic [2, 17], and its popularity continues today in hybrid work settings.

Our university used Zoom for remote teaching and learning throughout the COVID-19 pandemic. Zoom worked extremely well during the Fall 2020 semester, when almost all users were working and learning from home. However, the Zoom application encountered many performance problems during the Fall 2021 semester, when our university switched to a hybrid learning mode (i.e., some in-person classes, and some online classes using Zoom) with about 50% of students back on campus.

Anecdotally, the Zoom performance problems were attributed simply to the increased presence of students on campus, and the concomitant increase in network traffic, including streaming and WiFi traffic. However, we believe that the Zoom application itself is at least partly to blame for the network performance issues. We thus seek to find a technically deeper and more satisfying explanation for the Zoom performance anomalies that we observed.

As motivation for our work, Figure 1 shows time-series plots of the Zoom packet traffic collected using Wireshark [25] from two different Zoom meetings on campus. Figure 1(a) shows a normal Zoom session that lasted about 40 minutes. This meeting took place on August 31, 2021, about a week before the start of the Fall 2021 semester. The host of the meeting was on campus, while the other two participants were not. The meeting entered peer-to-peer (P2P) mode when the second person joined, and then switched to client-server mode when the third person joined. The graph shows the number of Zoom packets observed in each one-second interval of the Wireshark trace, which was collected by the on-campus user. Figure 1(b) shows an anomalous Zoom meeting from October 6, 2021, on a busy day in the middle of the Fall 2021 semester. This was a client-server Zoom meeting with eight participants, one of whom was on campus. During this one-hour meeting, the participants experienced audio stalls, choppy video, and several occurrences of the "Your Internet connection is unstable" warning message in Zoom. The Wireshark trace shows extreme spikes in the network traffic, which disrupted the session several times. Each of these disruptions affected the user-perceived quality of the Zoom session. Performance problems such as these were fairly common when using Zoom on campus during the Fall 2021 semester.

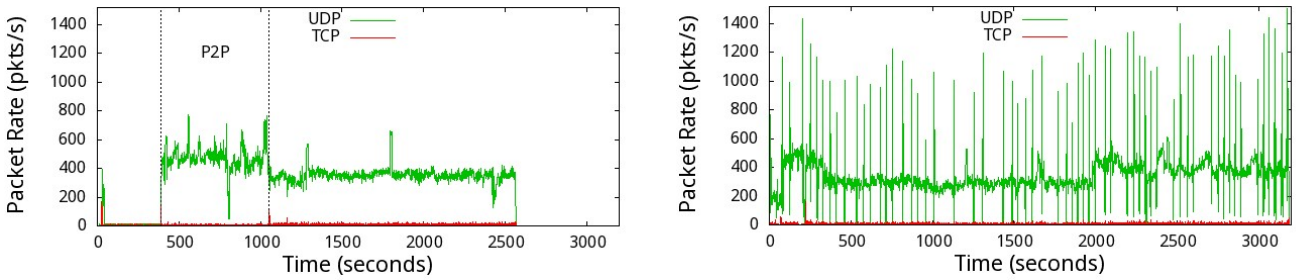
Our goal in this paper is to better understand why these Zoom performance anomalies occurred so often on our campus network. To do so, we provide a detailed packet-level analysis of Zoom traffic using Wireshark traces. Using Wireshark to study network applications is not a new idea in itself. Rather, our main novelty is focusing on the content of the application-layer headers carried in Zoom UDP packets. An initial entropy analysis on these headers identified several fields containing useful information regarding network application behavior [4]. Our detailed analysis of these traces has since provided much deeper insight into the structure of Zoom traffic, and its performance issues on our campus network.

The main contributions of our paper are: (1) packet-level analysis of Zoom test sessions on our campus network; (2) insights into the structural characteristics of Zoom traffic; (3) insights into the performance problems on our campus network; and (4) recommendations to improve Zoom performance on enterprise networks.

The main insights that emerge from our work are:

ICPE '23, April 15–19, 2023, Coimbra, Portugal

ACM ISBN 979-8-4007-0068-2/23/04…$15.00
https://doi.org/10.1145/3578244.3583725



(a) Trace A (August 31, 2021)

(b) Trace B (October 6, 2021)

Figure 1: Time series plots of packet arrival rates during two Zoom test sessions

- Zoom UDP packets carry application-layer header information that is unencrypted. Sequence numbers within these media streams can be used to estimate packet loss, and embedded timing probes can be used to estimate network delay and jitter;
- Zoom has several resilience mechanisms, such as connection-level restart and application-layer retransmission;
- Zoom uses aggressive bandwidth probing to dynamically adapt video bit rates based on network conditions;
- a congested external Internet link is the root cause of Zoom performance problems on our campus network; and
- multi-layer protocol interactions exacerbate the network performance problems for Zoom.

The rest of this paper is organized as follows. Section 2 covers background information and prior related work. Section 3 describes our measurement methodology for the collection and analysis of Wireshark packet traces. Section 4 presents results regarding Zoom traffic analysis and performance problems observed. Section 5 summarizes several takeaways from our work, discusses performance implications, and provides recommendations to improve Zoom performance in the future. Finally, Section 6 concludes the paper.

2 BACKGROUND AND RELATED WORK

2.1 Zoom

In a Zoom meeting, there are multiple Zoom *sessions*, with one session for each participating user. If only two users are in a meeting, then the meeting operates in peer-to-peer (P2P) mode, with direct communication between the two participants. If there are more than two users, then Zoom switches to client-server (C-S) mode with all media content streamed to and from a Zoom Multimedia Router (MMR) in the cloud [27].

Within a Zoom session, there are multiple transport-layer *connections*. Zoom uses both TCP and UDP[1] at the transport layer. TCP is used for control and management of the Zoom session, including operating mode, screen layout, document transfer, and chat. UDP is used for real-time streaming of audio, video, and screensharing data. In C-S mode, each of these media streams is associated with a distinct UDP port at the client. In P2P mode, all media content is multiplexed onto a single UDP port at each client.

[1]Though UDP is connectionless, we still refer to these as connections, because of the application-layer state information associated with each bidirectional UDP flow.

Zoom is a highly resilient network application. It can dynamically adapt its video bit rate to different network conditions [3, 17]. It also has several failure recovery mechanisms, including Forward Error Correction (FEC), application-layer retransmission, and connection re-establishment [5, 17].

2.2 Related Work

Many researchers have studied the impacts of the COVID-19 pandemic on Internet traffic [1, 3, 8–10, 16]. In particular, videoconferencing applications became a key focus with their increasing usage for remote work and learning [17–19].

Some earlier works [18, 19] focused on the privacy and security of Zoom, which was a concern at the onset of the pandemic. Marczak *et al.* [19] studied the cryptographic properties of Zoom, while Mahr *et al.* [18] conducted a deeper forensic analysis of Zoom's packet traffic. Our own work focuses specifically on Zoom-related performance issues, as evidenced by its packet-level traffic.

Sander *et al.* [23] investigated the flow-rate fairness of Zoom congestion control. They showed that Zoom uses two to three times the bandwidth of TCP in low-bandwidth scenarios. Other authors have also documented the bandwidth unfairness of Zoom [17].

Several recent works have studied how Zoom interacts with other competing network applications, including Meet, Teams, Webex, and itself. For example, Chang *et al.* [3] experimentally evaluated the user-perceived Quality of Experience (QoE) for Meet, Webex, and Zoom using a cloud-based framework. They compared the three applications by emulating geographically distributed videoconferencing sessions, and measuring the media bit rates and video quality metrics. As another example, MacMillan *et al.* [17] compared the performance of Meet, Teams, and Zoom under different emulated network conditions. They showed that the three applications vary significantly both in terms of resource utilization and performance, especially when network capacity is limited.

The most recent and most relevant prior work is by Michel *et al.* [20]. These authors conducted a detailed analysis of unencrypted Zoom packet headers, identifying the RTP/RTCP header information embedded within Zoom UDP packets, as well as the Zoom media encapsulation headers that precede them. They then used this information to analyze a 12-hour trace of campus network traffic to assess Zoom performance (e.g., media bit rate, number of meetings, delay, jitter). Their approach is also effective at detecting P2P Zoom traffic, which prior approaches were unable to do.

In our own prior work [12], we conducted a longitudinal study of Meet, Teams, and Zoom on a campus network. This earlier work provided evidence of issues with Zoom TCP connections and session management. In subsequent followup work [5], we provided an in-depth connection-level analysis of Zoom traffic and showed how Zoom QoE can suffer on a large campus network. We identified concurrent meetings, high video bit rates, correlated session arrivals, and long-lasting sessions as key contributing factors to the performance problems observed. These two prior works studied tens of thousands of empirical Zoom sessions on our campus network using passive connection-level analysis, and identified many sessions with Zoom performance problems. Our current paper uses packet-level analysis to better understand these performance anomalies, using Wireshark traces from Zoom test sessions.

In [13], we developed a synthetic workload model and a simulation model for Zoom traffic. The simulation results showed that campus network load depends greatly on the location of Zoom users, and that load-aware policies for Zoom server selection can improve robustness. Multicast support for Zoom is also suggested to decrease the load on congested enterprise networks.

Our work is complementary to these previous studies. We study Zoom traffic "in the wild" on a campus network, rather than emulated or simulated traffic as in many of the foregoing studies. Furthermore, we use packet-level analysis, rather than connection-level analysis, to identify specific Zoom performance problems.

3 EXPERIMENTAL METHODOLOGY

This section discusses the methodology that we used to collect and analyze Zoom traffic at the packet level.

3.1 Network Environment

We study Zoom usage on the University of Calgary network, as an example of a campus edge network. This network has a user community of about 35,000 students, faculty, and staff.

Figure 2 shows the network setup used for our experiments. The three large circles in the diagram show the campus network, an ISP network, and the Zoom cloud infrastructure. A router (R) connects the university network (U) to the Zoom network (Z), as well as to the ISP network. Within the campus network, users (i.e., faculty, staff, students) can be on the wireless network (WiFi), or desktop users on Ethernet LANs. We also consider possible P2P Zoom users (P1 and P2) on the campus network, who could be on wired or wireless networks. Inside the ISP network, there can be home users (H) using Zoom, including possible P2P users P3 and P4. Regarding Zoom infrastructure, regional data centers in Vancouver and Toronto handle over 90% of our campus Zoom traffic.

On our campus network, the external U-R link for commercial Internet traffic is 6 Gbps. As will become evident later in the paper, this link is a network bottleneck, and is undoubtedly the root cause for many of the Zoom performance problems observed. As evidence for this claim, Figure 3 presents a time-series graph of the traffic volume traversing this link on a per-minute basis on Wednesday, October 6, 2021. The black line shows the total traffic, while the other[2] four lines show TCP and UDP traffic volumes, each separated into inbound and outbound traffic.

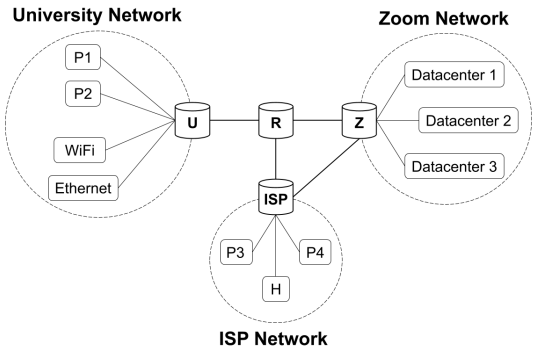


Figure 2: Network setup for Zoom traffic measurements

Figure 3 shows that our external Internet link was fully saturated most of the day, roughly from 8:00am to midnight. Zoom UDP traffic has a distinctive pattern, reflecting the hourly class lecture slots on this day. Zoom contributes about 1 Gbps of UDP traffic volume during the main part of the day, indicating a large number of Zoom participants (faculty and students) on campus. What is also interesting is how the TCP traffic reacts inversely to the UDP traffic, declining when UDP rises, and rising when UDP declines. The explanation is TCP's congestion control mechanism, which detects and adapts to the network bottleneck, while Zoom UDP traffic does not. This behavior provides further evidence that the external Internet link is indeed "full" most of the time.

Figure 3 indicates that the aggregate network traffic on the external link is being capped by a rate-limiting device, either on campus or in the provider's network. Analysis of historical campus-level data (not shown here) indicates that this rate limit has been in effect since March 2018, though its impact was only noticed recently, when Zoom usage became prevalent. Interestingly, the 6 Gbps limit is independent of the direction of the traffic, and protocol-agnostic. We thus expect the campus network bottleneck to affect all traffic (i.e., inbound and outbound, wired and wireless, TCP and UDP, Zoom and non-Zoom), though we focus specifically on Zoom traffic in the rest of the paper.

3.2 Wireshark Traces

We used Wireshark to collect packet-level traces from the client endpoints in Zoom test sessions. Traces were only collected with the explicit knowledge and consent of Zoom meeting participants, who opted in for research data collection.

We collected several Wireshark traces each week, over the span of about eight weeks during the Fall 2021 semester. Most traces were collected consistently on the same days and times each week,

[2]To improve clarity, we have excluded protocols such as ICMP, IPv6, and internal subnet traffic from the graph. The transient spikes at 6am, noon, 6pm, and midnight are artifacts of restarting Zeek [22] every 6 hours to avoid crashing during scanning attacks [12]. These spikes reflect inaccurate traffic counts, which can be ignored.

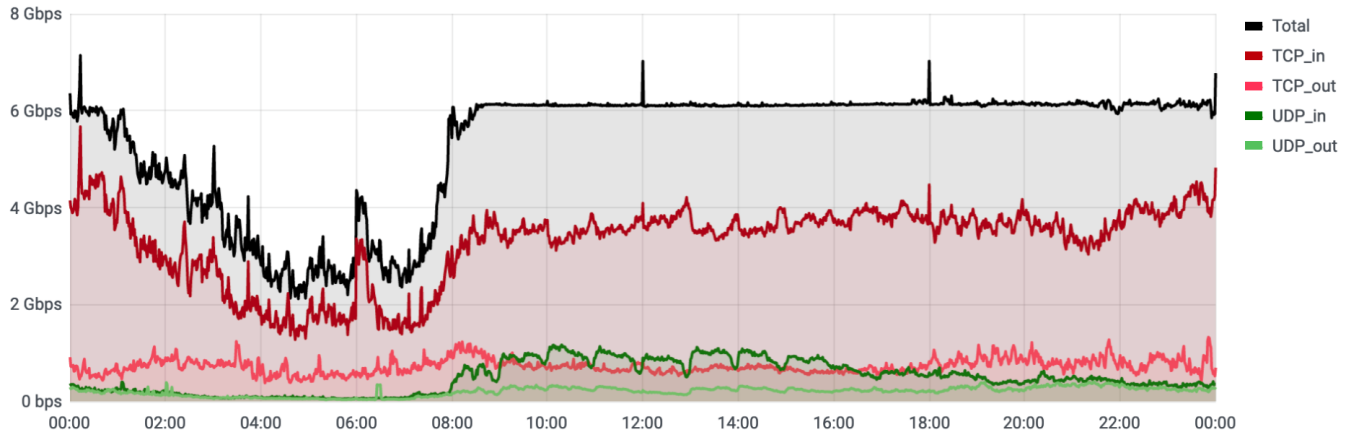


Figure 3: Daily Traffic Volume on Campus External Link for Commercial Internet Traffic (Wednesday, October 6, 2021)

while others were collected spontaneously when Zoom performance problems were particularly apparent on the network. We varied the locations from which traces were collected, with some on campus, and some at home. We also varied the number of participants in the test sessions, to obtain P2P Zoom examples as well as C-S Zoom examples.

We have a total of 40 Wireshark traces, a small subset of which are used in this paper, as shown in Table 1. The leftmost columns provide metadata about the traces; the remaining columns are explained later in Section 4. The most valuable traces (C and D) were those collected concurrently at two different endpoints, with one user on campus, and one user at home (e.g., P1 to P3 in Figure 2).

For each trace, we ran Wireshark in promiscuous mode to capture the network traffic from *all* concurrently running applications on the laptop. However, the analysis in the paper focuses only on the Zoom traffic. In Trace A, for example, UDP accounts for 96.7% of the packets. Only 8 of the 277 UDP connections are for Zoom, but they account for 99.9% of the UDP packets, since the others are mostly DNS. Similarly, only 12 of 125 TCP connections are Zoom-related, but they account for 65% of the TCP packets; the other TCP traffic is mostly HTTPS for Google, Office365, Fortinet VPN, etc.

Running Zoom and Wireshark at the same time does not seem to be much of an issue on a modern laptop. Trace A, for example, shows that only 0.1% of the (high-rate) video packets in C-S mode are missing, and these are most likely due to network losses, rather than Wireshark drops. Wireshark itself reports an estimate of 0.0% dropped packets in the Capture File Properties for Trace A (and the other traces as well).

As with most empirical measurement work, repeatable experiments are a challenge, since ambient campus network congestion changes each time. We believe that the traces presented in the paper are a minimal subset to pinpoint and explain the Zoom performance problems that we have observed. We have dozens of other traces, many of which show similar problems, and some (such as those when all users are at home) show no problems at all.

3.3 Empirical Observations

From our analysis of our Wireshark traces, we have made the following empirical observations about Zoom traffic:

(1) Within the UDP segments carrying Zoom traffic, the data payload contains a short RTP-like [24] application-level header that is unencrypted [20, 21]. This header provides information that is used in demultiplexing UDP Zoom traffic.
(2) In a client-server Zoom session, the first byte of the UDP data payload (i.e., `data[0]` in Wireshark) indicates an opcode that is related to the Zoom session [19]. Table 2 shows the main opcodes observed in this traffic, and our interpretations of them, based on their timing and location in the Zoom session. These opcodes occur for each client port used for the Zoom session. Several of these opcodes match those reported by Marczak *et al.* [19], although Zoom's application-layer header format has changed slightly since their work.
(3) The vast majority (over 90%) of the UDP packets carry opcode 0x05, indicating a media data unit (e.g., video, audio, or screensharing data). In client-server mode, there are three different UDP ports at the client side, with one for each of these three streams. At the server side, all Zoom traffic goes in and out on UDP port 8801 [26].
(4) Within the UDP segments carrying opcode 0x03 (ECHO REQUEST), there is a sequence number and timestamp inserted by the sender. These values are echoed back by the receiver using opcode 0x04 (ECHO RESPONSE). These exchanges happen every 1.0 seconds on the audio stream, and every 3.75 seconds on the video and data streams. Both endpoints generate these timing probes throughout the session, doing so for each client UDP port. Each endpoint uses its own sequence numbers, which start from 1.
(5) Within the UDP segments carrying opcode 0x05 (MEDIA UNIT), there is an additional type code in UDP payload byte `data[8]`. The most salient ones are summarized in Table 2. In addition, there is a 16-bit sequence number associated with the media unit being carried. Sequence numbers start from 0 and typically increase monotonically, but may wrap around every few minutes, depending on the type of media stream

Table 1: Summary of Zoom Wireshark Trace Collection and Analysis

Wireshark Trace Metadata							UDP Packet Loss Estimates				TCP Packet Events		
Trace	Date	Time	Location	Mode	Duration	Packets	Video	Audio	Data	Probes	Retx	Spur	Dup
A	Tue Aug 31 2021	12:08pm	Campus	P2P	11 min	315,666	1.27%	1.04%	N/A	1.53%	13	8	18
			Campus	C-S	32 min	559,827	0.10%	0.17%	0.00%	0.50%	117	100	380
B	Wed Oct 6	2:08pm	Campus	C-S	53 min	1,114,451	2.04%	1.67%	N/A	1.65%	696	386	1,277
C1	Wed Oct 13 2021	12:58pm	Campus	P2P	36 min	884,467	0.67%	0.96%	N/A	0.96%	1,389	796	2,506
C2		1:00pm	Home	P2P	34 min	844,538	1.15%	1.45%	N/A	0.50%	1,751	655	938
D1	Wed Oct 27 2021	12:57pm	Campus	P2P	4 min	94,400	1.97%	3.10%	N/A	2.17%	151	60	410
			Campus	C-S	35 min	653,494	3.93%	6.13%	N/A	4.54%	649	421	1,270
D2		1:02pm	Home	P2P	2 min	37,612	4.27%	4.96%	N/A	5.26%	1	0	26
			Home	C-S	32 min	593,973	0.00%	0.00%	N/A	0.00%	1	0	189

Table 2: Opcodes Observed in Zoom UDP Traffic (C-S Mode)

Opcode	Offset	Interpretation	Description
0x01	0	READY	C-S initialization
0x02	0	GO	S-C initialization
0x03	0	ECHO REQUEST	Timing probe request
0x04	0	ECHO RESPONSE	Timing probe response
0x05	0	MEDIA UNIT	Media unit exchange
0x06	0	N/A	Not used
0x07	0	QUIT	Terminate connection
0x0a	8	DATA	Screen-sharing (S-C)
0x0d	8	DATA	Screen-sharing (C-S)
0x0f	8	AUDIO	Audio media unit
0x10	8	VIDEO	Video media unit
0x15	8	BWPROBE	Video bandwidth test

Table 3: Opcodes Observed in Zoom UDP Traffic (P2P Mode)

Opcode	Offset	Interpretation	Description
0x0f	0	AUDIO	Audio media unit
0x10	0	VIDEO	Video media unit
0x15	0	BWPROBE	Video bandwidth test
0x1e	0	DATA	Screen-sharing (data)
0x20	0	DATA	Screen-sharing (control)
0x21	0	VIDEO PROBE	Video timing probe
0x22	0	AUDIO PROBE	Audio timing probe

being carried. Each endpoint has its own sequence numbers. Type codes 0x10 and 0x15 share the same sequence number space. Gaps in the sequence numbers likely indicate lost or mis-ordered packets, changes in the media encoding format, or different media sub-streams [20]. Repeats in the sequence numbers indicate application-layer retransmissions.

(6) In a P2P Zoom session, all UDP traffic is sent and received using a single UDP port, rather than three client ports. The opcode functionality mentioned previously is still present, as shown in Table 3, but it is not always in the same place. For example, the first few opcodes are the same, and appear in `data[0]`, but the 0x05 opcode is not used. Rather, the media type codes appear directly in `data[0]`, and the locations of sequence numbers and timestamps vary with the media type. Video timing probes are at 5-second intervals.

(7) Within Wireshark, the packet loss rates can be calculated based on the starting and ending sequence numbers used for a particular opcode or media type, compared against the number of such frames captured in the Wireshark trace. An adjustment may be needed for sequence number wraparound, if it occurs.

(8) Within Wireshark, the timing probes can be used to estimate the network delay and jitter.

With the foregoing knowledge about Zoom UDP payload information, we can quantify network congestion effects (i.e., delay, jitter, loss, retransmission) on Zoom traffic, using Wireshark traces.

4 MEASUREMENT RESULTS

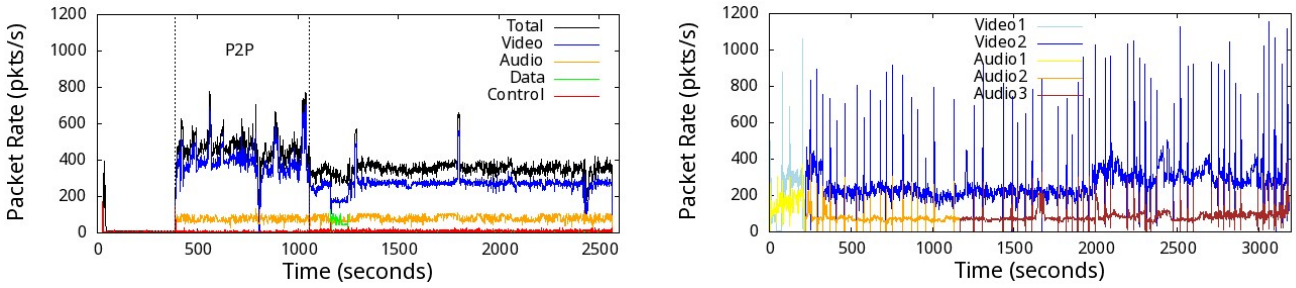
This section presents selected results from our packet-level analysis of Zoom Wireshark traces. Our goals are to understand how Zoom performance is affected by the campus network, as well as how Zoom's behavior affects the campus network.

4.1 Media Stream Analysis

We start our detailed Zoom analysis by looking at the individual media streams within a Zoom session, to see if any anomalies are evident. Figure 4 shows the results of this analysis.

Figure 4(a) shows a detailed breakdown of the normal Zoom session from Trace A. The initial P2P mode is shown in the plot, while both the P2P and C-S portions are broken down into their constituent media streams. The main observations here are that the video traffic dominates, and that P2P mode uses a higher video bit rate than C-S mode. These observations are as expected, and consistent with those made by other researchers [3, 5, 17]. One new observation here is the presence of occasional spikes in the video traffic, which are attributable to Zoom's bandwidth probing. We defer detailed analysis of this feature to Section 4.7.

Figure 4(b) shows a detailed breakdown of the Zoom session from Trace B. To improve clarity, we only show the audio and video traffic in this trace, with additional colors used to indicate the multiple instances of UDP connections used during this session. We consider the network activity in this trace to be anomalous



(a) Trace A (b) Trace B

Figure 4: Media stream analysis of two Zoom test sessions

due to the extreme spikes observed in the traffic. The spikes affect audio traffic and video traffic in the same way, suggesting that both experience significant queueing at the network bottleneck.

A more detailed analysis of Trace B shows occasional idle periods of 2-3 seconds with no arriving traffic at all, followed by the back-to-back arrival of hundreds of packets, as indicated in the graph. Furthermore, the spikes are quasi-periodic, often appearing about every 35 seconds. We speculate that the saturation of the bottleneck link is causing the formation of a massive queue, which takes several seconds to dissipate. The traffic volume then returns to its usual level for the next 30 seconds or so, before the next spike occurs. This pattern occurs many times during this particular trace.

4.2 Packet Loss

We next try to estimate the packet loss ratio for Zoom traffic, using the sequence numbers embedded in media streams. Table 1 provides a quantitative summary of these results, in the middle columns of the table. Note that estimates can be made independently for each media type, as well as for the timing probes. While the number of samples vary for each media type, consistency in the estimates across media types provides greater confidence in the estimates, to cope with the possibility of Wireshark itself missing packets.

Table 1 shows Zoom packet loss estimates for our Wireshark traces, based on sequence number analysis for each media type. As can be seen, the packet loss rates for Trace A are quite low for the P2P portion of the meeting, and even lower for the C-S portion. For Trace B, the packet loss rates are much higher, reflecting the higher load placed upon the campus network in Fall 2021 when many students were back on campus. The packet loss rates are even higher for Trace D1 from October 27, which exhibited similar performance problems as Trace B from three weeks earlier.

The foregoing observations indicate a performance bottleneck for Zoom traffic, but it is not clear whether this bottleneck is at the client, at the server, or within the network itself. Our next experiment offers further insight into this issue.

4.3 Delay and Jitter

The timing probes embedded within Zoom media traffic can be used to assess network latency effects. In particular, we use the video traffic timing probes that are sent by the Zoom server to each user every 3.75 seconds. Figure 5 shows the results from our analysis, using Cumulative Distribution Function (CDF) plots to show the inter-arrival time (IAT) distribution for the timing probes arriving at the client. These CDF plots are generated from specific individual traces, each with several hundred measurement samples. The plots from other traces are qualitatively similar, though unique for each trace. Statistical analysis and comparison is often noisy because of several outliers in the distributions.

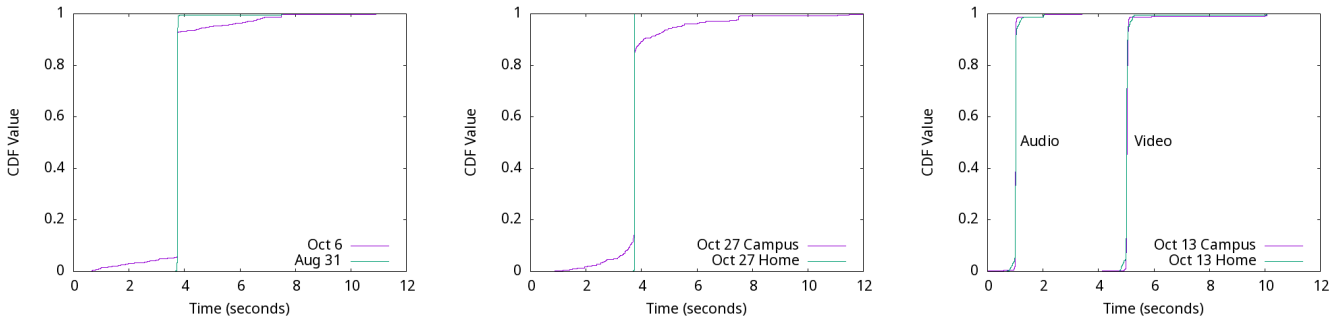
Figure 5(a) compares the timing probes for Trace A with those from Trace B. For Trace A on August 31, the server probes arrive almost unperturbed, resulting in a nearly vertical CDF at the expected value of 3.75 seconds. For Trace B on October 6, the CDF is altered a lot, with about 5% of the probes arriving late, and another 5% of the probes seemingly arriving "early", since the IAT is shorter for the next probe if the previous probe was late. The graph shows that some of the probes are up to 3 seconds late, consistent with the delays noted earlier. Furthermore, the delays are almost uniformly distributed between 0 and 3 seconds, indicating some randomness in when these hiccups occur on the network. There is also a small jump in the IAT distribution at 7.50 seconds, indicating some missing (lost) timing probes.

Figure 5(b) presents the IAT CDF for Traces D1 and D2 from the October 27 meeting, for which both the on-campus user and the at-home user collected Wireshark traces. This graph is particularly illuminating, showing that the timing probes sent from the server experience problems when entering the campus network, but *not* when traversing the ISP network. This result clearly indicates that the bottleneck is on the campus network.

For completeness, we also checked the timing probes in Traces C1 and C2 for the P2P Zoom meeting from October 13, 2021. Figure 5(c) shows these results, broken out by audio probes (1-second intervals) and video probes (5-second intervals). The plot shows that both users had similar conditions on the network path, and only small perturbations to the CDF of probe inter-arrival times. Since these network conditions were reasonable, the Zoom meeting stayed in P2P mode (as expected) throughout the session.

4.4 Directionality

We next try to ascertain whether the network bottleneck affects inbound traffic, outbound traffic, or both. For this purpose, Figure 6 presents time series plots of the packet rates for Traces D1 and D2 from the October 27 Zoom session, as viewed by the campus user (Figure 6(a)) and the home user (Figure 6(b)). In each of these graphs, we have separated the UDP traffic into inbound (green) and



(a) Traces A and B (b) Traces D1 and D2 (c) Traces C1 and C2

Figure 5: CDF analysis of inter-arrival times for Zoom timing probes

outbound (brown), with Zoom's TCP control traffic shown in red. All of the following time series plots use data values extracted from the traces, using Wireshark's graphical visualization tools. Table 1 summarizes the key statistical properties of these traces.

As a cautionary note, recall that Wireshark traces collected at a client endpoint inherently show biased timing information. Specifically, Wireshark sees outgoing packets when they are launched, which is *before* they have traversed the network bottleneck, while incoming network packets are seen *after* they have traversed the network path, including the bottleneck. As such, traces D1 and D2 both show that outgoing Zoom traffic is unaffected by the bottleneck. However, because we have Wireshark traces from each client endpoint, we can further study this traffic, based on the packet arrivals at the other endpoint.

Figure 6(a) shows prominent spikes in the packet traffic arriving from the Zoom server. This observation confirms that the bottleneck affects the incoming traffic. However, Figure 6(b) for the home user also shows some spikes (albeit smaller ones) in the incoming traffic from the Zoom server, indicating that the outgoing traffic from the campus user was affected by the bottleneck as well. In particular, this outgoing traffic arrives at the Zoom server with some spikes, and is relayed to the home user in similar form (recall that the video timing probes from the Zoom server to the home user in Trace D2 arrived unperturbed).

While the amplitude of the traffic spikes is diminished for the home user, they are still prominent. This observation confirms that the campus network bottleneck also affects the outbound traffic. Further evidence for this claim is the brief 4-minute P2P portion of these traces, which shows spikes in the packet traffic arriving at each endpoint, despite the absence of spikes in the outgoing traffic sent by each client. The spikes again occur about every 35 seconds. This periodicity seems to arise from the confluence of a 3-second delay at a networking device (possibly due to a queue, a timeout, or a software reset) and the 32-second interval used by Zoom's bandwidth probes (see Section 4.7) when recovering from such events. Bandwidth probing seems to be triggered by significant changes in loss or delay [4].

4.5 Retransmissions

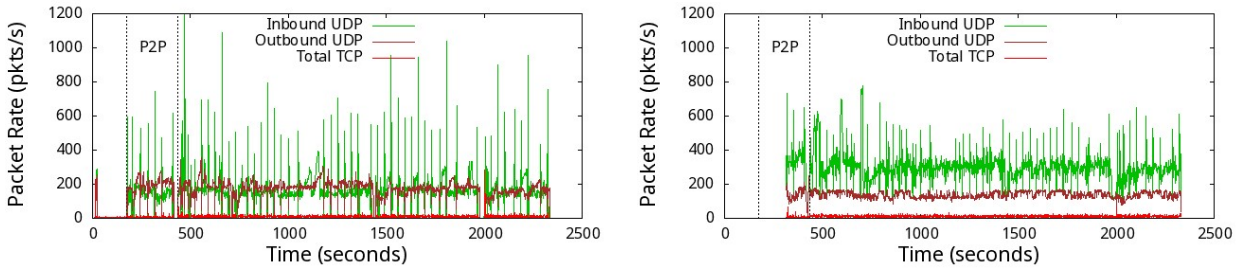
In our next experiment, we study the impact of the network bottleneck on TCP traffic, rather than UDP traffic. This experiment offers further insight into the nature of Zoom performance problems.

Recall that TCP is a connection-oriented reliable data transfer protocol, unlike UDP, which is connectionless and unreliable. In particular, TCP uses sequence numbers, acknowledgements, timers, and retransmissions to achieve reliable data delivery. It also has flow control and congestion control mechanisms that can detect and react to the delay and/or loss of TCP packets. Some manifestations of network performance problems include lost packets, out-of-order packets, retransmissions, duplicate ACKs, and spurious retransmissions (i.e., retransmissions that occur while a previous copy of the same packet is still in flight).

The rightmost three columns of Table 1 report the counts for these TCP-related events, as indicated in our Wireshark traces. We use "Retx" for ordinary retransmissions, "Spur" for spurious retransmissions, and "Dup" for duplicate ACKs. These events occur quite often, especially in the anomalous traces. These results confirm that TCP is affected by the network bottleneck just as much as UDP is, although TCP reacts differently.

Figure 7 shows a time-series plot of the spurious retransmissions observed on the TCP control connections for the campus user in Trace D1. The volume of retransmissions is extremely high, as is the number of duplicate ACKs (see Table 1). Detailed analysis of the Wireshark traces shows that there are occasional brief delays in packet delivery for up to 3 seconds, followed by the rapid delivery of hundreds of backlogged packets, as noted earlier.

These multi-second delays cause numerous problems for TCP. First, the slow delivery on the forward path to the client causes the server to retransmit some packets in an attempt to achieve delivery. Second, these retransmissions lead to multiple copies of the same TCP packet in the queue for delivery. These are detected by Wireshark as spurious retransmissions, as shown in Figure 7. Third, the eventual delivery of duplicate TCP data to the client triggers a duplicate ACK from the client. Fourth, the delays in the return of ACKs to the server complicate the RTT estimation, as well as the setting of the RTO timer and the congestion window. These result in sub-optimal TCP performance. Finally, the Zoom application itself sometimes terminates problematic TCP connections, perhaps due



(a) Trace D1 (Campus) (b) Trace D2 (Home)

Figure 6: Analysis of directional properties of UDP traffic in Zoom test sessions (October 27, 2021)

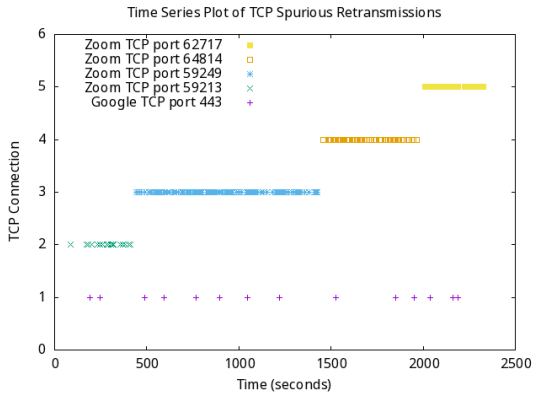


Figure 7: TCP spurious retransmissions in Zoom Trace D1

to excessive delay, loss, or retransmission, before re-establishing a new one. We study this behavior in the next subsection.

As a side comment, Zoom is not the only application that experiences spurious TCP retransmissions. For example, Figure 7 shows a long-lived Google application that was captured in our Wireshark trace as well. This application exchanges a small amount of data with a Google server every 25 seconds. Occasionally, this application encounters a massive queue at the campus network, and has a spurious retransmission. Zoom experiences more spurious retransmissions because it is sending TCP packets more frequently.

4.6 Connection Analysis

Figure 8 shows the connection-level view of three Zoom sessions. These graphs are produced using the traffic analysis tools developed by Choi et al. [5], which we have adapted and applied to the TCP and UDP connections in our Wireshark traces.

Figure 8(a) shows the TCP and UDP connections for the normal Zoom session in Trace A. In this session, there is a single TCP control connection throughout the meeting, as expected. Initially, there are three UDP connections (video, audio, data) to the Zoom server for the meeting host. Once the second participant arrives, the meeting switches to P2P mode, with a single UDP connection for the media streams. When the third participant arrives, the meeting switches back into client-server mode, with three UDP connections to the Zoom server. This behavior is as expected.

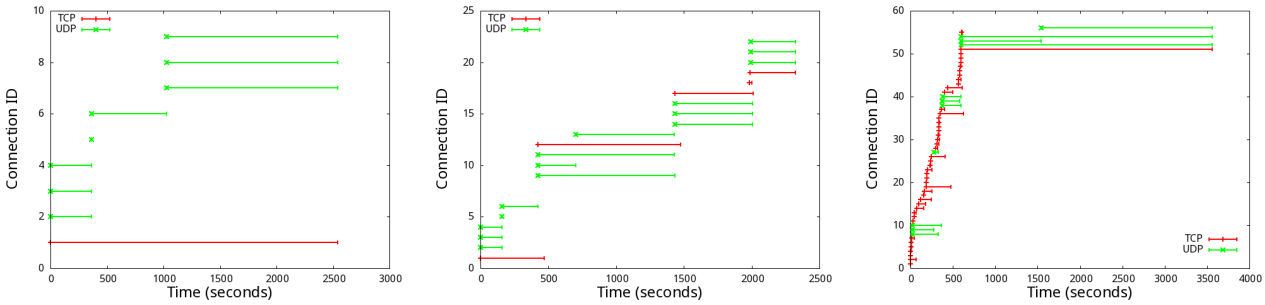
Figure 8(b) shows the different TCP and UDP connections for Trace D1. Recall that this was a two-person meeting, which was disrupted[3] and restarted several times. Several discontinuities are evident in this plot, for both UDP and TCP. Detailed analysis of the Wireshark trace shows that it was the Zoom server (not the client) that proactively terminated the UDP connections (i.e., opcode 0x07 from Table 2), forcing the client to re-initiate these. Furthermore, the TCP connection was closed and restarted several times, perhaps because of the excessive number of TCP retransmissions noted earlier. Most curious of all, this two-person meeting switched from P2P mode to C-S mode, and remained in C-S mode for the remainder of the meeting. This transition was likely due to the high packet loss rate experienced during P2P mode (see Table 1).

Finally, Figure 8(c) shows the TCP and UDP connection profiles for Trace B. While this session had problems similar to the previous example, there are new issues as well. Most notably, the TCP connection to the Zoom server had many disruptions, and required multiple attempts (about 40) before it was successfully re-established. Detailed analysis of the Wireshark traces shows that the problem arose during the TLS handshake, which did not complete soon enough to avoid a 3.5-second TLS timeout/abort at the Zoom server. Rather, repeated TLS handshakes were attempted until one finally succeeded within this time limit.

The anomalous behavior in Figure 8(c) indicates a high volume of TLS handshake requests arriving at the server, somewhat like a TLS/SSL flooding attack. The key insight here is that when a single Zoom user at home has a network connectivity issue, it only affects one user and one TLS connection, but when hundreds of Zoom users on the campus network have a connectivity issue, *they may all be re-establishing TLS connections to the same Zoom server at the same time*. This creates a surge of demand at the server, potentially making it sluggish or unresponsive.

One of Zoom's resilience mechanisms is to switch to a different Zoom server when needed. This approach solves the problem here, with the new Zoom server handling the TCP control connection (but not the UDP connections) for the remainder of the session.

[3] In our prior work [5], we showed that our campus community initiated about 3,500 Zoom meetings per day, and that these meetings used about 400 different Zoom MMR servers, for an average of 9 meetings per server per day. However, some servers were used far more often than others on a given day. More importantly, we have empirically observed some Zoom servers concurrently hosting as many as 6 different Zoom meetings during the busiest part of the day. Cases such as these often exhibited performance problems.



(a) Trace A (b) Trace D1 (c) Trace B

Figure 8: Connection profiles for several Zoom test sessions

Further evidence for this server-side performance issue is provided in Figure 9, which provides the TCP and UDP connection profiles for the P2P Zoom sessions from Traces C1 and C2. The left graph is for the on-campus user, while the right graph is for the home user. In both graphs, there is a single direct UDP connection between the two peers to carry the media traffic throughout the meeting. This UDP connection operates without any disruption. However, the TCP control connection to the Zoom server is disrupted and re-established many times: about 50 times for the on-campus user, and about 30 times for the home user. This confirms that the TCP/TLS performance issue is at the server end.

4.7 Bandwidth Probing

The final part of our investigation focuses on Zoom's bandwidth probing, to see if and how this contributes to the network performance problems observed.

Figure 10 shows the results from this analysis, for both a normal Zoom session (Trace A) as well as an anomalous one (Trace D1). In these graphs, the upper line (in green) shows the total Zoom UDP traffic, while the lower line (in purple) shows the video bandwidth probing traffic carrying opcode 0x15 (see Tables 2 and 3).

Figure 10(a) shows that bandwidth probing is present throughout a normal Zoom session. It is particularly prominent during the initial P2P part of the session, during which higher video bit rates are observed. During the client-server portion of the session, the bandwidth probing has distinct sustained 10-second spikes with hundreds of probe packets, but the frequency at which these spikes occur diminishes. Detailed analysis suggests a multiplicative decrease strategy, with intervals of 256 and 512 seconds observed here. The probe packets themselves are typically about 1 KB in size, with each containing dummy byte values (e.g., 0x01, 0x02, …0x20) in unencrypted form.

Figure 10(b) shows the video bandwidth probing during the problematic Zoom session in Trace D1. The probing during the brief P2P part of the session is similar to that in the previous example. However, the bandwidth probing during the client-server portion of the session is quite different, in two ways. First, the frequency at which it occurs is higher, with a 32-second interval between most probes. This increased frequency is likely because of the apparent instability of the network path. Second, some of the sustained probes last much longer than 10 seconds, even though the total number of probe packets sent remains similar. This observation suggests the inherent sharing of a congested network path with video bandwidth probes destined for other users.

Figure 11 offers additional insight into the bandwidth probing issue, by comparing Traces D1 and D2 side-by-side. In Figures 11(a) and (b) we have also separated the bandwidth probing traffic into that generated by the server (purple) and that generated by the clients (yellow). Two important observations are evident here. First, the Zoom server only does the aggressive bandwidth probing on the unstable path (Campus), and not the stable path (Home). Second, the bandwidth probes occasionally get queued at the network bottleneck and arrive as a traffic spike. In fact, these spikes occur almost periodically, about 150 seconds apart. We surmise that the bandwidth probes for this user are drifting in and out of phase with those from other campus network users in other Zoom meetings.

The key insight here is that the Zoom servers appear to be *conducting bandwidth probes independently for every user in every Zoom meeting on the congested campus network*. This probing adds a lot of extra load to the network, which is already congested, potentially leading to the queueing phenomenon noted earlier. Furthermore, the bandwidth probes for different users can also overlap, analogous to the router synchronization problem first identified by Floyd and Jacobson [11]. In our campus network configuration, these traffic spikes actually contribute to the network performance problems, which in turn affect Zoom's user-perceived performance.

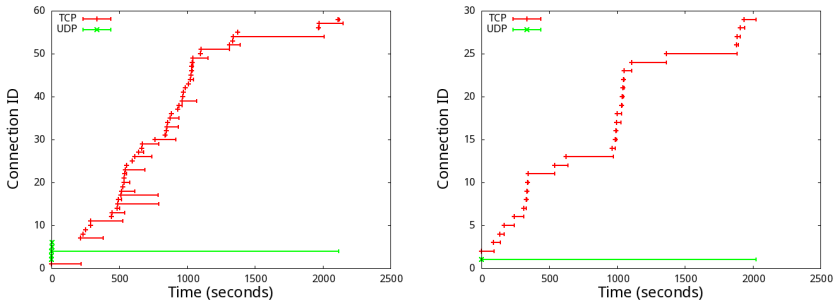
5 DISCUSSION

This section reflects upon our measurement results, discusses performance implications, and provides recommendations for improved Zoom performance on our campus network.

5.1 Reflections

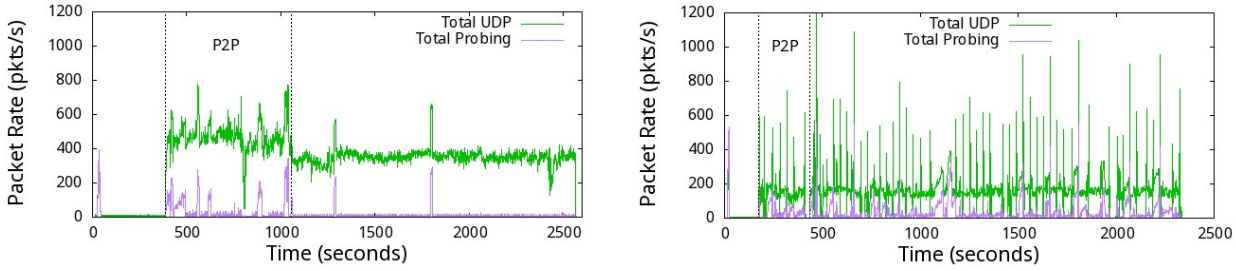
The methods used in our packet-level analysis are simple and generally applicable on any network where Zoom performance is a concern. If a performance bottleneck is due to the local network, then our study acts as a use case for a similar investigation. If a performance bottleneck is due to Zoom infrastructure, then our study again acts as a use case for how to investigate it, and as evidence for encouraging Zoom to make design changes to their product.

One takeaway message from our work is the immense value of Wireshark for the performance debugging of network applications.



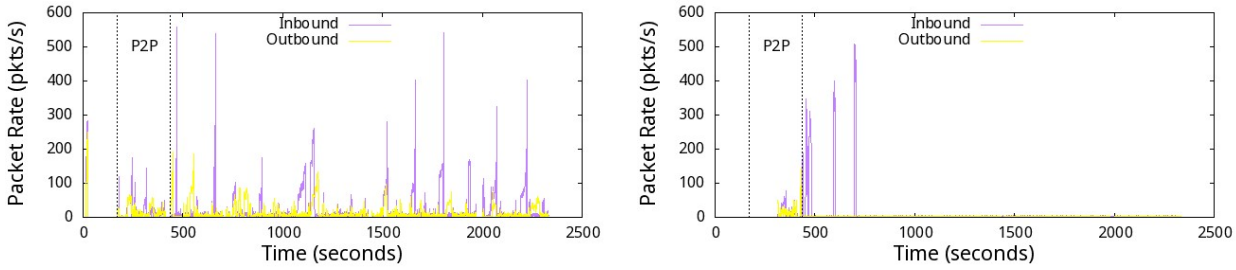
(a) Trace C1 (Campus) (b) Trace C2 (Home)

Figure 9: Connection profiles for P2P Zoom test session (October 13, 2021)



(a) Trace A (b) Trace D1

Figure 10: Analysis of video bandwidth probing traffic in Zoom test sessions



(a) Trace D1 (Campus) (b) Trace D2 (Home)

Figure 11: Analysis of video bandwidth probing traffic in Zoom test session (October 27, 2021)

Other than the campus backbone traffic results in Figure 3, all of our data collection and analysis was done using Wireshark. This in-depth analysis offered insights into the structure of Zoom traffic and some of its algorithmic behaviors.

A second takeaway is the importance of understanding new network applications when they are deployed on existing network infrastructures. Because Zoom generates high-rate UDP traffic that is not subject to congestion control, it behaves quite differently from other protocols or applications, such as Web browsing, QUIC, or TCP-based on-demand video streaming (e.g., Netflix, YouTube). Zoom traffic definitely increases congestion on our campus network, and takes bandwidth away from other network applications. Furthermore, Zoom's video bandwidth probing is overly aggressive, and exacerbates the performance problems on our campus network.

5.2 Performance Implications

Two themes pervade our network traffic measurement results: one is scale, and the other is multi-layer protocol interactions.

Regarding scale, Zoom worked fine on our campus in Fall 2020 when almost everyone was at home, because Zoom traffic bypassed the campus network. However, Zoom performance suffered in Fall 2021 when half of our user community returned to campus. The increased traffic volume from on-campus users, as well as Zoom itself, saturated the external Internet link, causing congestion for all campus network users. Zoom's response to this congestion was very different from other protocols and applications, resulting in increased bandwidth probing, large traffic spikes, and poor Zoom session quality. These behaviors only emerged at scale, with thousands of collocated Zoom users.

Multi-layer protocol interactions were also at play here. Some of these involve the combination of TCP and UDP to manage Zoom sessions. Others involve the use of WiFi for many campus users. The most interesting, however, were the failed TCP and UDP connections, due to network congestion, and Zoom's application-layer response to this congestion, which involved more TCP connections, more TLS handshakes, and increased UDP bandwidth probing. These actions, coupled with potential synchronization across users, led to numerous repeated TLS handshakes at the server and increased traffic spikes from bandwidth probing. These effects further degraded Zoom performance on our campus network.

5.3 Recommendations

We have several recommendations for improving Zoom performance on our campus network, or any enterprise network for that matter. These recommendations fall into two categories: those for network operators, and those for Zoom itself.

Regarding our campus network, nobody within the university foresaw the potential performance implications of Zoom in a hybrid learning scenario, with half of our students on campus. Furthermore, the external Internet link was congested well before the pandemic occurred. The most obvious technical solution would be to route Zoom traffic over the uncongested link for research and education traffic, rather than the congested link for commercial Internet traffic. Since our campus Zoom traffic involves only a handful of network prefixes, Zoom could be manually configured into the network traffic management policies. A second solution, though more expensive, would be to upgrade the external commercial Internet link, say from 6 Gbps to 10 Gbps. This should provide adequate room for growth in Internet traffic over the next few years, whether for Zoom or other network applications, though it does not address Zoom's underlying behavioral properties.

Regarding Zoom itself, three recommendations come to mind. First, better load balancing across a larger pool of Zoom servers would reduce the likelihood of a single server hosting multiple large meetings at the same time. This would help diffuse the load and potential synchronization issues identified in our paper. Second, bandwidth probing should be less aggressive, particularly on congested network paths. Ideally, this probing should be done on a per-network-prefix basis, rather than a per-user or per-IP basis, which would reduce the volume dramatically. Sharing such state information across multiple meetings hosted by the same Zoom server would also help. Furthermore, randomization in the timing and duration of probing might ameliorate some of the traffic synchronization problems observed. Last, but not least, the Zoom application seems like a natural fit for classic networking protocols such as IP multicast, Protocol Independent Multicast (PIM) [6], and Scalable Reliable Multicast (SRM) [14, 15], or emerging protocols like QUIC [7], which also support multicast. Such an approach would help alleviate network congestion in the case of multiple collocated clients for a Zoom meeting, which was the common case on our campus network. Finding a practical solution for Zoom multicast delivery would be a worthwhile pursuit.

6 CONCLUSIONS

The main conclusions from our work are as follows. First, packet-level analysis of Wireshark traces can provide deep insights into the structure and performance of Zoom. Our analysis exploited opcodes, media types, sequence numbers, and timing probes to quantify delay, jitter, loss, and retransmission behavior in Zoom's UDP and TCP traffic. Second, Zoom performance suffers when traversing a congested network. The 6 Gbps limit on the external link of our campus network is the root cause of the Zoom performance issues we experienced in Fall 2021, when hybrid learning began. Third, the Zoom application itself is partly to blame for the problems that we observed. In particular, Zoom's aggressive bandwidth probing exacerbates network congestion. Finally, multi-layer protocol interactions can lead to further performance issues with Zoom. For example, excessive TCP retransmissions can cause connection-level restarts and repeated TLS handshakes at Zoom servers. These problems are most evident when multiple collocated Zoom users are sharing the same congested network path.

Our ongoing work is investigating empirically-observed performance differences between wired and wireless Zoom users, as well as the dynamics of Zoom's bandwidth probing algorithm.

ACKNOWLEDGMENTS

The authors are grateful to Albert Choi for his undergraduate honours project work [4], which laid the foundation for our packet-level analysis of Zoom. The authors also thank UCIT for access to long-term campus network traffic measurement data. Financial support for this research was provided via Canada's Natural Sciences and Engineering Research Council (NSERC), as well as the Department of Computer Science at the University of Calgary.

APPENDIX: WIRESHARK ANALYSIS TIPS

For those unfamiliar with Wireshark, this appendix provides a few tips on how to work with large packet-level Zoom traces, such as those used in this paper. There are also several good Wireshark tutorials and videos available online [25]. Our Top 10 tips follow:

(1) When collecting and saving a Wireshark trace, always check the `Capture File Properties` for metadata about your packet capture, such as date, time, duration, number of captured packets, lost packets, etc. This feature is available under the `Statistics` tab, which offers the main menu for traffic analysis.
(2) When analyzing a trace, the `Conversations` tool is a good starting point. Select the protocol of interest (e.g., TCP, UDP, IP) for a statistical overview. The displayed data can be sorted easily (ascending or descending) by clicking on the desired column (e.g., packets, bytes, time, IP address) to find the flows, ports, and IP addresses of interest.
(3) By default, the main Wireshark window displays your entire packet trace, but you can apply a display filter to restrict this to specific IP addresses, protocols, or ports if you wish (e.g., `tcp`, `udp`, `dns`, `udp.port==8801`, `ip.srcaddr==10.0.0.12 && udp.dstport==8801`). The status line at the bottom always tells you how many packets are displayed. If you text select any packets, it will also show how many are selected.
(4) If you click on a specific packet in the trace, the window beneath shows you the detailed content of that packet. You can select different layers of protocol headers to study here, as well as the application-layer payload.

(5) You can export displayed packets to a file, using a variety of different formats (e.g., PCAP, JSON, CSV, XML, text). This function is available under the `File` menu, via either `Export Specified Packets` or `Export Packet Dissections`. This process may be slow if the capture file is large. On a Linux system, the `editcap` utility can be used to manipulate a PCAP file, such as trimming large packet payloads to a specified snapshot size, or only keeping packets that match particular IP addresses, protocols, or ports.
(6) Use the `I/O Graph` feature for a quick graphical overview of your captured traffic. By default, it shows all traffic, but you can also apply filters here to visualize (for example) TCP and UDP traffic separately, or inbound and outbound traffic separately. In addition to filtering on protocols, ports, and addresses, you can also filter on data content in the payload (e.g., `udp.port==8801 && data[0]==0x05`). The same filters used here can also be used in the main window for packet selection.
(7) If you like a graph that you made, you can export it directly as a PDF or several other formats. If you just want the data points and not the graph itself, you can export as a CSV file instead, and then draw the graph yourself using other tools.
(8) If you are interested in the details of a particular TCP flow, you can use the `TCP Stream Graphs` feature to draw a sequence-number plot, throughput plot, or RTT plot. Mouse clicks can be used to select a flow, toggle between the two directions on a TCP connection, or even scroll to the next TCP stream flow in your trace.
(9) Wireshark may misinterpret some Zoom UDP packets (e.g., opcode 0x04) as WireGuard VPN packets, based on its header detection heuristics for WireGuard. If this happens, you can force Wireshark to consider them as UDP packets using the `Analyze -> Decode As` feature.
(10) When working with multiple trace files that were collected concurrently from different vantage points, the `View -> Time Display Format` feature is useful for switching between relative and absolute timestamps. Establishing proper time synchronization between multiple traces is helpful when looking for Zoom packets, since the start times and durations of the Wireshark traces will differ a bit.

A video demo of several of these Wireshark tips is available at `pages.cpsc.ucalgary.ca/~mehdi.karamollahi/Wireshark-Demo.mp4`

REFERENCES

[1] Timm Böttger, Ghida Ibrahim, and Ben Vallis, "How the Internet Reacted to Covid-19: A Perspective from Facebook's Edge Network", *Proceedings of the ACM Internet Measurement Conference*, Pittsburgh, PA, virtual event, pp. 34-41, October 2020. doi:10.1145/3487552.3487842

[2] Ashley Carman, "Why Zoom Became So Popular", https://www.theverge.com/2020/4/3/21207053/zoom-video-conferencing-security-privacy-risk-popularity

[3] Hyunseok Chang, Matteo Varvello, Fang Hao, and Sarit Mukherjee, "Can You See Me Now? A Measurement Study of Zoom, Webex, and Meet", *Proceedings of the ACM Internet Measurement Conference*, virtual event, pp. 216–228, November 2021. doi:10.1145/3487552.3487847

[4] Albert Choi, "Analysis of Zoom Network Traffic", CPSC 502 Honours Project, Department of Computer Science, University of Calgary, 8 pages, April 2022.

[5] Albert Choi, Mehdi Karamollahi, Carey Williamson, and Martin Arlitt, "Zoom Session Quality: A Network-Level View", *Proceedings of Passive and Active Measurement (PAM) Conference*, virtual event, Spinger LNCS Vol. 13210, pp. 555–572, March 2022. doi:10.1007/978-3-030-98785-5_25

[6] Stephen Deering, Deborah Estrin, Dino Farinacci, Van Jacobson, Ching-Gung Liu, and Liming Wei, "An Architecture for Wide-Area Multicast Routing", *Proceedings of ACM SIGCOMM Conference*, London, UK, pp. 126-135, August 1994. doi:10.1145/190314.190326

[7] Mathis Engelbart and Jörg Ott, "Congestion Control for Real-Time Media over QUIC", *Proceedings of the 3rdWorkshop on the Evolution, Performance, and Interoperability of QUIC (EPIQ)*, Munich, Germany, virtual event, pp. 1-7, December 2021. doi:10.1145/3488660.3493801

[8] Thomas Favale, Francesca Soro, Martino Trevisan, Idilio Drago, and Marco Mellia, "Campus Traffic and E-Learning during COVID-19 Pandemic", *Computer Networks*, Vol. 176, Article 107290, pp. 1-9, July 2020. doi:10.1016/j.comnet.2020.107290

[9] Nick Feamster, "2020 Pandemic Network Performance", *Broadband Internet Technical Advisory Group*, 2021.

[10] Anja Feldmann, Oliver Gasser, Franziska Lichtblau, Enric Pujol, Ingmar Poese, Christoph Dietzel, *et al.*, "The Lockdown Effect: Implications of the COVID-19 Pandemic on Internet Traffic", *Proceedings of the ACM Internet Measurement Conference*, Pittsburgh, PA, virtual event, pp. 1-18, October 2020. doi:10.1145/3419394.3423658

[11] Sally Floyd and Van Jacobson, "The Synchronization of Periodic Routing Messages", *Proceedings of ACM SIGCOMM Conference*, San Francisco, CA, pp. 33-44, September 1993. doi:10.1145/166237.166241

[12] Mehdi Karamollahi, Carey Williamson, and Martin Arlitt, "Zoomiversity: A Case Study of Pandemic Effects on Post-Secondary Teaching and Learning", *Proceedings of Passive and Active Measurement (PAM) Conference*, virtual event, Spinger LNCS Vol. 13210, pp. 573–599, March 2022. doi:10.1007/978-3-030-98785-5_26

[13] Mehdi Karamollahi and Carey Williamson, "Simulation Modeling of Zoom Network Traffic", *Proceedings of IEEE International Symposium on the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Nice, France, hybrid event, pp. 57-64, October 2022.

[14] Sneha K. Kasera, Jim Kurose, and Don Towsley, "Scalable Reliable Multicast using Multiple Multicast Groups", *Proceedings of ACM SIGMETRICS Conference*, Seattle, WA, pp. 64-74, June 1997. doi:10.1145/258612.258676

[15] Ching-Gung Liu, Deborah Estrin, Scott Shenker, and Lixia Zhang, "Local Error Recovery in SRM: Comparison of Two Approaches", *IEEE/ACM Transactions on Networking*, Vol. 6, No. 6, pp. 686-699, December 1998. doi:10.1109/90.748082

[16] Andra Lutu, Diego Perino, Marcelo Bagnulo, Enrique Frias-Martinez, and Javad Khangosstar, "A Characterization of the COVID-19 Pandemic Impact on a Mobile Network Operator Traffic", *Proceedings of the ACM Internet Measurement Conference*, Pittsburgh, PA, virtual event, pp. 19-33, October 2020. doi:10.1145/3419394.3423655

[17] Kyle MacMillan, Tarun Mangla, James Saxon, and Nick Feamster, "Measuring the Performance and Network Utilization of Popular Video Conferencing Applications", *Proceedings of the ACM Internet Measurement Conference*, virtual event, pp. 229–244, November 2021. doi:10.1145/3487552.3487842

[18] Andrew Mahr, Meghan Cichon, Sophia Mateo, Cinthya Grajeda, and Ibrahim Baggili, "Zooming into the pandemic! A forensic analysis of the Zoom Application", *Forensic Science International: Digital Investigation*, Vol. 36, pp. 301107, March 2021. doi:10.1016/j.fsidi.2021.301107

[19] Bill Marczak and John Scott-Railton, "Move Fast and Roll Your Own Crypto: A Quick Look at the Confidentiality of Zoom", Citizen Lab, University of Toronto, April 2020. https://citizenlab.ca/2020/04/move-fast-roll-your-own-crypto-a-quick-look-at-the-confidentiality-of-zoom-meetings/

[20] Oliver Michel, Satadal Sengupta, Hyojoon Kim, Ravi Netravali, and Jennifer Rexford, "Enabling Passive Measurement of Zoom Performance in Production Networks", *Proceedings of ACM Internet Measurement Conference*, Nice, France, pp. 244-260, October 2022. doi:10.1145/3517745.3561414

[21] Antonio Nisticò, Dena Markudova, Martino Trevisan, Michela Meo, and Giovanna Carofiglio, "A Comparative Study of RTC Applications", *Proceedings of IEEE International Symposium on Multimedia*, Naples, Italy, pp. 1-8, December 2020. doi:10.1109/ISM.2020.00007

[22] Vern Paxson, "Bro: A System for Detecting Network Intruders in Real-time", *Computer Networks*, Vol. 31, No. 23, pp. 2435–2463, Dec 1999. doi:10.1016/s1389-1286(99)00112-7

[23] Constantin Sander, Ike Kunze, Klaus Wehrle, and Jan Rüth, "Video Conferencing and Flow-Rate Fairness: a First Look at Zoom and the Impact of Flow-Queuing AQM", *Proceedings of Passive and Active Measurement (PAM) Conference*, virtual event, Springer LNCS Vol. 12671, pp. 3–19, March 2021. doi:10.1007/978-3-030-72582-2_1

[24] Henning Schulzrinne, Stephen L. Casner, Ron Frederick, and Van Jacobson, "RTP: A Transport Protocol for Real-Time Applications", IETF RFC 3550, July 2003. doi:doi.org/10.17487/rfc3550

[25] Wireshark.org, Wireshark Frequently Asked Questions, https://wireshark.org/faq.html

[26] Zoom, Network and Firewall Settings for Zoom. https://support.zoom.us/hc/en-us/articles/201362683-Network-firewall-or-proxy-settings-for-Zoom

[27] Zoom, https://www.zoom.us/docs/doc/ZoomConnectionProcessWhitepaper.pdf