M-Lab Fellowship Report

SANJAY CHANDRASEKARAN, University of California Santa Barbara, USA
IRENE PATTARACHANYAKUL, University of California Santa Barbara, USA
PUNNAL ISMAIL KHAN, University of California Santa Barbara, USA
ETHAN WU, University of California Santa Barbara, USA
SAM LIANG, University of California Santa Barbara, USA
ELIZABETH BELDING, University of California Santa Barbara, USA
ARPIT GUPTA, University of California Santa Barbara, USA

1 INTRODUCTION

In a post-pandemic world, we have become heavily reliant on networked applications such as video streaming (e.g., YouTube [8], Nextflix [12], Twitch [17]) and video conferencing (e.g., Zoom [18], Teams [10], Google Meet [7]). These applications must use feedback about the state of congestion in the network to provide a high quality of experience (QoE) to the end user. These developments have shifted the focus toward last-mile networks, which are generally the least-provisioned part of the network. Recent technological innovations enable these last-mile service providers to support high-throughput and low-latency. However, the traffic in last-mile networks is still susceptible to congestion events [5], caused by issues such as bufferbloat [6] that contribute to latency inflation and as a result, degrade the QoE of the application [11].

Our proposal focused on two main research questions:

(1) Can we characterize any biases within M-Lab's active measurements?
(2) Can we draw a relationship between QoE and lightweight metrics, collected through both active and passive techniques?

To this end, we provide insight into potential biases and limitations of active measurements due to the duration and frequency of ephemeral events. We also analyze the correlation between downstream RTT inflation and the QoE of a common video conferencing applications, Google Meet.

We first focus on characterizing the presence of ephemeral congestion events in our campus network that spans multiple residential buildings and academic facilities. The goal of this characterization is to understand potential biases and limitations when using active measurements to detect ephemeral congestion events. We use packet-level traces that are passively collected from the network's border router that sees all traffic entering or leaving the network. We extract last-mile RTT measurements from these packet traces by computing the delays that packets experience between end-hosts and the campus' border router. We use this last-mile RTT metric as a proxy for congestion in the last-mile network.

Next, we develop an active measurement framework using Raspberry Pis [14] deployed around campus to run video conferencing sessions while recording the QoE of these applications. We use this setup to understand the correlation between the presence of ephemeral events and degradations in application QoE (e.g., lower resolution, loss of frames, rebuffering).

Our main contribution is to highlight the presence of ephemeral congestion events in last-mile networks to help refine our methodology for active measurements and network monitoring. Our analysis shows that almost 23% of active hosts observe at least one ephemeral event in an hour. At any time instance, approximately 150 (around 0.3%) hosts experience a short-lived congestion event. Moreover, we show that the distributions of both the duration of an ephemeral congestion event

and the length of the gap between two such events exhibit heavy-tailed behavior. This observation implies that most of these events are very short-lived.

2 PASSIVE MEASUREMENTS

In this section, we describe the passive measurements we used for our analysis along with the methodology for extracting the last-mile RTT samples associated with each active host. Additionally, we are able to map hosts to different access points, rooms, and buildings within our campus network to understand the spatial nature of latency inflations.

2.1 Collecting packet traces

To collect packet traces from our campus network's border router, we mirror traffic from our border routers to an Intel Tofino-based programmable switch [2] that anonymizes and load balances traffic between our collection servers. The collection servers use `tcpdump` to capture and save the anonymized packet traces. However, for each captured packet, instead of using the timestamp that is generated locally by `tcpdump`, we rely on the hardware timestamp added by our switch to each packet's metadata. This approach ensures that the timing information and ordering for each packet are as close as possible to ones observed at the border router and that the timing information itself is not skewed by local processing delays and clock skews.

The average data rate at our border router was approximately 9 Gbps, which translates to more than a million packets per second. Load-balancing the traffic over eight different ports (on two different servers) ensured minimal packet drops at the collection servers because each core had to processes at most 150 k packets/second. For our passive data analysis, we used packet traces collected from four different weekdays (4/26/22-4/29/22[1]); the hour was always from 12:15 am to 1:15 pm, at a time when we noticed peak activity on our campus network. Applying this data-collection pipeline, we collected data from approximately 54 k distinct internal hosts that communicated with about 924 k distinct external hosts, generating over 91 M different flows and exchanging a total of 16.5 B packets.

2.2 Extracting last-mile RTTs.

To extract last-mile RTT samples from the collected packet traces, we use the difference in timestamps between the downstream TCP data packet and the corresponding upstream acknowledgement packet. However, when computing these last-mile RTT samples, we have to filter out samples that are not representative of last-mile network delays. In particular, we need to filter out RTT samples for retransmitted packets, delayed ACKs, and selected/stretched ACKs [3]. Out of a total of 11.6 B TCP packets in the dataset, we curated a dataset with 1.6 B (14%) valid last-mile RTT samples after filtering. The median last-mile RTT for our dataset is 12 ms and for more than 5 % of the samples, we observe last-mile RTTs inflated by an order of magnitude.

While it is possible to leverage existing tools such as `tcptrace` [9] for filtering RTT samples, we developed a custom Spark-based [1] streaming analytics pipeline for this task. Compared to `tcptrace`, our pipeline effectively uses all the available CPU cores and is able to perform the task of extracting and filtering last-mile RTT samples from our packet traces at least an order of magnitude faster than `tcptrace`. Below we discuss how we sanitized these RTT values to accurately represent the downstream RTT for each host.

2.2.1 Filter out delayed ACKs. TCP allows a receiver to reduce the number of acknowledgements by delaying the transmission of an acknowledgement for data packets whose size is not equal to the maximum MTU. Any such delays in ACK delays will skew our estimation of network-level

[1] All classes were in session during this week.

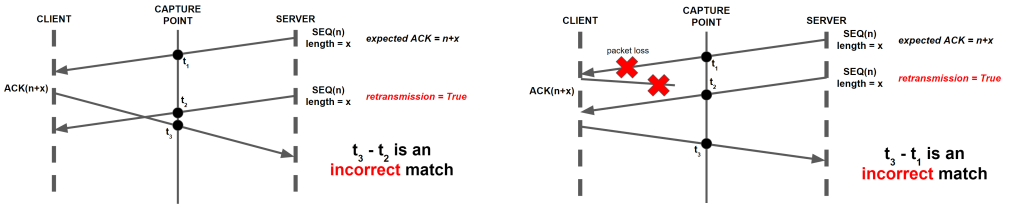


Fig. 1. Cases where retransmissions can cause incorrect RTT samples

delay. Thus, we only consider samples for full-sized packets. Also, TCP uses the PSH flag to indicate that the receiver should not wait for additional data. Thus, since the presence of a PSH flag ensures that the RTT sample is only measuring the network-level delays, we consider all samples where the PSH flag is set for the downstream data packet, even if it is not full-sized. As a result, we filtered 2.5 % of RTT samples from this step.

2.2.2 Disregard retransmitted packets. When capturing packets between a sender and receiver, we cannot determine the exact times when the packets are received and sent by the downstream host. For example, a sender may choose to retransmit data based on a timeout or because of a request from the receiver. In this case, we cannot determine if an ACK was sent by the receiver before or after it received the retransmitted portion of the data. To avoid over-estimating last-mile latency due to this uncertainty, we do not compute RTT samples for any data packet that is retransmitted. These cases are further illustrated in Figure 1. As a result of this step, we further filtered out 0.37 % of the total RTT samples. Note that we should expect concurrence of latency inflations and packet losses. Thus, we might miss out on reporting some of the transient congestion events for a subset of hosts that see high retransmission rates. f

2.3 Adding Spatial Attributes

Ephemeral congestion events have temporal as well as spatial attributes, and characterizing these events requires understanding both when (in time) and where (in space) they occur and how long they last. To characterize the spatial behavior of these congestion events, we map end-hosts to different coarser spatial granularities, such as access points (APs), rooms, and buildings. To this end, we use Aruba's Advanced Monitoring (AMON) data that enables end-host mapping to APs. The naming convention for each AP on our campus network uniquely specifies its location attributes, such as room and building identifiers. We used this information to map each host to a specific room and building. Our campus network has deployed approximately 4 k APs in 2.5 k separate rooms across 250 buildings. Out of a total of 54 k local hosts, we are able to map about 23 k (i.e., 43 %) to an AP.

3 ACTIVE MEASUREMENTS

In this section we describe our active measurement setup using Raspberry Pi nodes that we deployed across our campus. Each device records the QoE of its video conferencing session using the webRTC stats, as well as passively captures all incoming and outgoing packets.

3.1 Deploying end hosts

We deployed Raspberry Pi devices around our campus to mimic end hosts on the network and provide us with ground truth about the quality of experience for video conferencing and video streaming applications. Each of these devices is managed using a centralized SaltStack [15] server. We created automated programs with Selenium to control browser functionality for running

applications on the Raspberry Pis. We used power-over-ethernet for a subset of device deployments where there are limited power outlets.

3.2 Measuring QoE

For this study, we used QoE measurements from two video conferencing applications: Google Meet and Microsoft Teams. For each video conferencing session, we host a call playing a looped video. Each raspberry pi is ordered to join the call with the microphone muted and no video feed. Using the webRTC stats from the browser, we can see measure attributes of the call at a 1-second granularity. We focus on the frames-per-second, frames dropped, and rebuffering time which we use as a proxy for the QoE of the video call.

4 EPHEMERAL CONGESTION EVENTS

This section will first describe our methodology for identifying ephemeral congestion events. We then characterize their spatial and temporal properties that can inform the future design of telemetry systems capable of detecting these events in last-mile networks in (near) real-time.

4.1 Defining ephemeral events

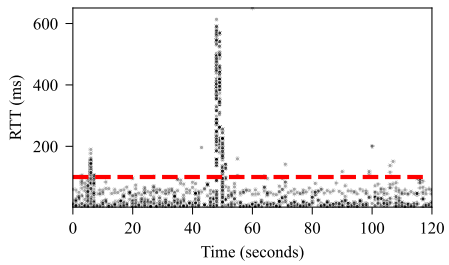
Informally, for last-mile networks, *an ephemeral congestion event is a short period of time during which there is an inflation in (last-mile) RTT that is high enough to adversely affect user-perceived QoE for video applications* (e.g., video streaming, video conferencing, online gaming, etc.) that are running on end hosts inside the last-mile network.

To formally define an ephemeral congestion event for a given host in a last-mile network (i.e., unique `dstIP`), we first generate a time series that consists of all the valid RTT samples associated with the downstream host's data packets. For this time-series, we seek to identify periods of time (windows) when the inflation in RTT values is "abnormally" high. To this end, we window the time series data and compute a representative aggregate metric. Deciding on an appropriate window size requires making a compromise. On the one hand, the ability to detect short-lived congestion events argues for selecting as small a window as possible (i.e., tens or hundreds of msec). On the other hand, to ensure that there are sufficiently many samples in a window for identifying such an event with high confidence requires considering larger window sizes (i.e., seconds).

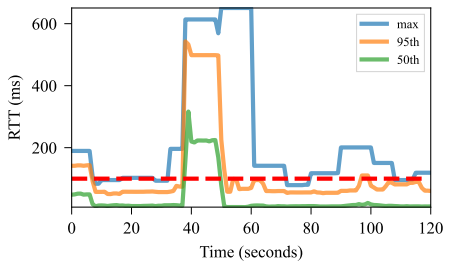
Note that our decision to ignore windows with fewer samples might contribute to the under-reporting of transient congestion events. However, in the absence of any ground truth, we argue that with these choices, we can ensure that the reported events are largely insensitive to noisy observations, possibly at the cost of not detecting all possible ephemeral congestion episodes. Our current methodology will also miss reporting ephemeral events experienced by hosts that mostly send (or receive) UDP packets[2]. Also note that our definition of ephemeral congestion events is inclusive in the sense that it also accounts for instances of congestion that are long-lived and may reflect more conventional episodes of congestion.

To decide which representative aggregate metric to compute and report for each window, Figure 2a shows the time series of RTT samples for a randomly selected host from our last-mile RTT dataset. We observe that for a majority of the samples, the last-mile RTT values are in the the $10 - 25$ ms range. However, about 5% of RTT samples appear to be inflated by over one order of magnitude. While fully consistent with the well-known bursty nature of Internet traffic, we expect such random instances of significant RTT inflation to have only a marginal impact on QoE for most applications, unless the fraction of the total number of samples in a window that experience this type of RTT inflation is significant (at least 1% for video conferencing applications [13]). This

[2]Most video conferencing applications use UDP as transport protocol.



(a) Raw samples.



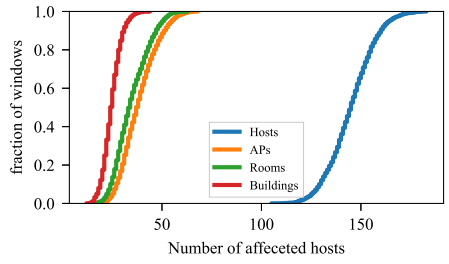
(b) Aggregate statistics.

observation suggests that we consider large quantiles (e.g., 95^{th} percentile) of the empirical distribution of RTT values as the representative aggregate statistic for identifying ephemeral congestion events. To illustrate, Figure 2b shows the median (50^{th} percentile), 95^{th} percentile, and max (100^{th} percentile) of the last-mile RTT values using a ten-second sliding window for the same host. We observe that while using extreme quantiles (e.g., max) for quantifying RTT inflation results in overly noisy reporting, utilizing mid-range quantiles such as the median is not sufficiently sensitive to identify short-lived instances of RTT inflation. Thus, we use the 95^{th} percentile value, ensuring that we observe at least 5 inflated samples while reporting a RTT inflation event.

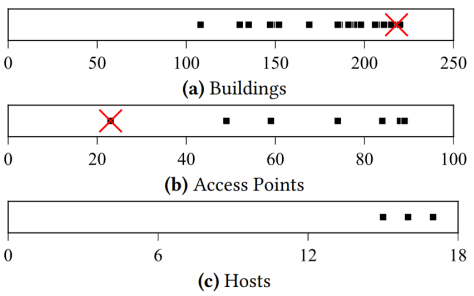
Previous studies showed that last-mile RTTs greater than 100 ms can severely degrade QoE for most real-time applications [13, 16]. We report all ten-second sliding windows, sliding one second at a time, with more than a hundred samples for which the 95^{th} percentile of last-mile RTT values exceeds 100 ms as the ones experiencing an ephemeral congestion event.

4.2 Characterizing Ephemeral Events

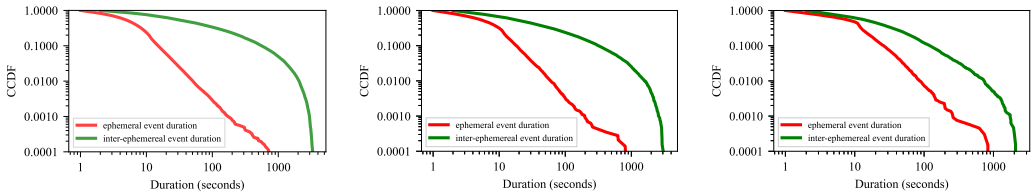
Spatial attributes. Given our formal definition of what constitutes an ephemeral congestion event, we first analyze our last-mile RTT dataset to understand and quantify the pervasiveness of these events in terms of affected hosts. To this end, we compute the total number of distinct ephemeral events experienced by each active host in one hour. We observe that almost 23% (about 13 k) of all active hosts experienced at least one ephemeral event, indicating that these events are not spatially skewed (i.e., limited to a small number of hosts). Similarly, at coarser spatial granularity, 63% (i.e., 1.5 k) of APs experience at least one ephemeral event. Here, we mark a window as ephemeral for an AP if the 95^{th} percentile of all valid RTT samples from all hosts associated with this AP during that window exceeds 100 ms. Given that there is a single AP in most rooms on our campus network,



(a) More than hundred hosts experience ephemeral congestion events at any given time.



(b) Spatial behavior exhibited by ephemeral events.



(a) Hosts (b) APs (c) Buildings

Fig. 4. Temporal behavior exhibited by ephemeral events.

the findings at the granularity level of rooms are similar to those at the AP-level, which in turn have the same trend as those at the level of individual buildings.

In view of the pervasive nature of the identified ephemeral congestion events in our campus network, we next examine whether ephemeral events are skewed towards certain time intervals. To this end, we compute for each window the number of hosts/APs/rooms/buildings that experience ephemeral events in that window and show in Figure 3a the resulting distribution functions. For example, we observe between 120 – 160 (around 0.3%) affected hosts in each window. Moreover, even though the number of affected hosts is relatively stable over time (i.e., around 150), the set of hosts experiencing ephemeral events keeps changing over time (not shown here).

To better understand the spatial structure of these ephemeral events, we explore if the ephemeral congestion events exhibit a "cluster-within-cluster" structure. Specifically, we check whether the measured events at a building are attributable to a cluster of APs within that building. Similarly, whether the observed events at an individual AP are attributable to a cluster of hosts associated with that AP. Figure 3b shows this behavior for a randomly selected time window. We observe that a small set of buildings contribute to the ephemeral congestion events at the campus level. For a randomly selected such building (marked red), only a small subset of APs experience ephemeral congestion events, and for a randomly chosen such AP (marked red), only a small subset of hosts suffer from ephemeral congestion events. We confirmed that this cluster-within-cluster property holds irrespective of our choice of window or selection of building or AP.

Temporal attributes. Next, focusing our attention on the temporal behavior exhibited by the measured ephemeral congestion events, we are interested in two metrics: (1) affected duration, i.e., the length of ephemeral events) and (2) unaffected duration, i.e., the gap between two ephemeral events. Figure 4a shows the distribution of these metrics for all ephemeral events across all hosts. The graph shows the complementary CDFs (CCDFs) on a log-log scale, where an approximately straight line is a strong indication of heavy-tailed behavior [4]. From Figure 4a, we can conclude that the durations of ephemeral congestion events exhibit heavy-tailed behavior; that is, most events are short-lived with a median value of about 5 seconds and a small fraction of events (some 0.7%) last for more than one minute. This finding implies that network telemetry systems for last-mile networks have to be able to monitor last-mile RTTs at very small time scales to accurately detect ephemeral congestion events.

We observe skewness or high variability in the metric that quantifies the gap between two ephemeral congestion events is not consistent with heavy-tailed behavior. However, the saturation near one hour at the right tail of the CCDF is an artifact of the finite time horizon of the dataset. We expect that repeating this analysis for datasets that have been collected for longer periods of time will also show heavy-tailed behavior for the lengths of the gaps between two ephemeral congestion

		Host	AP	Room	Building
Percentile of window Default: 95	93	−0.92	−0.86	−0.84	−0.77
	94	−0.92	−0.85	−0.84	−0.76
	96	−0.92	−0.85	−0.83	−0.76
	97	−0.92	−0.85	−0.83	−0.75
Sliding window (s) Default: 10	8	−0.97	−0.90	−0.88	−0.81
	9	−0.95	−0.87	−0.86	−0.78
	11	−0.91	−0.83	−0.82	−0.75
	12	−0.89	−0.81	−0.80	−0.73
RTT Threshold (ms) Default: 100	90	−0.92	−0.85	−0.83	−0.76
	95	−0.92	−0.85	−0.83	−0.76
	105	−0.93	−0.85	−0.83	−0.76
	110	−0.93	−0.85	−0.83	−0.76

Table 1. Sensitivity analysis: shows slope for ephemeral events' distribution (CCDF) on a log-log plot. We observe heavy-tailed behavior for all combinations of parameters at all four spatial granularities.

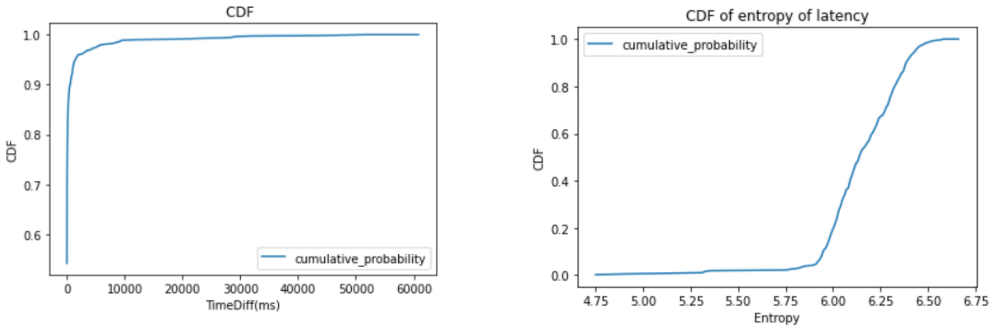
events. This finding would imply that the likelihood of seeing an ephemeral event is higher if we have observed one in the recent past compared to not having observed an event for some time.

Figure 4b and Figure 4c show the CCDFs of these two metrics at the granularity of APs and buildings, respectively. We do not include the result for rooms because it mimics the trend for APs. We observe that even though when compared to Figure 4a, the shape of these distributions is slightly different, the same overall findings apply; that is, the considered metrics are heavy-tailed, implying that ephemeral congestion events are scale-invariant in time, irrespective the level of spatial granularity. Together, these empirical observations motivate further studies into the fractal-like spatial-temporal structure of measured ephemeral congestion events in last-mile networks.

Sensitivity analysis. We need to ensure that the above observations are not an artifact of our parametric choices. More specifically, we are interested in demonstrating that the heavy-tailed and spatial-invariance behavior is not sensitive to our specific parameters. To this end, we run an extensive sensitivity analysis, where we perturb each of the three parameters, i.e., window length (10±2 s), percentile (95±2%), and threshold (100± 10 ms), one at a time. For each parameter, we report the slope of the line for ephemeral events' duration on the log-log plot. Our analysis shows that both the heavy-tailed and scale-invariance behavior are insensitive to specific parameters (see Table 1 in Appendix). We also confirmed that the spatial properties are also agnostic to these parameters.

4.3 Detecting ephemeral events

Now that we understand the nature of ephemeral events, we can try to understand how we can detect them using active measurements. For each host experiencing an ephemeral event, we try to understand the right sampling frequency and duration needed to still identify congestion events, with the intuition that we can determine the Nyquist rate for these RTT values if the RTT inflations are not abrupt. For each host, we measure the time between RTT samples transitioning between 10 ms and 100 ms. A CDF of this duration for all hosts is shown in Figure 5a. We observe this transition between normal and inflated RTTs occurs within a very short period of time, making it difficult to measure with active measurements. Figure 5b measures the slope for each of these transitions to see how gradual the RTT inflation is. The average transition happens in less than a second making active measurements infeasible for monitoring such ephemeral events.



(a) Time difference

(b) Entropy

Fig. 5. Measuring transition between inflated (greater than 100ms) and non-inflated (less than 10ms) states

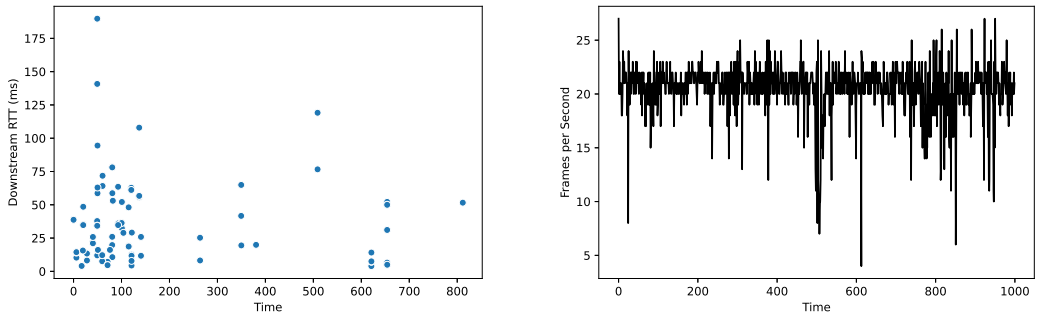
5 IMPACT ON QOE

In this section we present a case study for a single Raspberry Pi deployed in the library of UCSB's campus that joins a Google Meet video call and receives poor QoE. We capture data passively alongside running active measurements on a raspberry pi. We narrow our focus to just one Raspberry Pi due to the processing time for the passive measurements, requiring us to filter on just the hosts that we control. With more time, we will extend this analysis to higher granularities such as access point or building, which will require us to compute RTTs for all hosts in the data. At first when we try to apply our ephemeral event methodology, we run into a limitation. Our calculation of downstream RTT is only applicable to TCP packets however a majority of real-time video conferencing sends data over UDP. Understanding how to apply our ephemeral event methodology to hosts sending mostly UDP traffic is still an open task that we are working on. Currently, we utilize RTT samples from other TCP flows originating from the same host to proxy for congestion that the UDP flows would also receive. On our raspberry pi joining a Google Meet session for 15 minutes, we only observe 96 valid RTT samples over the entire call which doesn't suffice for our ephemeral event methodology. In Figure 6 we show the raw RTT values over time along with the QoE measurements from the webRTC stats including frames per second, dropped frames, and rebuffering time. We can see at around 500 seconds latency inflations occur for the TCP samples we can measure and as a result we see reduced FPS, more frames dropped, and rebuferring occur around that time.

6 LIMITATIONS & FUTURE WORK

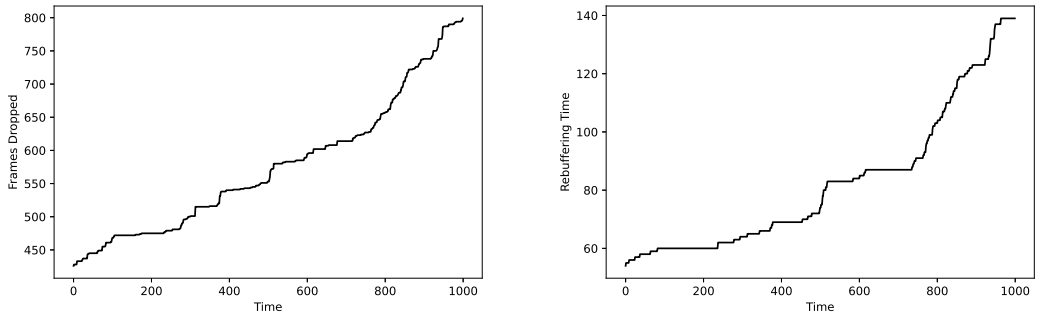
Developing our active measurement pipeline to get ground truth measurements within our campus network has been challenging. Getting coverage over different parts of our campus took time as supply chain issues caused vendors to be out of stock of raspberry pis. Over the past 4 months we have accumulated over 100 raspberry pi devices for our deployment. Additionally, automating the browser interactions with video conferencing applications can be fragile. Applications update the HTML on their page constantly, leading to scripts needing frequent updates. Additionally, Google has added in counter-measures to limit bot usage that require two-step mobile authentication making scaling up the deployment difficult.

One important next step is to understand how we can measure ephemeral congestion events when there is mostly UDP packets. We also need to scale up the data processing to analyze data for all hosts during the periods we are running active measurements. We hypothesize that some attributes



(a) raw RTT values

(b) FPS



(c) Cumulative dropped frames

(d) Rebuffering

Fig. 6. Measurements for a single Raspberry Pi

at the access-point-level or room-level may be useful in identifying or predicting congestion at a lower sampling frequency.

7 CONCLUSION

To sum up, we show that almost 23% of hosts in our campus network experience at least one ephemeral event in an hour, with more than a hundred hosts suffering congestion at any given time. We also show that both the duration of these events and the gaps between them are heavy-tailed. Most of these events are short-lived and are therefore difficult to detect with most existing network telemetry systems. We also report on a spatial scale-invariance property of these measured events. The signal at the host level appears to be too noisy to poll at a reasonable rate with active measurements. We hypothesize that monitoring downstream RTT inflations at the access-point-level or building-level with normalized values may provide us with a smoother signal.

Our initial analysis motivates further investigations. Specifically, we should explore if we observe similar behavior in other campus networks or more complex last-mile networks. In particular, we argue that this characterization is critical for designing next-generation telemetry systems for last-mile networks. These properties motivate future research explorations into how to model real-world ephemeral congestion events and whether or not they can be predicted. It is also worth exploring if it is possible to identify if hosts that mainly send/receive UDP packets (e.g., video conferencing) are experiencing ephemeral congestion events by monitoring latency inflation at

coarser spatial granularities. To encourage and support further research in this area, we will open-source our curated last-mile RTT dataset as well as webRTC stats collected through our active measurement pipeline.

REFERENCES

[1] Apache PySpark. https://spark.apache.org/docs/latest/api/python/, 2022.

[2] Barefoot networks wedge 100bf-32x programmable data center switch. https://www.edge-core.com/productsInfo.php?id=335, 2022.

[3] Chen, X., Kim, H., Aman, J. M., Chang, W., Lee, M., and Rexford, J. Measuring tcp round-trip time in the data plane. In *Proceedings of the Workshop on Secure Programmable Network Infrastructure* (2020).

[4] Crovella, M. E., and Taqqu, M. S. A Tool For Estimating the Heavy Tail Index from Scaling Properties . https://www.cs.bu.edu/fac/crovella/aest.html/, 1999.

[5] Fontugne, R., Shah, A., and Cho, K. Persistent last-mile congestion: Not so uncommon. In *Proceedings of the ACM Internet Measurement Conference* (2020), pp. 420–427.

[6] Gettys, J. Bufferbloat: Dark buffers in the internet. *IEEE Internet Computing 15*, 3 (2011), 96–96.

[7] Google Meet. https://meet.google.com/, 2022.

[8] Youtube. http://www.youtube.com/, 2022.

[9] tcptrace. https://linux.die.net/man/1/tcptrace, 2022.

[10] Microsoft teams. https://www.microsoft.com/en-us/microsoft-teams, 2022.

[11] Narayanan, A., Zhang, X., Zhu, R., Hassan, A., Jin, S., Zhu, X., Zhang, X., Rybkin, D., Yang, Z., Mao, Z. M., Qian, F., and Zhang, Z.-L. A variegated look at 5g in the wild: Performance, power, and qoe implications. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (New York, NY, USA, 2021), SIGCOMM '21, Association for Computing Machinery, p. 610–625.

[12] Netflix. https://www.netflix.com/, 2022.

[13] Ozer, S. Fastest path to low latency services: How can cable operators deliver consistent latency by following an efficient and future-proof design path? In *2021 Technical Forum SCTA / NCTA / CableLabs* (2021).

[14] Raspberry pi foundation. https://www.raspberrypi.org/, 2022.

[15] Saltstack. https://github.com/saltstack/salt, 2022.

[16] Sevcik, P., Jones, A., and Wetzel, R. 2020 internet latency benchmark report. In *NetForecast* (2021).

[17] Twitch. https://www.twitch.tv/, 2022.

[18] Zoom. https://www.zoom.us/, 2022.