Lost in Encryption: Monitoring Audio and Video Flows without Payload in Video-Conferencing Applications

Julien Gamba
Cisco ThousandEyes
Spain

Andre Felipe Zanella*
Telefónica Innovación Digital
Spain

Ricardo Morla
Cisco ThousandEyes
Portugal

Kyle Schomp
Cisco ThousandEyes
USA

Álvaro Feal
Cisco ThousandEyes
USA

Arash Molavi Kakhki
Cisco ThousandEyes
USA

ABSTRACT

With the increasing popularity of remote work, ensuring a sufficient level of Quality of Experience (QoE) in video conferencing applications (VCA) has become critical to ensure that employees can work reliably from anywhere. As such, monitoring of VCA has received much attention and requires solving several problems. First, because these applications typically generate a variety of network flows, those used for transporting critical media must be isolated from the rest of the traffic. Second, this identification must be performed at run-time because the VCA often selects the server IP addresses dynamically. Third, standards and apps are moving towards more encryption, making it harder to identify media flows and extract app-layer metrics.

We present a method for efficient and near real-time identification and classification of media flows from VCA, for both native and WebRTC-based versions. Our method relies on insights drawn from traffic patterns to detect media flows accurately in seconds, without prior knowledge of the app's internals, relying only on IP/UDP layer metadata, without depending upon payload or even RTP headers. Then, we extract application-layer metrics used for QoE estimation of media flows by using only IP/UDP packet metadata, and demonstrate that our heuristic-based estimators perform well under network degradation for Microsoft Teams.

CCS CONCEPTS

• **Networks** → **Network measurement**; **Network management**;

KEYWORDS

Video Conferencing, Quality of Experience, Access Networks, Encrypted traffic

ACM Reference Format:
Julien Gamba, Andre Felipe Zanella, Ricardo Morla, Kyle Schomp, Álvaro Feal, and Arash Molavi Kakhki. 2025. Lost in Encryption: Monitoring Audio and Video Flows without Payload in Video-Conferencing Applications. In *1st Workshop on Next-Generation Network Observability (NGNO '25), September 8–11, 2025, Coimbra, Portugal.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3748496.3748987

1 INTRODUCTION

Remote participation has seen much growth in the last few years, especially in the work place, where it is common for many meetings to be held virtually so that some participants may join remotely. According to a report by OwlLabs, in 2023, 88% of meetings had at least one remote participant.[1] This increase in remote collaboration has put QoE video-conferencing as one of top priorities of enterprises, leading to significant effort in monitoring QoE and adjusting to a degradation in QoE. In particular, the research community has long been focusing on the detection, classification and QoE monitoring of media traffic generated by video conferencing applications [6, 9, 26, 30, 32].

Many of the most frequently used video conferencing tools (e.g., Microsoft Teams, Cisco Webex, or Zoom) rely upon the Real-time Transport Protocol (RTP) for delivering media to the participants in a call [28]. While initially insecure, the payload is now frequently encrypted using the Secure Real-time Transport Protocol (SRTP) [5]. However, the headers are still unencrypted, and most of prior work relies on application-layer information to measure and gain deeper

[*] Work done as a PhD intern at Cisco ThousandEyes.

[1] https://owllabs.com/state-of-hybrid-work/2023

NGNO '25, September 8–11, 2025, Coimbra, Portugal

ACM ISBN 979-8-4007-2087-1/25/09...$15.00
https://doi.org/10.1145/3748496.3748987

insights about the performance of VCA [4, 19], using fields such as payload type (to classify media type), sequence number (to estimate packet loss), timestamp (to estimate frame boundaries), among others.

Yet, recent contributions by the IETF sought to encrypt fields of the RTP packet left unprotected by SRTP, such as the header extensions and contributing sources fields [11]. In this paper, we present methods to preemptively restore this monitoring gap. We take the approach of assuming that application-layer information is no longer available in today's video conferencing applications, and develop solutions for media flow detection, classification, and QoE monitoring for today's most frequently used VCA. Our methods rely solely upon information available at the network and transport layers, such as packet sizes and timings, making our methods robust to future changes and next generation protocols, as the network and transport layers will always be available to ensure compatibility with existing network equipment. We make the following contributions:

- We develop a universal method for media flow detection that works on any video conferencing application, WebRTC-based apps included, and relies only on packet timing information and flow metadata (i.e., a 5-tuple). Our method works in near real-time and can be used for monitoring media flows as the call takes place, and does not require any pre-existing dataset to be trained on. We achieve an average precision of 85% and an average recall of 96% (§3).
- We expand on the state-of-the-art of video conference QoE estimation relying solely on packet size and timing information, allowing measuring QoE metrics passively with network degradation (§4.1).
- We propose new methods for screen share detection (§4.2) and estimate network degradation (§4.3).

2 OVERVIEW OF VIDEO CONFERENCING APPLICATIONS

Video conferencing apps allow for real-time communication between two or more users. With two users, most VCA try to establish a peer-to-peer connection [1, 10]. In this case, the app must be able to perform Network Address Translation (NAT [7, 8]) traversal to establish the connection between the peers, typically using the STUN [17] or TURN [23] protocols. This is necessary as most networks, either residential [15] or ISP networks [25], implement some flavor of NAT to cope with the shortage of public IPv4 addresses [24], leading to multiple machines behind the same public IP address. With more than two users it is necessary to use a media relay server [31]. The relay performs the task of distributing media to all participants. Using a media relay can potentially downgrade the app's performance, depending upon the location of the participants and the relays [14], hence why apps prefer P2P connections for two-participant calls.

The majority of video conferencing apps rely on RTP and the RTP Control Protocol (RTCP) to send media traffic [27, 28], although some applications may choose to use other protocols or to encapsulate RTP into a proprietary protocol [20]. RTP and RTCP provides a framework that allow applications to deliver media traffic in a real-time fashion. These two protocols provide features to app developers that are not limited to the transport of media traffic (e.g., loss detection and correction, payload type identification, or membership management). Apps will periodically send RTCP reports which are then used by other participants of the call to synchronize the different media flows together, or to provide feedback about the quality of the received media. Among others, it is possible to use RTCP reports to know how many packets were lost during the call or to compute the packet jitter along the network path. By default, the majority of VCA use UDP as their transport protocol, with TCP as a fallback if UDP is not available (e.g., policy blocked).

It is common for VCA to also offer web variants that run in a web browser. These rely on the WebRTC standard for video calls. WebRTC provides a set of standard APIs[2] that is implemented by the major web browsers. Using WebRTC, an app can establish a direct connection to another host and allow audio and video communication. Under the hood, WebRTC relies on RTP to transport the media traffic [2, 3, 21]. Although WebRTC allows for other protocols to be used instead [12], the apps we study in this paper all rely on RTP for media transport.

3 MEDIA FLOWS DETECTION AND CLASSIFICATION

In this section, we describe our method to detect media flows during calls and classify them by the type of traffic they transport. Previous work relied mostly on RTP headers to identify media flows [22, 26], however, there is nothing preventing applications from using other protocols. This is the case with Zoom which uses a custom, proprietary protocol that encapsulates RTP [16, 18]. We instead propose a method to detect media flows that relies only on the 5-tuple of a flow and packet timing information. Our intuition is straightforward: in real-time video-conferencing applications, video and audio must be encoded and sent as fast as possible, to minimize delay in communication. This translates into frequent packet transmissions to avoid buffering and delay, negatively impacting the real-time experience. Figure 1 provides experimental results that confirm this intuition: it shows the size of packets and the time in which these packets are transmitted

[2] https://www.w3.org/TR/webrtc/

for two different media flows captured during a video call using Microsoft Teams. The orange lines show the timing of certain user actions which help us manually pinpoint the type of media transported by each flow.

We take advantage of this insight to detect media flows in video-conferencing apps from the client side, automatically and at-scale. We monitor the traffic of a specific app and for each flow (identified by its 5-tuple) we (1) discretize its traffic into windows of S seconds; (2) count the number of packets either sent or received during each window. Each given flow with more than N packets either sent or received during W consecutive windows is classified as a media flow.

We explored the range of possible values for the parameters N, S, and W to minimize the false positive and false negative rates, especially in the case of degradation of the network conditions during a call. Choosing parameters that are too low lead to a significant increase of false positives, i.e., non-media flows flagged as media, such as control traffic or long-lived requests. After extensive testing we settled on windows of $S = 2$ seconds, $N = 10$ packets per window, and $W = 10$ consecutive windows. We find that increasing them more does not help reducing the false positive rate and automatically increases the detection time for no benefit.

Our method does not require training of any kind and only relies on the 5-tuple of a flow and packet timing information, which makes it easy to implement and maintain. Our method is suitable for deployment at scale, and works in near real-time: only 20 seconds are needed to identify accurately all VCA media flows, which makes it suitable for network monitoring.

In-lab evaluation: we validated our method in a controlled environment on popular video-conferencing applications: Microsoft Teams, Zoom (app only), Cisco Webex, and Google Meet (WebRTC only). For Teams and Webex we tested both the app and the WebRTC version. We chose these applications as they are the most used video conferencing applications according to various online resources.[3] Our setup consists of a laptop able to capture packets and a router on which we can artificially add loss, delay, and jitter to the traffic to simulate bad network conditions. For each app, we start a call between two computers. One of the computers is connected to our access point, and we record a PCAP from that machine on the pktap virtual interface, an Apple-specific interface type that allow us to also capture the PID of the process that sent or received each packet[4], making it possible to filter out all packets that belong to other apps.

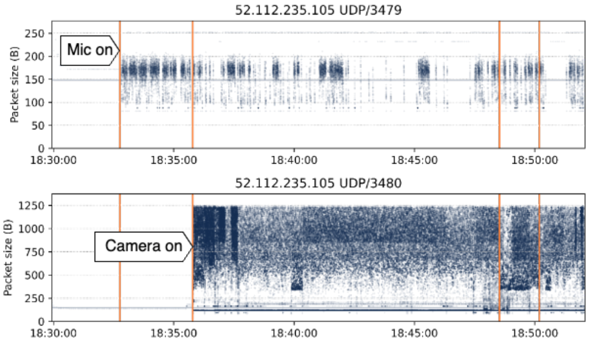


Figure 1: Media traffic of an MS Teams call (top: audio, bottom: video). Orange lines represent user actions.

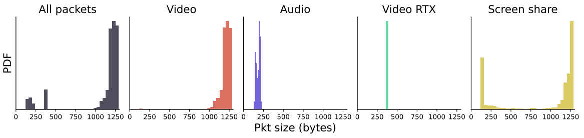


Figure 2: For a 2 participants call, size of received packets considering: (a) all packets, (b) video, (c) audio, (d) video re-transmission and (e) screen sharing packets.

We test our media flow detection method under normal conditions as well as with: (1) synthetic loss (20% of the packets), delay (500 ms), and jitter (50 ms); (2) UDP blocked, otherwise normal network conditions; (3) UDP blocked, with same synthetic loss, delay, and jitter as in (1). Comparison of the results of our method with the PCAP files recorded on the client confirms that we can accurately detect all media flows with no false positives, regardless of network conditions.

Validation at scale: we collaborated with a large software company to implement and deploy our method at scale on customer organizations. The company uses our method to detect media flows during Microsoft Teams calls from customers' laptops and monitor the media server used for each call. For validation, we rely on Microsoft Teams Call Quality Dashboard (CQD) which contains, among other things, the IP address of the media relays used during a call[5].

Our final validation dataset contains data from 124,278 calls from 27,200 Microsoft Teams users from 18 different customer organizations, taking place between the 24th of February, 2024 to the 6th of May, 2024. Our method achieves a true positive rate of 96%. The average precision per call is 85% but the average recall is 96%. We find that the average precision is lowered by IPs that we detect but are not reported by the CQD. Such IPs could be file transfers happening in chat messages during calls, or IPs reported with a masked last byte by the CQD. Regardless, this shows the effectiveness of our method and its suitability for real-world scenarios.

[3] https://www.statista.com/statistics/1331323/videoconferencing-market-share/, https://zapier.com/blog/best-video-conferencing-apps/
[4] https://www.tcpdump.org/linktypes/LINKTYPE_PKTAP.html
[5] https://learn.microsoft.com/en-us/microsoftteams/cqd-what-is-call-quality-dashboard

Classifying flows media types: recent work by Sharma et al. [29] showed that the packet size is a reliable indicator for the type of media. We confirm this result for the applications we consider, as illustrated in Figure 2 which shows the distribution of packet size for a single two-participants call that lasted three minutes under normal network conditions. The distributions are for all packets received by one of the users (we rely on RTP headers to obtain the ground truth of the media types). Audio and video were transmitted for the whole call, with screen share on for one minute. All audio packets are below 250 bytes, while video-related packets are above 750 bytes. Note that the packet size distribution can vary depending on the number of participants in the call. We conducted an experiment with six participants where, while the packet size in video flows remains the largest of all media types, it can sometimes drop below 750 bytes which was used as a lower bound by Sharma et al. We therefore use a lower bound of 250 bytes to identify video flows.

Figure 2 also reveals that screen sharing has a wider range of packet sizes including below the 250 bytes limit. This introduces an extra challenge when estimating application layer metrics of calls, as users sharing screen is an expected behavior. We will discuss our method for screen sharing detection in details in section 4.2.

4 PASSIVE ESTIMATION OF VIDEO METRICS WITHOUT RTP HEADERS

RTP headers and RTCP reports can provide application-layer metrics for inferring QoE but only if they are unencrypted. In case of encryption, we can passively estimate such metrics without access to RTP headers, relying on the 5-tuple of a flow and timing information. In this section we expand on previous work [29] and present our methods to detect frame boundaries and frame rate and to estimate video resolution (§4.1), to identify screen sharing (§4.2), and propose heuristics to detect network degradation (§4.3). We focus on Microsoft Teams, the most widely used VCA in professional settings.[6] However, our methods are not app-specific and should generalize to other VCA with minimal adjustments (e.g., parameters tuning). We tested our method using both UDP and TCP, observing similar results in both cases.

In-lab evaluation: We used two computers joining the same Microsoft Teams call using a media relay. On the first computer, we replaced the camera feed with the *FourPeople*[7] test sequence from the `Xiph.org` Video Test Media archive, a 1280 × 720 uncompressed video at 60 fps, looped using OBS Studio to control the resolution and framerate. The second computer joined the call with identical video settings. We captured the packets at both endpoints. All calls were conducted using Google Chrome and the WebRTC implementation of Microsoft Teams. We gathered ground-truth data using the built-in `chrome://webrtc-internals` tool which provides us with resolution, framerate, and packet loss for all incoming and outgoing media flows.

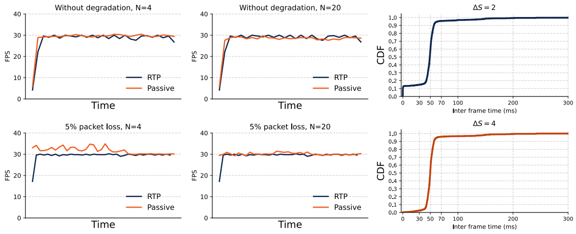


Figure 3: Parameter adjustment for frame rate estimation under network degradation. (a) left: increasing $N = 4$ to $N = 20$ improves detection with packet loss and remains efficient in normal conditions. (b) right: inter-frame time (IFT) distributions show how byte tolerance affects boundary detection. $\Delta S = 2$ gives 10% of frames with IFT=0ms, $\Delta S = 4$ yields accurate results.

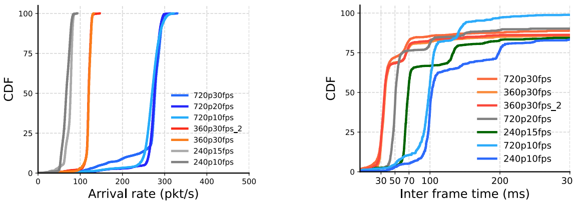


Figure 4: For 2 participants calls with varying resolution and frame rate: relationship between (left) packet arrival rate and resolution, (right) IFT and frame rate.

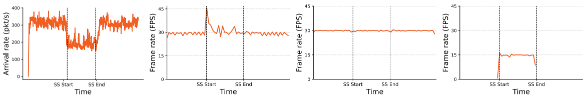
4.1 Measuring frame rate and resolution

Frame boundary detection: we rely on the passive frame boundary detection method proposed by Sharma et al. [29]. The authors observed that packets from the same frame have similar sizes while packets from different frames show significant variation, which make it possible to detect frame boundaries by comparing consecutive packets' sizes with tolerance for minimal size variation and packet reordering.

We further expand their work by validating their technique under network degradation and propose new parameters to make it more resilient. For each incoming packet P, we compare its size S_P against the previous $N = 20$ packets (resp. 1–3 packets in Sharma et al.) within a $\Delta S = 4$ bytes

[6] 93% of Fortune 100 companies report using Microsoft Teams — https://www.sci-tech-today.com/stats/microsoft-teams-statistics/

[7] https://media.xiph.org/video/derf/y4m/FourPeople_1280x720_60.y4m



(a) Arrival rate (b) FPS (all) (c) FPS (video) (d) FPS (SS)

Figure 5: For a 720p 2 participant call with screen sharing and time interval delimited by dashed lines: (a) the arrival rate of packets on the receiver’s side; (b) the passive estimation of frame rate of the call, showing a spike related to the new media track; The frame rate for (c) video and (d) screen sharing tracks, which when summed explains the spike on the passive estimation.

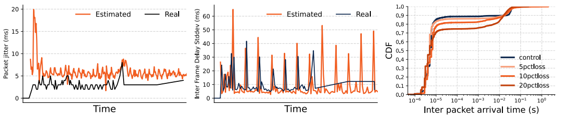
tolerance (resp. 2 bytes). If S_P falls within ±ΔS of the previous packets’ size range, it belongs to the current frame F_M; otherwise, it initiates a new frame F_{M+1}. This parameter combination addresses two issues: (1) packet loss (up to 10%) and out-of-order delivery corrupting boundary detection with lower tolerance (Figure 3a), and (2) incorrect frame splits from natural size variations, where $\Delta S = 2$ incorrectly identified 10% of frames (Figure 3b, indicated by frames with IFT=0 ms).

Frame rate estimation: we compute the frame rate by measuring the IFT after identifying frame boundaries and compare our results with the frame rate reported by Microsoft Teams. Microsoft Teams constrains frame rates to 1–30 FPS[8]. Our tests show our adjusted parameters maintain accurate estimation within this range even with 100ms jitter and delay, outperforming the baseline from previous work that overestimates FPS at 5% packet loss.

Video resolution: there is a direct correlation between packet arrival rates and video resolution as shown in Figure 4. The resolution can therefore be derived from the following threshold of the packet arrival rate (in packets per second): (1) over 250 pkt/s: resolution is 720p; (2) between 200 and 250 pkt/s: 540p; (3) between 110 and 150 pkt/s: 360p; and (4) below 110 pkt/s: 240p. Our method makes it possible to monitor the resolution continuously simply by measuring the packet arrival rate. Network degradation did not affect this estimator, as Microsoft Teams handles this by lowering the resolution, which our heuristic can keep track of.

4.2 Identifying screen sharing

During two-participant calls, we observe a drop in packet arrival rate when screen sharing begins (Figure 5a), along with a transient spike in our passive frame rate estimator from 30 to 45 FPS before stabilizing back at 30 FPS (Figure 5b). This stems from the combination of two distinct streams: the original video feed maintaining 30 FPS and reducing resolution (Figure 5c) and the screen share operating at 15 FPS (Figure 5d). We leverage this phenomenon to create a detection heuristic based on two sequential indicators: the initial packet arrival rate drop followed by an abnormal FPS spike. This approach enables reliable screen sharing detection relying solely on network-layer measurements.



(a) Packet jitter (b) Frame jitter (c) Packet loss

Figure 6: Real time estimations of (a) packet jitter and (b) frame jitter; (c) CDF of the inter packet arrival time for different levels of artificial packet loss.

4.3 Passive estimation of degrading network conditions

We propose passive estimators for three key network degradation metrics: packet jitter, frame jitter, and packet loss.

Packet jitter: calculated using mean deviation of packet arrival times (i.e., the time between the reception of two consecutive packets of a flow). While our results (Figure 6a) show slightly higher values compared to ground truth, our method effectively tracks network variations, which can provide insights about worsening conditions in real time.

Frame jitter: calculated using the standard deviation of inter-frame delay, which is the difference between the arrival time of the first packet of each frame. This relies on our frame boundary detection method. As shown in Figure 6b, our estimator matches the ground truth.

Packet loss: calculated using the inter-packet arrival time distributions. Our approach analyzes two distinct intervals in arrival times: microsecond scale intra-frame bursts and millisecond scale inter-frame gaps (evident in Figure 6c’s control curve). Increased loss reduces the proportion of sub-10μs intervals as retransmissions disrupt packet clustering. We map specific inter-packet time thresholds through quantile functions to empirical loss levels observed during testing.

5 LIMITATIONS AND FUTURE WORK

Multi-participant estimation: our methods were validated for two-participants calls. However, additional participants also increases the amount of packets received, which might skew our estimators. Figure 7a illustrates this problem: the receiver needs to know how many users are present to have

[8] https://support.microsoft.com/en-us/office/monitor-call-and-meeting-quality-in-microsoft-teams-7bb1747c-d91a-4fbb-84f6-ad3f48e73511

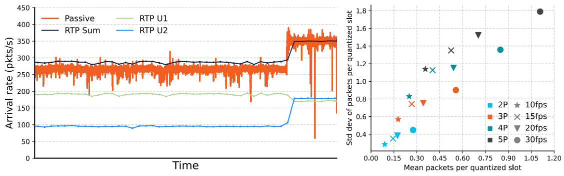


Figure 7: (a, left) packet arrival rate for a 3 participants call (b, right) estimation of the number of participants

an accurate estimation of the QoE metrics. Our initial analysis shows that we could leverage the variance in packet arrival rate to estimate the number of participants. We create a quantized time grid and a sampling frequency at least double the highest signal frequency present (in order to respect the Nyquist-Shannon sampling theorem). Here the signal is the video feed of a individual participant. Video feeds of MS Teams are limited to 30Hz so we chose a sampling frequency of 90Hz. We then count the number of packets that arrived at each slot of the grid across a time period and calculate the mean and standard deviation of the count of packets over the slots. Figure 7b shows a direct correlation between these statistics and the number of participants, which is expected: more participants means more packets sent from new video feed and more packets seen at each slot of the quantized grid.

Some combinations of number of participants and frame rates can be ambiguous if analyzing solely the mean packets (x-axis of Figure 7b). The standard deviation in the number of packets over the quantized grid solves this: a higher number of participants leads to jumps in the standard deviation. Figure 7b shows that this leads to a separation between mean and standard deviation across the possible combinations of number of participants and frame rate. By performing collections at scale, we believe it is possible to map these two values and create a heuristic to determine the number of participants of a VCA call passively. We plan on investigating this further in the future.

Test on other applications and validation at scale: so far, our method has been proven to work on multiple applications, both native or WebRTC-based. We believe our methods are generic enough to work on any VCA. We intend to test this assertion on the most popular applications that we have not already tested. Moreover, as of the time of writing, only the media flow detection has been tested at scale. We are currently in the process of expanding our collaboration to deploy and test the rest of our methods at scale for all apps to measure their accuracy in the wild.

6 RELATED WORK

Traffic classification has long been the subject of attention in the research community. either using protocol information [13, 20, 26] or machine learning [6, 9]. Perna et al. [22] present a method to detect and classify media flows in Real-Time Communication (RTC) applications using RTP headers and further rely on the RTP protocol to extract features and train a machine learning algorithm to classify media flows. The adoption of traffic encryption hinders such methods. In their paper, Crotti et al. [6] create a classifier to identify the protocol of a media flow using inter-arrival times, size of the IP packets, and the order in which they are seen by the classifier. Such features can be used with or without encryption, but bad network conditions would impact them, which may in turn lower the accuracy of this method. Erman et al. [9] use K-Means and DBSCAN to classify traffic using features such as the total number of packetsor mean packet inter-arrival time. However, this require knowing the number of packets which prevents this method from being used live.

Sharma et al. [29] relies on heuristics to detect and classify media flows into video or audio. Their approach relies on automatic classification of packets transporting video by relying on their size: packets larger than a size threshold is marked as video. However, the authors do not consider the problem of identifying media flows or additional features such as screen sharing or calls with more than two participants.

7 CONCLUSION

In this paper, we present methods to detect, classify, and measure the QoE of media flows in VCA. Our methods only rely on IP level metadata and can be used with native or WebRTC-based apps. We demonstrated the efficiency of our methods both in lab with various network conditions, but also at scale by collaborating with a large software company. This work represent a significant improvement over previous methods which either relied on app- or protocol-level information. Future studies could build on our work and explore new passive estimators for QoE.

REFERENCES

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

[2] H. Alvestrand. 2021. Overview: Real-Time Protocols for Browser-Based Applications (RFC 8825).

[3] H. Alvestrand. 2021. Transports for WebRTC (RFC 8835).

[4] G. Carofiglio, G. Grassi, E. Loparco, L. Muscariello, M. Papalini, and J. Samain. 2021. Characterizing the relationship between application QoE and network QoS for real-time services. In *ACM SIGCOMM workshop on network-application integration*.

[5] E. Carrara, K. Norrman, D. McGrew, M. Naslund, and M. Baugher. 2004. The Secure Real-time Transport Protocol (SRTP) (RFC 3711).

[6] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli. 2007. Traffic classification through simple statistical fingerprinting. *SIGCOMM Computer Communication Review*.

[7] K. Egevang and P. Francis. 1994. The IP Network Address Translator (NAT) (RFC 1631).

[8] K. Egevang and P. Srisuresh. 2001. Traditional IP Network Address Translator (Traditional NAT) (RFC 3022).

[9] J. Erman, M. Arlitt, and A. Mahanti. 2006. Traffic classification using clustering algorithms. In *SIGCOMM workshop on Mining network data*.

[10] S. Guha and N. Daswani. 2005. *An experimental study of the skype peer-to-peer voip system*. Technical Report. Cornell University.

[11] C. Jennings J. Uberti and S. Garcia Murillo. 2023. Completely Encrypting RTP Header Extensions and Contributing Sources (RFC 9335).

[12] R. Jesup, S. Loreto, and M. Tüxen. 2021. WebRTC Data Channels (RFC 8831).

[13] M. Lyu, S. Madanapalli Chandra, A. Vishwanath, and V. Sivaraman. 2024. Network Anatomy and Real-Time Measurement of Nvidia GeForce NOW Cloud Gaming. In *International Conference on Passive and Active Network Measurement*.

[14] K. MacMillan, T. Mangla, J. Saxon, and N. Feamster. 2021. Measuring the performance and network utilization of popular video conferencing applications. In *Internet Measurement Conference*.

[15] G. Maier, F. Schneider, and A. Feldmann. 2011. NAT usage in residential broadband networks. In *International Conference on Passive and Active Network Measurement*.

[16] B. Marczak and J. Scott-Railton. 2020. *Move Fast and Roll Your Own Crypto: A Quick Look at the Confidentiality*. Technical Report. The Citizen Lab.

[17] P. Matthews, J. Rosenberg, D. Wing, and R. Mahy. 2008. Session Traversal Utilities for NAT (STUN) (RFC 5389).

[18] O. Michel, S. Sengupta, H. Kim, R. Netravali, and J. Rexford. 2022. Enabling passive measurement of zoom performance in production networks. In *ACM Internet Measurement Conference*.

[19] A. Nikravesh, D. Hong Ke, Q. Chen Alfred, H. Madhyastha, and Z. Mao Morley. 2016. QoE inference without application control. In *ACM SIGCOMM Workshop on QoE-based Analysis and Management of Data Communication Networks*.

[20] A. Nisticó, D. Markudova, M. Trevisan, M. Meo, and G. Carofiglio. 2020. A comparative study of RTC applications. In *IEEE International Symposium on Multimedia*.

[21] C. Perkins, M. Westerlund, and J. Ott. 2021. Media Transport and Use of RTP in WebRTC (RFC 8834).

[22] G. Perna, D. Markudova, M. Trevisan, P. Garza, M. Meo, M. Munafò, and G. Carofiglio. 2022. Real-time classification of real-time communications. *IEEE Transactions on Network and Service Management*.

[23] T. Reddy, A. Johnston, P. Matthews, and J. Rosenberg. 2020. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN) (RFC 8656).

[24] P. Richter, M. Allman, R. Bush, and V. Paxson. 2015. A primer on IPv4 scarcity. *ACM SIGCOMM Computer Communication Review*.

[25] P. Richter, F. Wohlfart, N. Vallina-Rodriguez, M. Allman, R. Bush, A. Feldmann, C. Kreibich, N. Weaver, and V. Paxson. 2016. A multi-perspective analysis of carrier-grade NAT deployment. In *ACM Internet Measurement Conference*.

[26] A. Buyukkayhan S., A. Kavak, and E. Yaprak. 2013. Differentiating voice and data traffic using statistical properties. In *International Conference on Electronics, Computer and Computation*.

[27] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. 1996. RTP: A Transport Protocol for Real-Time Applications (RFC 1889).

[28] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. 2003. RTP: A Transport Protocol for Real-Time Applications (RFC 3550).

[29] T. Sharma, T. Mangla, A. Gupta, J. Jiang, and N. Feamster. 2023. Estimating WebRTC Video QoE Metrics Without Using Application Headers. In *Proceedings of the 2023 ACM on Internet Measurement Conference*.

[30] A. Moore W and K. Papagiannaki. 2005. Toward the accurate identification of network applications. In *International workshop on passive and active network measurement*.

[31] C. Yu, Y. Xu, B. Liu, and Y. Liu. 2014. "Can you SEE me now?" A measurement study of mobile video calls. In *IEEE INFOCOM*.

[32] M. Zhang, W. John, K. Claffy, and N. Brownlee. 2009. State of the art in traffic classification: A research review. In *PAM Student Workshop*.