Silhouette – Identifying YouTube Video Flows from Encrypted Traffic

Feng Li
Verizon Labs
Waltham, MA, 02145
feng.li@verizon.com

Jae Won Chung
Verizon Labs
Waltham, MA, 02145
jae.won.chung@verizon.com

Mark Claypool
Worcester Polytechnic Institute
Worcester, MA, 01609
claypool@cs.wpi.edu

ABSTRACT

Video streaming traffic often dominates mobile wireless networks, forcing Internet Service Providers (ISPs) to deploy video shaping to identify and then manage traffic during congested periods. Unfortunately, the increasing use of end-to-end encryption (e.g., TSL/SSL) makes it difficult to identify video flows even with deep packet inspection. As an alternative, this paper presents *Silhouette* – a real-time, lightweight video classification method suitable for ISP middle-boxes. Silhouette uses only flow statistics (i.e., "shape") for video identification making it payload-agnostic, effective for identifying video flow even when encrypted. Preliminary results with pre-classified YouTube traffic shows the promise of the Silhouette approach, yielding high identification accuracy over a range of video content and encoding qualities.

CCS CONCEPTS

• **Information systems → Multimedia streaming**; • **Networks** → *Packet classification*;

KEYWORDS

YouTube, Service Classification, HTTP Adaptive Streaming, QUIC

ACM Reference Format:
Feng Li, Jae Won Chung, and Mark Claypool. 2018. Silhouette – Identifying YouTube Video Flows from Encrypted Traffic. In *NOSSDAV'18: 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video, June 12–15, 2018, Amsterdam, Netherlands.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3210445.3210448

1 INTRODUCTION

Video traffic is the *king of the hill* [10], accounting for around 50% of traffic volume over a U.S. tier-1 mobile Internet service provider (ISP), and is projected to increase 11x by 2020, accounting for 75% of all mobile data traffic [3].

As a response to the deluge of video, mobile Internet service providers deploy traffic shaping or policing solutions to mitigate the impact of video traffic. For example, T-Mobile's BingeOn plan throttled video bitrates to 1.5 Mb/s for unlimited plan customers [6], and other U.S. mobile ISPs soon followed suit, limiting video rates to lower radio link utilizations. However, given the self-imposed rate limits of video by encoding and the latency resilience of video from buffering, there is the potential to smooth video rather than just limit its rate, easing congestion without significantly impacting video quality [11].

But if such video shaping potential is to be realized, ISPs need real-time, accurate identification of video traffic. Blindly pacing or throttling flows that may or may not be video can impair application performance in general. Currently, in an attempt to identify video flows, ISPs often deploy deep packet inspection (DPI) engines in network middle-boxes. These DPI engines typically rely on packet content, such as host headers and Transport Layer Security (TLS), but may also use Server Name Indications (SNIs) in order to selectively apply shaping rules to flows [12]. However, DPI-based video detection has several drawbacks:

Easy to spoof. Kakhki *et al.* demonstrated a simple spoofing method for BingeOn with an HTTP proxy only several months after the service became public available [6]. Li *et al.* [12] developed a library that exposed traffic classification policies used inside mobile ISP's middle-boxes, making it even easier to circumvent traffic classification rules.

Ineffective with encryption. End-to-end encryption, such as TLS/SSL, renders content-based DPI engines ineffective. Unfortunately, over mobile networks today, nearly 70% of traffic is protected by TLS or SSL [10]. Thus, unless video end points are unencrypted or are terminated at ISP proxies, payload-based identification alone will fail.

Ineffective when source unknown. Many DPI engines build a traffic signature from training data sets where video sources are known *a priori*. If the pre-trained DPI engine is presented with video from a new source, it is unable to properly classify the traffic as video.

This paper presents *Silhouette* ,[1] a video classification method that uses only the transport layer flow characteristics (the "outline" or "shape" of the flow), not needing to inspect transport layer sequence numbers or application layer payloads. Silhouette delimits flows with the classic 5-tuple (source & destination IP addresses, source & destination ports, and protocol type), but then records only average transport layer payload size and data rate to decide whether a given flow is video or not. The Silhouette algorithms are generic enough to work with HTTPS/TCP and QUIC/UDP-based streaming and does not require prior collection and training, nor run-time machine learning.

[1] A *silhouette* is a shape/outline of something visible against the background.

NOSSDAV'18, June 12–15, 2018, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5772-2/18/06…$15.00
https://doi.org/10.1145/3210445.3210448

Evaluation of Silhouette with 660 video traces – 66 video clips with five different resolutions (240p, 360p, 480p, 720p and 1080p) and two different protocols (TCP and QUIC) – yields an accuracy of 88%, with most of the unidentified videos those with low encoding rates.

The rest of the paper is organized as follows: Section 2 summarizes related work; Section 3 provides background on YouTube streaming; Section 4 presents Silhouette and an illustrative example; Section 5 evaluates Silhouette with a broad range of YouTube videos; and Section 6 summarizes our conclusions and presents possible future work.

2 RELATED WORK

Identifying video flows amid encrypted traffic is not a new research area, spawning at least dozens of machine learning (ML)-based proposed solutions over the last two decades [8, 14, 15, 18]. Unfortunately, while promising, ML techniques require *a priori* training and are unusable for new, as yet unclassified applications and can be processing-intensive if run in real-time. Deep packet inspection (DPI)-based approaches can achieve high accuracy [13], but the high computational complexity and ineffectiveness for encrypted payloads make the DPI-based approach impractical for real-time classification.

Other research has focused on measuring video QoE for encrypted traffic by passively monitoring the network and application statistics [7]. For example, Dimopoulos et al. [5] infer stalling, average video quality and quality variations by observing network statistics. However, their method requires a data "cleaning" phase to filter out traffic from domain names not related to the services. Thus, although they infer the QoE of previously-identified encrypted video sessions, they do not address the problem of identifying video flows from encrypted traffic.

Reed et al. [16] identify specific Netflix videos using only the information provided by TCP/IP headers. However, their approach requires a "finger print" database built from a large number of video clips, making it ineffective for systems such as YouTube where around 400 hours worth of new video are uploaded per minute.[2]. Also, since their approach uses TCP sequence numbers, it is ineffective on video protocols with negative acknowledgments, such as QUIC [1]. Lastly, ISPs typically do not need to know the exact video content for treatment, but rather just need to correctly identify a flow as video.

Other work has proposed treatment-based traffic classification [4, 11], where video traffic, once identified, is paced without significantly degrading user QoE. However, their video detection is designed for the real-time streaming protocol (RTSP) and may not be accurate for modern video traffic which predominantly streams over HTTP.

3 YOUTUBE VIDEO STREAMING

For the past decade or so, most content providers have used HTTP for video streaming. HTTP streaming combines the advantages of firewall penetration and easy network address translation [5, 9] with TCP's reliable packet delivery and congestion control. Even QUIC over UDP is similar for video streaming with TCP + TLS + HTTP/2 [1]. YouTube HTTP video streaming has two Transport Layer implementations: QUIC-enabled HTTP streaming and TCP-enabled HTTP streaming. While Bhat et al. [2] find QUIC does not necessarily improve performance over TCP-based streaming, from the application layer perspective, QUIC-based YouTube streaming should behave similarly to TCP-based YouTube Streaming. Although layers under the application may impact some flow statistics (e.g., retransmission behaviors for QUIC and TCP), given the application-layer commonality, the transport layer statistics should appear similar. This motivates our search for a general classification method to detect HTTP video traffic for both HTTPS/TCP and QUIC/UDP.

YouTube video streaming supports two different approaches [5, 9]: *progressive downloading* for low quality videos (240p, 360p, and 480p) and *HTTP adaptive streaming (HAS)* for high definition (HD) videos (720p and 1080p).

For streaming with progressive downloading, each video session consists of two phases: startup and steady state. In startup, the video session downloads rapidly to fill the client player's playout buffer as fast as possible. Once the buffer is filled, the video session enters steady state and pauses downloading, resuming only when the playout buffer depletes.

For HAS, videos are split on the server into multiple segments, each corresponding to 2 to 10 seconds of playback time. Each video segment is encoded into a range of different qualities [9]. The client determines which segment to download as a function of the throughput observed while downloading the previous segments and the available seconds of playback in its playout buffer [5].

In both progressive downloading and HAS, the video player sends an HTTP request to the server to request the next video segment. In HAS, the next segment may have a different quality encoding than the previous segment, while in progressive downloading, all segments have the same quality. Therefore, for HTTP YouTube streaming, we expect to observe multiple HTTP requests in the uplink direction.

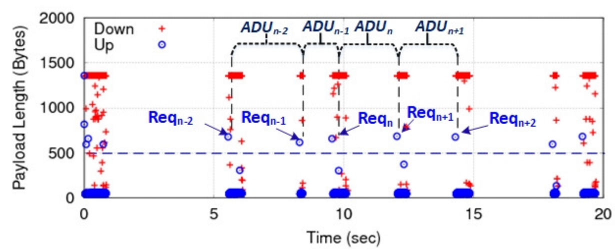


Figure 1: YouTube Streaming with QUIC (*Alpha* Trailer)

Figure 1 shows the first 20 seconds of network behavior for a QUIC/UDP, 1080p, 24 f/s YouTube video,[3] the official movie trailer for *Alpha 2018*, using a Chrome browser[4] on Linux. The x-axis is the elapsed packet capture time and the y-axis is the transport layer payload size – excluding UDP/IP headers. The red plus (+) symbols are downlink packets and the blue circles are uplink packets.

The session is bursty and has large downlink packets but mostly small uplink packets. However, some uplink packets are larger than 500 bytes, and these precede a series of large downlink packets.

[2]https://www.youtube.com/yt/about/press/

[3]HTTPS/TCP streaming appears similar, but is not shown due to space constraints.

[4]Chrome version 63.0.3239.84.

Based on YouTube streaming behavior described above, the large uplink requests are likely video retrieval requests for the next video segments – Silhouette leverages this in the next section.

4 SILHOUETTE

Inspired by our observations of YouTube video streaming in Section 3, we propose Silhouette , a real-time heuristic method to detect video streaming flows based on Application Data Units (ADUs) and network statistics. This section describes Silhouette , which consists of two algorithms (Section 4.1), and shows an example with a YouTube streaming video (Section 4.2). A Python implementation of Silhouette is available online.[5]

4.1 Video Flow Identification

A key aspect of Silhouette is identifying Application Data Units (ADUs) (i.e., a video segment), depicted in Figure 1. When an observer (e.g., a traffic control middle-box) sees a packet sent from the client (e.g., a video player) carrying a payload larger than a fixed threshold L_{adu}, the observer makes a note of the time as the start of a new ADU. Subsequently, the observer sums up the sizes of the packet payloads in the opposite direction until the next large request in the uplink direction. The sum of the payload sizes between the two requests is the size of the ADU (*adu*). The transport layer packets without any payload (e.g., pure TCP ACKs) do not carry any application layer information [11, 17] and are not used to decide ADU boundaries or video segment sizes.

4.1.1 Feature Selection.

Silhouette uses video segment size and inter-request time as major video classification features. Unlike other ADU approaches [16, 17], Silhouette ADU size calculation does not require segment sequence number analysis – this is advantageous in the presence of QUIC which supports multiplexing and uses Negative Acknowledgments (NACK).

Silhouette also uses other network layer statistics common to both HTTP/TCP and QUIC/UDP streaming to improve video identification accuracy. These metrics include:

average downlink payload size ($\bar{L}$): total downlink payload size divided by the number of non-empty downlink packets. Average payload size has been effective for traffic differentiation in previous work [4, 11].

data rate (R): cumulative transport layer payload size divided by flow elapsed time. Data rate is used to differentiate video from audio, which has a considerably lower data rate (less than 192 kb/s).

4.1.2 Video Identification with ADU.

The Silhouette algorithm consists of two parts: detecting an ADU (Algorithm 1) and classifying a video (Algorithm 2). Table 1 summarizes the default threshold values used by Silhouette .

In HTTPS/TCP streaming, the flow uplink data consists of requests for video segments and TCP ACKs. In QUIC/UDP, the flow uplink data is more varied since QUIC sends statistics data with its NACK/ACK packets in the uplink. Silhouette uses L_{req} as a minimum threshold to filter out small packets which are unlikely to carry video segment requests (Line 8 in Algorithm 1). The L_{req} default threshold is 500 bytes, tuned from observing hundreds of YouTube video sessions for both HTTPS/TCP and QUIC/UDP.

ADUs alone are often not sufficient to differentiate video flows from audio flows, especially for QUIC/UDP where video and audio can be multiplexed in a single connection. Audio segments can even be requested before the complete transfer of a video segment [9]. Thus, Silhouette uses T_v as a threshold for video ADU size to differentiate smaller audio. The T_v default threshold is 100 KB, approximately 2 seconds of 240p video encoded at 400 kb/s.[6]

As discussed in Section 4.1.1, Silhouette uses average downlink payload length ($\bar{L}$) and data rate (R) to differentiate audio and video traffic. Silhouette uses $L_v = 900$ bytes as the default payload length threshold for video ($\frac{2}{3}$ a typical MTU without network and transport headers), and $L_a = 450$ bytes as the default payload length threshold for audio ($\frac{1}{3}$ a typical MTU without network and transport headers). Silhouette uses $R_v = 300$ kb/s as the default data rate threshold for video, and $R_a = 192$ kb/s as the default data rate threshold for audio.

Algorithm 1 Application Data Unit Detection

```
1: variables
2:     adu: application data unit length.
3:     l: transport layer payload length of pkt p.
4:     C_adu: number of ADUs detected.
5: end variables
6: for each uplink packet p in flow id do
7:     l ← payload_len(p)
8:     if l < L_req then                    ▷ Too small to be a request
9:         return
10:    end if
11:    if adu ≥ T_v then
12:        C_adu ← C_adu + 1
13:    end if
14:    adu ← 0
15: end for
```

Table 1: Thresholds used in Algorithms 1 and 2

Thresh.	Description	Defaults
T_v	segment length threshold for video ADU	100 KB
R_v	rate threshold for video	300 Kb/s
R_a	rate threshold for non-video	192 Kb/s
L_v	pkt length threshold for video	900 B
L_a	pkt length threshold for non-video	450 B
L_{req}	pkt length threshold for request	500 B
T_{adu}	number of ADUs threshold for video	3

Algorithm 2 consists of two parts: updating the statistics information for the flow and the current ADU (Lines 10 to 18), and identifying if the flow is a video based on the statistics. Line 19 in Algorithm 2 does the video identification. Only if the flow meets *all* three conditions is it classified as video: 1) data rate (R) is greater than the minimum video rate threshold (R_v), 2) average payload length ($\bar{L}$) is greater than the video payload length threshold (L_v), and 3) at least 3 video ADUs (T_{adu}). Note, this last criteria is because HTTP streaming always has at least three uplink requests:

[5] http://perform.wpi.edu/downloads/#silhouette

[6] YouTube recommends encoding 240p video at 400 kb/s.

Algorithm 2 Video Flow Identification

1: **variables**
2: c: number of pkts with transport layer payload.
3: l: transport layer payload length of pkt p.
4: t_e: elapsed time from 1st pkt.
5: $\bar{L}$: average length of transport layer payload.
6: S: cumulative length of transport layer payload.
7: R: application layer throughput.
8: **end variables**
9: **for** each downlink packet p in flow id **do**
10: $l \leftarrow payload\_len(p)$
11: **if** $l = 0$ **then** ▷ no application level data
12: **return** $current\_type(id)$
13: **end if**
14: $adu \leftarrow adu + l$
15: $c \leftarrow c + 1$
16: $S \leftarrow S + l$
17: $\bar{L} \leftarrow S/c$ ▷ avg payload length
18: $R \leftarrow S/t_e$
19: **if** $(R \geq R_v) \cap (\bar{L} \geq L_v) \cap (C_{adu} \geq T_{adu})$ **then**
20: **return** $true$ ▷ likely video flow
21: **else if** $(S < T_v) \cup (R < R_a) \cup (\bar{L} < L_a)$ **then**
22: **return** $false$ ▷ unlikely video flow
23: **else**
24: **return** $maybe$ ▷ maybe video flow
25: **end if**
26: **end for**

one to retrieve the manifest file, one to retrieve the audio segment, and one to fetch video segments.

Line 19 in Algorithm 2 identifies a non-video flow. A flow is marked as non-video if *any* of the following three conditions is met: 1) the cumulative payload length (S) is less than one video ADU (L_{adu}); 2) the data rate (R) is less than the audio rate threshold (R_a); and 3) the average payload length ($\bar{L}$) is less than the audio payload length threshold (L_a).

If a flow fails to be identified as either video or non-video, it is marked as *maybe*, since it may be video, but it may not be. Usually, some file transfers fall into the maybe category (e.g., downloading a modest-sided picture, such as a thumbnail) since they have only one, large ADU.

4.2 Video Classification Example

Table 3 shows an example of Silhouette in action, using a YouTube QUIC/UDP streaming session of the movie trailer (*Alpha*). The video is 159 second long, encoded at 1080p with 24 f/s. Before the trailer begins, YouTube streams a 30 second advertisement. In order to validate the classification results, we manually enabled the "stats for nerds" option in the player.

The video format id is 248 (`webm video` at 1080p), and the audio format id is 250 (`opus` at 70 kb/s). For validation, we retrieved these two clips using the YouTube Downloader[7] tool, and confirm that the volume of Flow#20 and Flow#10 match with the downloaded media file size.

From Table 2, even the single video streaming session opens 24 connections. From the server name indications (SNIs), most of the connections are not related to streaming: some are used for account management, some are used for the advertisement, and some are used to collect statistics. The flows with SNIs of *r*—sn-*.googlevideo.com* might carry media content, with the exception of Flow #21 because its duration and volume are too small. The advertisement was provided from the server *r1—sn-9xp1vo-cvns*, and the Alpha trailer was provided from the server *r6—sn-8xgp1vo-cvne*. Silhouette correctly identifies Flow #13 and Flow #20 as video without checking their SNIs. Silhouette marks Flow #10 as *maybe* because it does not have a valid video ADU and its volume and duration are too small.

5 EVALUATION

Building upon the single-session classification example from Section 4.2, we conduct a large scale evaluation of Silhouette by: 1) gathering a set of traces with a known number of video flows (Section 5.1), 2) analyzing the (in-)effectiveness of SNI-based classification (Section 5.2), and 3) evaluating the effectiveness of Silhouette (Section 5.3).

5.1 Ground Truth Collection

In order to assess Silhouette more broadly, we gathered traces from 66 YouTube videos, repeatedly streamed over a range of encodings and protocols. This provides the "ground truth" – real traces known *a priori* to be video. We have made these traces available online for other researchers.[8]

Before streaming, we setup a testbed to collect packet level traces from a Linux desktop with a dual core Intel i5-4460 CPU, 32 GB RAM, running the 4.11.0-rc2 kernel. The computer connected to Internet with a broadband connection (150 Mb/s) through an IEEE 802.11 a/b/g/n router at a carrier frequency of 2.4 GHz. The desktop computer was close enough to the wireless router to achieve a maximum TCP downlink throughput of 50 Mb/s, which is more than fast enough to stream 1080p video, the highest encoding rate tested. The computer uses Google Chrome (v63.0.3239.84) for streaming. A custom Python script automatically plays each video through the Chrome browser, using `tcpdump` to collect full packet traces.

To cover a range of typical video flows, we stream all 66 videos from the "2018 Movie Trailers" YouTube play list,[9] with five different encoding qualities (240p, 360p, 480p, 720p, and 1080p) and two protocols (HTTPS/TCP and QUIC/UDP) – in total 660 video sessions. We manually enable the "stats for nerds" option in the YouTube player for each session to record the video server name and the format id for validation purposes. In order to avoid possible browser caching, we clear all history data after every session and restart the browser.

5.2 Server Name Indication

The server name indication (SNI) can be used to infer the possible role/content of each flow, a method used by major U.S. mobile ISPs [12].

Although our scripts initiated only 660 streaming sessions, they spawned over 15,000 flows. YouTube uses the additional flows for other tasks such as managing accounts (e.g., *{youtube|google}.com*), third party advertising (e.g., *{adservice|doubleclick}.com*), collecting

[7]https://rg3.github.io/youtube-dl

[8]http://perform.wpi.edu/downloads/#silhouette

[9]Play list id: `PLh2QSchbA3pnM2pTPI6mkj4kyhJfp_v1h`, January 2018

Table 2: Sample Results of Silhouette

flow	start (sec.)	duration (sec.)	vol. (KB)	rate (kb/s)	avg. pkt len.(B)	#ADUs	SNI	video
0	0.00	228.07	414	14	820	2	www.youtube.com	false
1	0.00	0.17	4	187	830	0	fonts.googleapis.com	false
2	0.00	42.15	7	1	336	0	googleads.g.doubleclick.net	false
3	0.00	0.17	4	195	830	0	fonts.gstatic.com	false
4	0.01	3.14	61	155	1117	0	i.ytimg.com	false
5	0.04	28.23	15	4	1256	0	www.gstatic.com	false
6	0.04	1.29	16	104	1012	0	s.ytimg.com	false
7	0.04	42.05	5	0	313	0	www.google.com	false
8	0.06	41.83	71	13	1192	0	yt3.ggpht.com	false
9	0.07	0.18	4	178	830	0	clients1.google.com	false
10	0.69	1.06	926	6990	1341	0	r6—sn-8xgp1vo-cvne.googlevideo.com	*maybe*
11	1.14	0.33	6	152	706	0	securepubads.g.doubleclick.net	false
12	1.53	0.29	5	140	647	0	www.googleapis.com	false
13	1.62	7.97	8251	8282	1349	10	r1—sn-8xgp1vo-cvns.googlevideo.com	**true**
14	2.10	30.39	10	2	542	0	pagead2.googlesyndication.com	false
15	2.34	15.15	4	2	589	0	ad.doubleclick.net	false
16	2.34	47.25	5	1	263	0	ade.googlesyndication.com	false
17	2.40	0.57	32	460	1251	0	tpc.googlesyndication.com	false
18	32.16	9.73	6	5	534	0	www.googleadservices.com	false
19	34.56	35.57	69	15	1229	0	i1.ytimg.com	false
20	41.58	88.90	32137	2892	1351	24	r6—sn-8xgp1vo-cvne.googlevideo.com	**true**
21	41.64	0.19	9	421	834	0	r2—sn-8xgp1vo-xfgy.googlevideo.com	false
22	118.43	0.28	6	182	732	0	clients2.google.com	false
23	200.36	17.18	180	83	1215	0	i.ytimg.com	false

statistics (e.g., *{googleapis|gstatic.com}*), and hosting thumbnail images (e.g., {ytimg|ggpht}.com).

Our previous measurement study showed that only flows with SNIs like *r*—sn*.googlevideo.com* carry video content [10]. Figure 2 shows a cumulative distribution function (CDF) for flow level statistics grouped into two categories: flows containing SNIs matching *r*—sn*.googlevideo.com* are marked as "googlevideo", and the rest are marked as "others". The left graph is the distribution of flow volumes and the right graph is the distribution of flow durations. Note that for both graphs, the x-axis is in logscale.

Based on the flow volume (Figure 2(a)), not all flows with an SNI of *r*—sn*.googlevide.com* carry video content – flows with a payload less than 1 MB are unlikely to be video. While flow volume can be accurate for identifying video, unfortunately by itself flow volume is a poor feature for real-time classification since it can only be definitive at the end of the flow. For similar reasons, flow duration (Figure 2(b)) is also a poor feature for real-time classification. Moreover, some flows with SNIs with *youtube.com* only terminate when the Chrome browser is closed – thus, there is not a strong correlation between video length and the duration of the flows. This shows that SNI-based approaches alone are not able to differentiate video from other types traffic if the same server has multiple roles (e.g., providing FTP and video streaming services on the same host).

In summary, SNIs alone are ineffective for differentiating video/non-video flows.

5.3 Silhouette

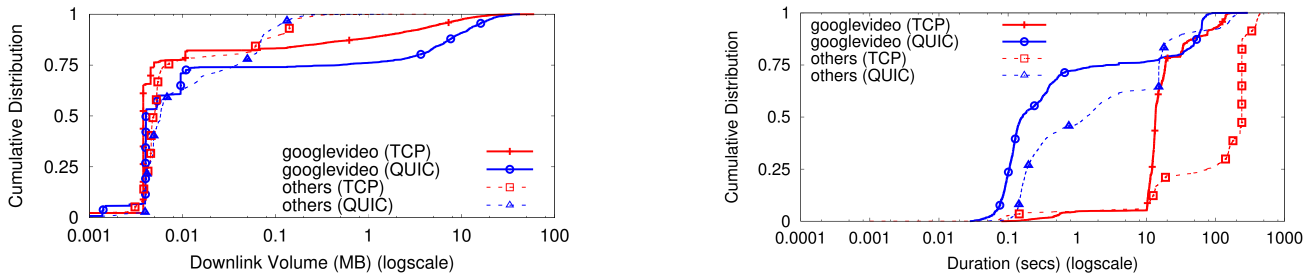
Table 3 summarizes the Silhouette classification results for our ground truth dataset. The videos are broken down by streaming protocol ("HTTPS/TCP" and "QUIC/UDP", the main columns) and then by video encoding (the rows). For each protocol, the classification results for the 66 video flows are tabulated: video (video correctly classified as video), maybe (video not classified as video, but not mis-classified as non-video), and non-video (video mis-classified as non-video). The "%" column depicts the percentage of video flows correctly classified as video. The final row tabulates the results for all video encoding types.

Figure 3 depicts the accuracy of Silhouette for the ground-truth dataset, with the x-axis the video encoding and the y-axis the accuracy percent. The overall accuracy for Silhouette is 87.9%, and for video qualities over 480p, accuracy reaches 95%. Silhouette accuracy is lowest for HTTPS/TCP 240p video (24%) where 39 out of the 66 videos are marked as *maybe*. Looking more closely, for 240p and 360p videos, HTTPS/TCP-based streaming experiences long idle durations that bring down average flow rates below the video rate threshold (R_v = 300 kb/s).

Although there are multiple connections in one single YouTube session (as the example in Table 2 shows), Silhouette does not classify any flows outside of SNI *r*—sn-*.googlevideo.com* as video — i.e., Silhouette has a zero false positive rate.

6 CONCLUSIONS

Accurate, real-time classification of encrypted video is becoming increasingly important given the dominance of HTTP-streaming video and the growth in end-to-end encryption.



(a) Distribution of Flow Volume

(b) Distribution of Flow Duration

Figure 2: Flow Statistics in "Ground Truth" Data Set

Table 3: Video Detection Results

Enc. Type	HTTPS/TCP vid.	maybe	non-vid.	%	QUIC/UDP vid.	maybe	non-vid.	%
240p	16	39	11	24	63	3	0	95
360p	49	14	3	74	65	1	0	98
480p	63	3	0	95	66	0	0	100
720p	63	3	0	95	66	0	0	100
1080p	65	0	1	98	66	0	0	100
All	256	59	15	77	326	4	0	99

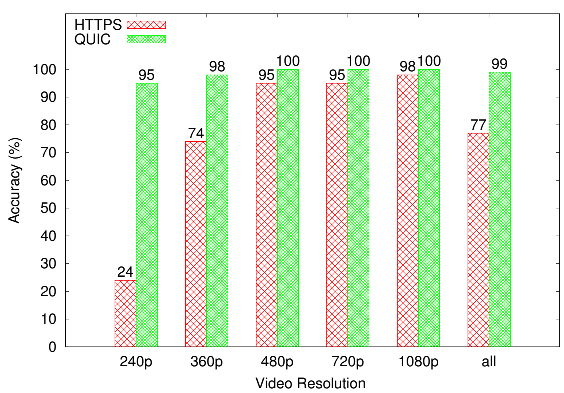


Figure 3: Silhouette Accuracy

This paper presents Silhouette , a heuristic video identification method suitable for YouTube streaming. Silhouette classifies based only on the flow "shape" in terms of data rates and payload lengths, making it suitable for both HTTPS/TCP-based and QUIC/UDP-based streaming. Since Silhouette does not require any application layer information nor server name indication (SNI), it is effective even for encrypted videos. Moreover, because Silhouette does not classify based on packet inter-arrival times, it should be robust for all link types. Silhouette is light-weight – it does not require any high-CPU intensive processing (e.g., deep packet inspection (DPI)) nor external cross layer signaling, making it suitable for ISP middle-boxes that need to classify and then treat video flows.

Evaluation with a "ground truth" dataset of 660 YouTube videos covering a range of content and encoding rates shows classification accuracy near 90% and closer to 100% for high-definition videos.

While Silhouette should be effective regardless of provider (e.g., Netflix versus YouTube), future work includes evaluation over more content providers. Future work also includes a real-time implementation using a Linux Traffic Control (TC) service, and deployment into middle-boxes at an ISP to detect and treat video flows.

REFERENCES

[1] A. Langle et al. 2017. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In *Proceedings of ACM SIGCOMM*. Los Angeles, CA, USA.
[2] D. Bhat, A. Rizk, and M. Zink. 2017. Not So QUIC: A Performance Study of DASH over QUIC. In *Proceedings of NOSSDAV*. Taipei, Taiwan.
[3] Cisco System Inc. 2017. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update*. Technical Report. Cisco System Inc.
[4] M. Claypool. 2008. Network Characteristics for Server Selection in Online Games. In *Proceedings of MMCN*. San Jose, CA, USA.
[5] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki. 2016. Measuring Video QoE from Encrypted Traffic. In *Proceedings of ACM SIGCOMM IMC*. Santa Monica, CA, USA.
[6] A. Kakhki, F. Li, D. Choffnes, A. Mislove, and E. Katz-Bassett. 2016. BingeOn Under the Microscope: Understanding T-Mobile's Zero-Rating Implementation. In *Internet-QoE Workshop*. Florianopolis, Brazil.
[7] M. Katsarakis, R. C. Teixeira, M. Papadopouli, and V. Christophides. 2016. Towards a Causal Analysis of Video QoE from Network and Application QoS. In *Proceedings of the Workshop on QoE-based Analysis and Management of Data Communication Networks*. Florianopolis, Brazil.
[8] J. Khalife, A. Hajjard, and J. Diaz-Verdejo. 2013. Performance of OpenDPI in Identifying Sampled Network Traffic. *Journal of Networks* 8, 1 (2013), 71–81.
[9] D. Krishnappa, D. Bhat, and M. Zink. 2013. DASHing YouTube: An Analysis of using DASH in YouTube Video Service. In *Proceedings of LCN*.
[10] F. Li, , J. Chung, X. Jiang, and M. Claypool. 2018. Who is the King of the Hill? Traffic Analysis over a 4G Network. In *Proceedings of IEEE International Conference on Communications (ICC)*. Kansas City, MO, USA.
[11] F. Li, M. Claypool, and R. Kinicki. 2015. Treatment-based Traffic Classification for Residential Wireless Networks. In *International Conference on Computing, Networking and Communications (ICNC)*. Anaheim, CA.
[12] F. Li, A. Razaghpanah, K. Molavi, N. Akhavan, D. Choffnes, P. Gill, and A. Mislove. 2017. liberate,(n): A Library for Exposing (Traffic-Classification) Rules and Avoiding them Efficiently. In *Proceedings of the Internet Measurement Conference*.
[13] A. Moore and K. Papagiannaki. 2005. Toward the Accurate Identification of Network Applications. In *Passive and Active Measurement Workshop*. Boston, MA.
[14] T.T.T. Nguyen, G. Armitage, P. Branch, and S. Zander. 2012. Timely and Continuous Machine-learning-based Classification for Interactive IP Traffic. *IEEE/ACM Transactions on Networking (TON)* 20, 6 (2012), 1880–1894.
[15] T. T. T. Nguyen and G. Armitage. 2008. A Survey of Techniques for Internet Traffic Classification Using Machine Learning. *IEEE Communications Surveys Tutorials* 10, 4 (April 2008).
[16] A. Reed and M. Kranch. 2017. Identifying HTTPS-Protected Netflix Videos in Real-Time. In *Proceedings of CODASPY*. Scottsdale, AZ, USA.
[17] J. Terrell, K. Jeffay, F.D. Smith, J. Gogan, and J. Keller. 2009. Passive, Streaming Inference of TCP Connection Structure for Network Server Management. In *Proceedings of the International Workshop on Traffic Monitoring and Analysis*.
[18] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and Y. Xiang. 2013. Internet Traffic Classification by Aggregating Correlated Naive Bayes Predictions. *Journal of IEEE Transactions on Information Forensics and Security* 8, 1 (January 2013), 5–15.