Real-Time Traffic Analysis over Mobile Wireless Networks

ABSTRACT

As radio link capacities increase, mobile service providers have deployed middle-boxes to achieve higher throughput over mobile networks. However, the growth in application flows using encryption such as TLS/SSL has limited the ability of middle-boxes to correctly classify traffic, causing middle-boxes to mistreat traffic. This paper presents *AppDetect*, a real-time, scalable plug-and-play solution for middle-boxes that automatically classifies flows without user intervention and privacy leakage. AppDetect primarily differentiates traffic through text searching during HTTP and TLS/SSL handshakes, reverse DNS caching, and flow-level statistics. We use AppDetect to collect traffic from 5000+ devices producing 2.5+ million flows from a major wireless carrier and make insightful observations on cross-layer network characteristics for IPv4/6, HTTP/S and QUIC traffic. To the best of our knowledge, our QUIC traffic analysis is the first from a carrier's perspective. We determine the top LTE applications and highlight their behavior. These findings can be used to develop new transport protocols or design simulations for better mobile network QoS support. We further propose a real-time detection algorithm that works with encrypted flows to detect High Definition video. This may be helpful for network providers to choose traffic optimization strategies that work with this growing source of network traffic.

1. INTRODUCTION

The low latencies and high link capacities of Long Term Evolution (LTE) networks have propelled subscribers to access many Internet services through their mobile devices. Huang *et al.* [12, 30] observed LTE traffic with Quality of Service (QoS) requirements beyond that of traditional Web traffic, including only game, voice and video traffic. Although LTE and LTE Advanced (LTE-A) have improved nominal link rates up to 300 Mbps, applications rarely reach the top throughput rates in uncontrolled environments and usually experience degraded performance [14].

To improve application throughput over LTE, many carriers have introduced performance-enhancing proxies (PEPs) as middle-boxes in their core networks [7, 8, 14, 21]. PEPs transparently split one end-to-end TCP connection into two and seek to improve Web performance by caching. However, due to the growth in firewalls and TLS/SSL use, it is increasingly difficult for PEPs to identify the content type of the traffic, and, hence, choose the correct traffic shaping strategy that will provide better QoS. Video playout rates, in particular, are typically limited by the video codec. So, instead of using TCP acceleration for video, such self rate-limited traffic should be optimized to lower the utilization of radio links [7] during congestion. Hence, blindly increasing throughput in the absence of correct traffic classification may not improve, and may even hurt, performance. Furthermore, PEPs face additional challenges when trying to support QoS. Without knowledge of application content, PEPs cannot set Type of Service (ToS) or Differentiated Services Code Point (DSCP) bits for connections to subscribers. Thus, wireless eNodeBs cannot prioritize traffic, such as video traffic, even when supporting traffic differentiation through QoS Class Identifiers (QCIs). Another problem is that PEPs confuse end-host bandwidth estimators. In such cases, video servers mistakenly send high definition (HD) videos to PEPs because of the high capacity and low latency link between PEPs and ISP servers. These HD videos increase the burden on radio links, causing subscribers to experience long buffering times before their videos can play.

Last but not least, due to privacy and performance concerns, there is a dearth of public data on real-time traffic differentiation data. Data scientists require network traces for training data, but carriers cannot release traces which contain customers' private information. Methods are needed to collect and share aggregated traffic statistics without privacy leakage.

In short, we believe there is a need to have a scalable, light-weight, real-time application classification service in middle-boxes to support QoS requirements in future mobile networks (e.g., 5G) [10], particularly for video transmission services. In addition, before deploying any QoS enhancement services, carriers need a better understanding of the "state of the art" of traffic over current

mobile networks.

We have begun to address this problem by cooperating with a world leading tier-1 wireless carrier for data acquisition and development of an application classification service, making the following contributions:

- Develop *AppDetect*, a light-weight, scalable, real-time flow characterization framework to collect key flow statistics such as volume, duration, throughput and network latency, that is easy to deploy with "in-line" middle-boxes. The collection process does not store packet details or subscriber information, reducing privacy leakage.
- Propose and demonstrate a real-time video traffic detection algorithm for encrypted flows, specifically designed to handle high definition (HD) video. This traffic detection algorithm can be used by middle-boxes to first classify and then choose appropriate traffic treatment strategies.
- Store collected flow information in a cloud-based database, providing a "data lake" that data scientists can access to develop new services without carrying out redundant efforts to collect and process large amounts of traffic from a production network.
- Gather traces from a major wireless carrier's network containing traffic from over 5000 devices contributing over 2.5 million flows. These traces form the basis for investigations into cross-layer traffic characteristics and application behaviors over a mobile network. Our analysis includes IPv4/6, HTTP/S and emerging QUIC traffic. To the best of our knowledge, our QUIC analysis is the first from a carrier's perspective.

Our analysis of the collected mobile network data provides for the following observations:

- IPv6 contributes 53% of total traffic in volume, with Google's IPv6 traffic alone contributing 23%. This implies that U.S. mobile networks have migrated from IPv4 to IPv6.
- HTTPS accounts for 38% of traffic in volume, higher than HTTP (31%). More than 53% of traffic is protected by TLS/SSL: 38% from HTTPS, 8% from HTTP2, and 8% from QUIC.
- QUIC contributes nearly 64% of UDP traffic in volume, while DNS still dominates UDP traffic in terms of the number of flows.
- YouTube accounts 22% of total traffic in volume, the largest of any kind of traffic. Google, Facebook and Apple are the top 3 contributors to traffic volume.

The rest of the paper is organized as follows: Section 2 summarizes related work in this area; Section 3 describes the architecture of our application detection service; Section 4 shows the cross-layer flow statistics and network characteristics from wireless packet traces collected in the field; Section 5 investigates the application behaviors over mobile networks; Section 6 describes our proposed video classification algorithm; Section 7 discusses the possible future work; and Section 8 summarizes our conclusions.

2. RELATED WORK

Accurately identifying and categorizing network traffic according to application type is of primary importance for many fundamental network research areas, such as QoS, traffic shaping, and intrusion detection. In the past two decades, researchers and network operators have designed and deployed hundreds of traffic classification algorithms.

Unfortunately, most traffic classification algorithms do not meet the real-time application detection requirements for middle-boxes that work with today's Internet. For example, port-based [24] or host name (including Server Name Identification) approaches are often inaccurate and unable to discern MPEG-DASH video traffic from Web traffic. Deep packet inspection approaches, which can yield 99% accuracy in certain cases [16], are generally computationally intensive and are ineffective on novel applications and/or encrypted traffic. Statistics-based approaches [27] and user behavior-based approaches [15] require substantial training data and feature selection in order to work. Moreover, the accuracy of such approaches tends to be sensitive to parameter tuning, with any mistake in tuning or in using a non-representative training set yielding inaccurate results.

More recently Huang *et al.* characterized mobile network usage and performance with a 10-day packet capture in one region [11, 12]. While helpful to better understand mobile network usage, their analysis only focused on TCP, especially HTTP traffic, with no application layer traffic analysis, such as video. Our work provides a more recent trace, and seeks to understand the share of video traffic over current mobile networks. Moreover, Huang *et al.*'s approach requires HTTP headers captured from production networks, and their trace-based method may not be able to provide traffic analysis nationwide over a long time period. Last but not least, nationwide packet capture over a long time period is expensive and impractical, and does not scale well.

Xu *et al.* analyzed app usage on smart phones by analyzing the HTTP Agent fields in HTTP headers [30]. While effective for illustrating HTTP use in 2010, their method does not work for the massive amounts of encrypted (HTTPS) traffic that has emerged since then. Our more recent observations show more HTTPS traf-

fic than HTTP on mobile networks, making it unlikely that Xu *et al.*'s method can provide an accurate representation of app usage over today's mobile networks.

Moreover, neither Huang *et al.* [12] nor Xu *et al.* [30] provide results for emerging Quick UDP Internet Connection (QUIC) [13] traffic. Being sent over UDP, the deployment of QUIC imposes a new challenge for network management and traffic classification. To the best of our knowledge, no industrial nor academic researchers provide QUIC traffic analysis over mobile networks, making our results the first presented from a carriers' perspective.

From the other direction, researchers and content providers also monitor mobile service providers which may deploy incorrect traffic differentiation schemes [8, 14, 21, 23]. Flach *et al.* find that globally 7% of connections are identified as policed [8]. Choffnes *et al.* even identify a U.S. carrier that mistakenly rate limits traffic which is not video, and mis-charges subscribers with degraded video streaming from non-partners' sites [14, 21]. The above research provides motivation for enhanced middle-boxes with the ability to classify traffic in real-time from the reverse direction. Without accurate traffic classification, middle-boxes can only blindly do traffic differentiation, leading to QoS degradation, especially for video quality.

3. METHODOLOGY

This section describes the high-level design of our scalable real-time traffic classification service: *AppDetect*.

Figure 1 depicts the AppDetect deployment architecture. AppDetect receives traffic mirrored from in-line middle-boxes (e.g., switches or routers), whereupon it builds up flow-level information based on 5-tuples (source port, source IP, destination port, destination IP, and transport protocol). Application types are detected by reverse DNS and basic text matching with the `HOST` headers in HTTP Requests and Server Name Identifications (SNIs) during TLS/SSL handshakes. Subsequent packets matching the same flow are used to build up flow statistics. When a flow sends a TCP FIN/FIN-ACK or does not transfer a packet for 60 seconds, the flow is marked "terminated", whereupon AppDetect writes the flow record (statistics and application type) into a cloud-based Cassandra[1] database, currently in the carrier's private cloud, for off-line analysis.

As emphasized earlier, AppDetect collects no personal identifiable information. The application is only identified by type and the flow record only contains basic traffic statistics. The end host IP addresses are private for the carrier and are typically reassigned by the DHCP servers every few hours and subscribers' identities cannot be determined through stored client IP addresses alone.

[1] `http://cassandra.apache.org/`

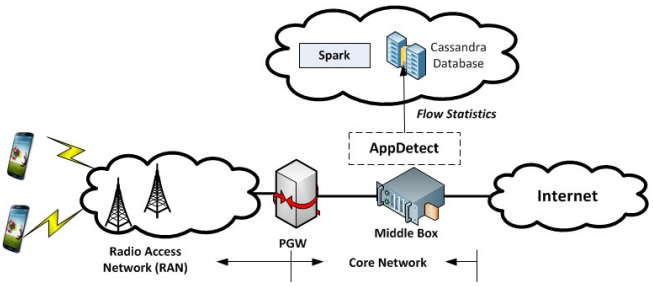


Figure 1: Deployment Architecture of AppDetect

Unfortunately, due to the wide deployment of cloud services, reverse DNS using servers' IP addresses alone does not provide sufficient information to determine application types. For example, the popular streaming video provider Netflix uses Amazon Web Services (AWS) to store video [1], so just using reverse DNS may mistakenly assume such traffic is from Amazon. Thus, AppDetect uses HOST names and SNIs to accurately detect application type. Host name and SNIs are the *only* Application Layer header contents recorded by AppDetect. Note, HOST and SNIs are public, well-known data and collecting such fields does not expose private information on the subscribers. Unlike related approaches [12, 30], AppDetect does not store any packet payload information in a persistent format, thus,reducing private information leakage by design.

AppDetect also provides an offline mode for validation and performance measurement. When running in offline mode, AppDetect reads packets previously captured in a *libpcap* format (e.g., via tcpdump or Wireshark) instead of using live traffic. For this paper, our offline setup was configured with a HP Proliant 460c Gen9 server with 128 GB RAM and a dual socket 2.60GHz ten-core Intel(R) Xeon(R) E5-2660 v3 CPU, running Ubuntu 14.04 with a 3.19.0-25-generic Linux kernel installed. AppDetect in this configuration and running in a single thread can process a 1-hour, 50 GB trace in about 48 seconds. We thus infer that AppDetect has a modest processing overhead, allowing for ease of deployment with most in-line middle-boxes.

4. NETWORK CHARACTERISTICS

We tested AppDetect with a major carrier's wireless network during the daytime (local time 9:00am-4:00pm) on two successive days, November 7-8, 2016. The carrier mirrored traffic from one end-user device (UE) pool with 5000+ to AppDetect. That UE pool serves a large geographic area in the South Central United States, large enough to provide a general picture of the traffic

over mobile U.S. networks. In total, AppDetect analyzed 85 GB of data, containing over 2.56 million flows. As described in Section 3, with respect to customer privacy, AppDetect did not collect subscribers' IDs, geographic locations, device types nor mobile numbers.

4.1 IP Protocol Version

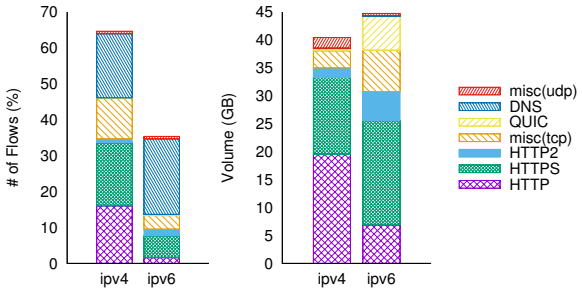


Figure 2: Application Layer Protocol Distribution

Figure 2 depicts a comparison of traffic for IPv4 and IPv6. The graph on the left is the number of flows (percent) and the graph on the right is the traffic volume (GBytes). 64.1% of the flows (1.6 million) are IPv4 compared to 35.9% for IPv6. However, the IPv6 volume is slightly larger than IPv4. The mean size of IPv6 TCP flows is 81.1 KBytes, 7 times larger than the mean size of IPv4 TCP flows which are 11.2 KBytes. With the growth in deployment of smart phones, tablets and Internet of Things (IoT) devices over the last 3 years, globally 92% of mobile devices are IPv6 capable [5]. With continued support for IPv6 in Android, iOS and U.S. ISP's, we expect to observe more IPv6 traffic over mobile networks in the near future.

In contrast, since most smart phones are supporting IPv4 and IPv6 simultaneously, approaches use private IP address within a short time period as subscriber IDs [12] are increasingly less effective. Because all traffic to the same device shares the same queue inside the eNodeBs, middle-boxes must be able to associate both IPv4 and IPv6 traffic to the same subscriber to achieve per-device rate control and avoid buffer bloat [28].

4.2 Use of Encryption

Figure 2 shows a breakdown of the major application types. The HTTP-family of protocols still dominates mobile networks in terms of number of flows, and accounts for 77.2% of the overall traffic volume. However, 59.8% of HTTP traffic is HTTPS or HTTP2, both of which are protected by encryption via TLS/SSL. This means techniques that classify traffic based solely on the HTTP Agent field [30] are not effective since only 30.2% traffic is unencrypted HTTP.

QUIC [26], a multiplexed stream transport protocol over UDP, accounts 7.1% of traffic volume, most of it over IPv6. From a middle-box flow perspective, QUIC is similar to TCP + TLS + HTTP/2 over UDP, and we consider QUIC an Application Layer protocol in this study. Note, at the time of this study, Google mainly uses QUIC for YouTube services, and collects statistics from Android devices. Section 5.4 discusses QUIC for YouTube in more detail.

In total, more than 53% of traffic is protected by encryption, with the expectation that the fraction of encrypted traffic will grow. Thus, there is a need for additional research that can differentiate QoS classes for encrypted traffic in order to provide appropriate QoS support.

4.3 Flow Statistics

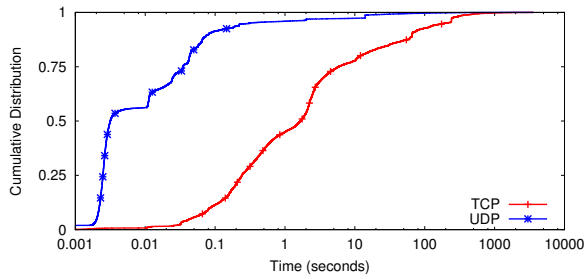
Of the 85 GBytes total data, 99.9% is either TCP or UDP, with the rest (e.g., ICMP, GTP, SCTP, or IPSpec) only contributing an insignificant 10 MBytes of data. In terms of the number of flows, 990K (38.9%) are UDP DNS flows, although DNS traffic is insignificant in terms of volume.

We analyze four key flow statistics: duration, volume (size), rate, and average packet length. Duration is calculated from the time between the first packet and the last packet of a flow. Volume is calculated from the summation of packet lengths within the flow, including IP and transport layer headers. Rate is the throughput calculated from the volume divided by the duration. Average packet length is calculated by dividing volume by the total number of packets in a flow. These characteristics are important for tuning performance in an LTE network, such as eNodeB scheduling, per-flow billing, radio access network (RAN) optimization and video detection over encrypted channels.

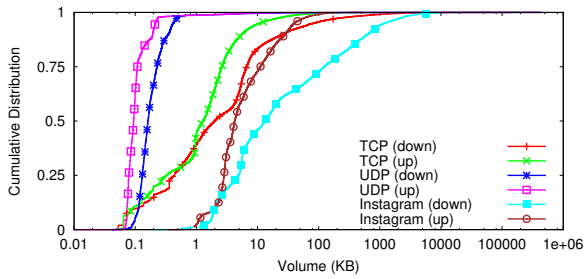
Note, we observe correlations among volume, duration and rate within a flow, similar to those reported in other measurement studies [12, 31]. Due to space constraints, we do not present such analysis here.

Duration Figure 3(a) shows a cumulative distribution function (CDF) of TCP and UDP flow durations. The x-axis is the flow duration in seconds and the y-axis is the cumulative distribution. Note the x-axis is logscale. There are two trendlines depicted, one with the cumulative distribution of TCP flows and the other with the cumulative distribution of UDP flows. From the graph, most flows are "dragonflies" (lasting less than 2 seconds) [2] – 55% of TCP flows and 95% of UDP flows are less than 2 seconds. Both TCP and UDP flow durations exhibit heavy-tailed distributions. None of the flows are "tortoises" (lasting more than 15 minutes) [2]. However, 11.2% of TCP flows and 0.49% of UDP flows last longer than 1 minute.

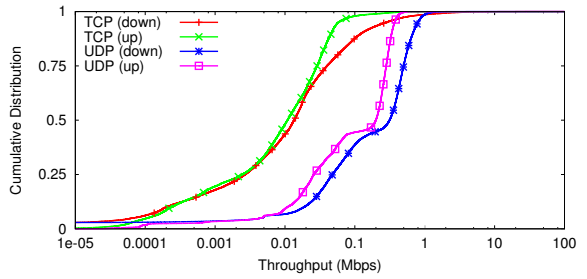
Size Figure 3(b) depicts CDFs of flow volume. The x-axis is the flow volume in KBytes and the y-axis is the cumulative distribution. Note the x-axis is logscale.



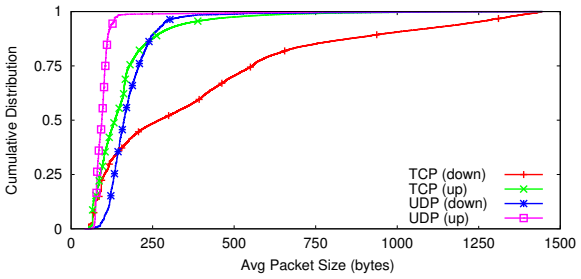
(a) Distribution of Flow Duration



(b) Distribution of Flow Volume



(c) Distribution of Flow Rate



(d) Distribution of Average Packet Length

Figure 3: Transport Layer Flow Analysis

There are six trendlines depicted, four of which are for TCP and UDP, both up (from the UE) and down (to the UE). Although most flows are small, the CDFs of both UDP and TCP flow volumes show heavy-tailed tendencies. 90% of TCP flows are less than 27 KBytes, and 90% of UDP flows are less than 0.5 KBytes. The uplink TCP flow sizes are somewhat smaller than downlink TCP flow sizes, but the difference is not as great as might be expected. While 90% of TCP uplink flows are smaller than 5 KBytes, the high link capacity of LTE networks coupled with the popularity of social media apps produce more uplink traffic than downlink for many TCP flows. To illustrate this, Figure 3(b) also plots the uplink and downlink flow volumes observed for Instagram apps. Instagram volumes for both uplink and downlink are much higher than the volumes of other TCP flows. Surprisingly, 13.5% of TCP flows are never successfully established, containing fewer than 3 packets – future work should investigate the cause of these TCP connection failures.

Meanwhile, UDP flow size distributions also show heavy-tail tendencies, and some of the UDP flows are more than 10 MBytes. These high volume flows are also long lasting (60+ seconds), contributing over 7 GBytes, about 80% of the overall UDP traffic. Starting in 2016, Google has deployed QUIC [13] in Chrome-based browsers, carrying YouTube and other Google-served traffic in North America. Section 5.4 provides more analysis of QUIC and YouTube traffic.

Rate Figure 3(c) depicts CDFs of flow rate (throughput). The x-axis is the flow volume in KBytes and the y-axis is the cumulative distribution. Note the x-axis is logscale. There are four trendlines depicted, for both TCP and UDP, up and down. Because of the asymmetric channel bandwidth and the encoding schemes for uplink versus downlink in LTE networks, the uplink rate is generally lower than downlink rate. 90% of TCP flows have rates less than 285 Kbps down and 60 Kbps up, while 90% of UDP flows have rates less than 815 Kbps down and 379 Kbps up.

Applications running with a request/response protocol (e.g., HTTP) might never send enough data to fill a high capacity LTE pipe [4], making it difficult to improve TCP throughput by simply adding more radio resources. TCP "splitting" or TCP "acceleration in a middle-box" may improve link utilization over LTE networks [28]. But, considering flow sizes in Figure 3(b), there is little opportunity to improve the throughput of small flows since all data can be fit into the initial congestion window (cwnd) during the first burst, and it may be more beneficial treat "elephant" flows that have large sizes [18, 28].

Average Packet Length Figure 3(d) depicts CDFs of average packet size. The x-axis is the average packet size in bytes and the y-axis is the cumulative distribution. There are four trendlines depicted, for both TCP and UDP, up and down. All packets are less than 1380 bytes. From the graph, the large majority of UDP packets are small (less than 200 bytes – usually DNS request/response packets), while the large majority of

uplink TCP packets are also small (less than 100 bytes – usually TCP ACKs).

In mobile networks, carriers generally want a maximum segment size of 1380 bytes to account for the GTP protocol (used for radio network signaling) overhead. Large amounts of small packets and unnecessary packet fragmentation can severely degrade TCP performance over LTE networks by causing under utilization of radio resources. Such behavior may arise from ill-configured Web servers or proxies which can generate many small packets, degrading LTE performance. Future work may seek to develop heuristics to detect and gracefully correct inappropriate server settings (such as the wrong MTU) in carriers' middle-boxes.

4.4 Network Latency

Passive round-trip time (RTT) estimation in middle-boxes is not trivial [3], requiring record tracking of sequence numbers in both directions for TCP flows and may not be possible for one-way UDP flows. Moreover, while an important network performance metric, RTT is not generally useful to help classify flows or applications. Thus, AppDetect only measures the initial RTT for TCP flows, easily obtained through TCP's three-way handshake, while more heavy-weight RTT measurements are a future option. AppDetect also infers RTTs for DNS request-response pairs.

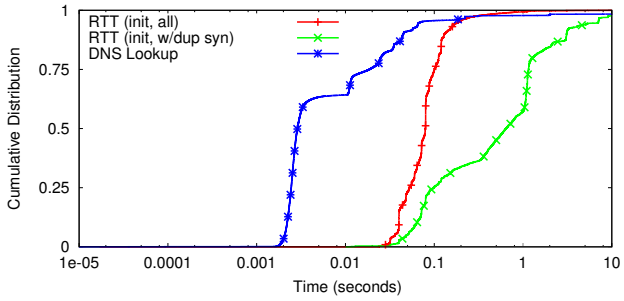


Figure 4: Distribution of Initial TCP RTT and DNS Request-Response

TCP Initial RTT Figure 4 depicts a CDF of TCP initial RTTs measured through the TCP three-way handshakes. The x-axis is the time in seconds and the y-axis is the cumulative distribution. Note the x-axis is logscale. There are two TCP trendlines depicted, one for a "normal" three-way handshake and the other for a three-way handshake with a duplicate SYN packet. The median of the TCP initial RTT is 79 ms. This data may be valuable for parameter selection for RTT based congestion control algorithms (e.g., BBR) [4]. The TCP initial RTT distribution shows a heavy-tailed tendency. Since propagation delays over 100 ms are unrealistic over LTE [11], further investigation suggests the large TCP initial RTTs are due to:

1. Retransmission of SYN/SYN-ACK. About 1% of TCP flows have at least one packet retransmission during the three-way handshake. Such retransmissions significantly increase the initial RTT measurements, as shown by the second RTT trendline in Figure 4.

2. Backwards compatible 2G/3G devices. 210 TCP flows without SYN/SYN-ACK retransmission also experienced initial RTTs larger than 900 ms, the amount observed on 2G/3G networks [11].

DNS Lookup Figure 4 also shows the DNS lookup time latencies, calculated as the time between the observation at the capture device of a DNS request and the subsequent response. Effectively, this is approximately the RTT between the middle-box and DNS server. Because DNS traffic is typically only an exchange of two packets, it cannot be used to measure the latency between middle-boxes and UEs. From the graph, the RTTs between the middle-box and DNS servers are short – 64% of DNS response times are less than 10 ms. Since the carrier provides DNS servers in its own core network, the short DNS request time confirms the time-efficiency of DNS caching for mobile core networks [12].

5. MOBILE APPLICATIONS

This section analyzes the application behaviors observed by AppDetect.

AppDetect classifies flows based on a minimum of information including server names, SNIs during TCP and TLS/SSL handshakes, and HOST names in HTTP requests, without any post processing. Therefore, AppDetect is unable to achieve 100% accuracy – 11.9% of the traffic (9.1 GBytes of TCP and 0.9 GBytes of UDP) is marked as "unknown". However, from the QoS enhancement perspective, AppDetect accuracy is effective in identifying (and allowing for treatment of) video flows – based on our offline emulation, QoS enhancements (e.g., pacing [23]) for detected video traffic can reduce radio utilization by 5% for heavily congested cell sectors, a significant savings.

From the view of middle-boxes, flows from either YouTube or Netflix have similar QoS requirements (i.e., both primarily provide videos) and special differentiation based on content provider is not needed. However, AppDetect is still able to provide analysis of traffic from well-known content providers, providing a better understanding of current traffic in mobile networks.

5.1 Top Content Providers

Figure 5 depicts the traffic volume from the top 3 content providers: Google, Facebook and Apple, observed during our tests. The y-axis is the traffic volume

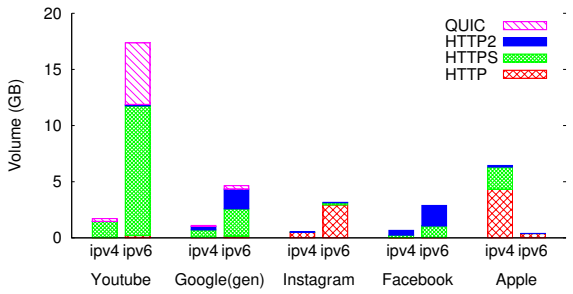


Figure 5: Traffic from Google, Facebook and Apple.

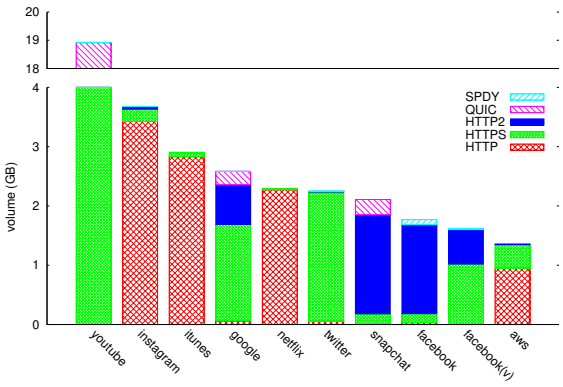


Figure 6: Traffic from Top 10 Applications.

in GBytes and the x-axis shows clusters for the top 3, separated into IPv4 and IPv6 columns. In addition, because of their popularity, YouTube and Instagram are pulled out separately – the "Google (gen)" group summaries all traffic from Google other than YouTube, and "Facebook" contains all Facebook traffic except Instagram.

From the graph:

- Google (including YouTube) is the largest content provider over the carrier's network. Most of Goggle's traffic is protected by TLS/SSL and is mainly over IPv6. In total, there is only 430 MBytes of unencrypted HTTP traffic from Google and YouTube. Google's 22 GBytes of IPv6 traffic accounts for 49.2% of all total IPv6 traffic, 9x higher than Google's IPv4 traffic (2.7 GBytes). It appears that Google has moved its default services into IPv6 for the North American market.

- A small number of QUIC flows (1250 QUIC IPv6) deliver 5.6 GBytes of data from YouTube. Although only recently announced (Q2 of 2016), QUIC already constitutes 25% of Google's overall traffic. This requires ISPs to have a better understanding of QUIC-based applications over their networks for planning and treatment.

- Instagram is the most popular service from Facebook – it constitutes about half of all Facebook traffic. From an implementation perspective, Instagram is mainly over HTTP unlike most other Facebook services. Similar to Google, Facebook also appears to have migrated most of their traffic to IPv6 in North America.

- Traffic from Apple consists of a variety of different services such as iTunes, Apple Maps, and iCloud, most of which is over IPv4. Our analysis (not shown) reveals that iTunes constitutes the majority of Apple's traffic.

5.2 Top Applications

Figure 6 depicts the traffic volume from the top 10 mobile apps/servers during our test: YouTube, Instagram, iTunes, Google, Netflix, Twitter, Snapchat, Facebook, Facebook Video, and Amazon Web Services (AWS). The y-axis is the traffic volume in GBytes and the x-axis shows a column for each application, sorted by volume from most to least going left to right. Note, YouTube alone has about 19 GBytes of traffic, so the y-axis is "broken" from 4-18 Gbytes. The Google group only contains Google Web traffic, excluding products such as YouTube, Play, GMail and Google Docs. Facebook has 3 applications in the top 10 list: Instagram, Facebook "generic", and Facebook Video (labeled as "facebook (v)" in the figure). The label "facebook" consists of 280 MBytes of Facebook Messenger traffic, but because Facebook Messenger is migrating to TLS/SSL, not all Messenger traffic can be differentiated from Facebook "generic" traffic, hence they are grouped together.

From the graph:

- The top 10 applications account for about 40 GBytes of traffic, or 46.4% of the total traffic.

- Videos and social apps are the two most popular mobile application types. Among the top 10 list, 3 are video applications (YouTube, Netflix, and Facebook Video) and 4 are social apps (Instagram, Twitter, Snapchat, and Facebook "generic"). Some social apps, such as Snapchat, produce a lot of video content also.

- Snapchat is the only application using QUIC other than Google, but all Snapchat QUIC traffic is served by Google servers. All Snapchat traffic detected is encrypted. Thus, we classify Snapchat QUIC traffic based on a build dictionary for possibly Google-owned Snapchat servers in offline testing. This may result in a high false positive rate for Snapchat.

- Cloud and CDN providers make traffic classification difficult. AppDetect shows 1.5 GBytes of data is from Akamai, higher than Amazon AWS (1.4

GBytes). However, Akamai provides services for iTunes, iCloud, Instagram, and Facebook. Thus, we have to deduct traffic from "well-known" applications from Akamai's volume total, leaving Akamai as the 12th largest, and therefore is not shown in the top 10 list in Figure 6.

5.3 YouTube Traffic

Since video traffic will make up 50% or more of traffic in near future, identifying video is critical for improving the overall QoS over mobile networks [10]. As Section 5.2 shows, YouTube, Netflix, and Facebook already contribute 23 GBytes (27%) of the total observed traffic. Thus, middle-boxes must detect video flows if they then hope to shape their traffic [21, 23].

Figure 7 shows a scatter plot of the size versus time for HTTPS-based YouTube flows with different SNIs. The x-axis is the flow duration time in seconds and the y-axis is the total payload size in MBytes. Note both axes are logscale. There are 4 data sets depicted: googlevideo, youtube.com, ytimg.com and everything else, where all flows are from YouTube. From the graph, there is no obvious correlation between payload size and flow duration, even for flows from the same domain server.

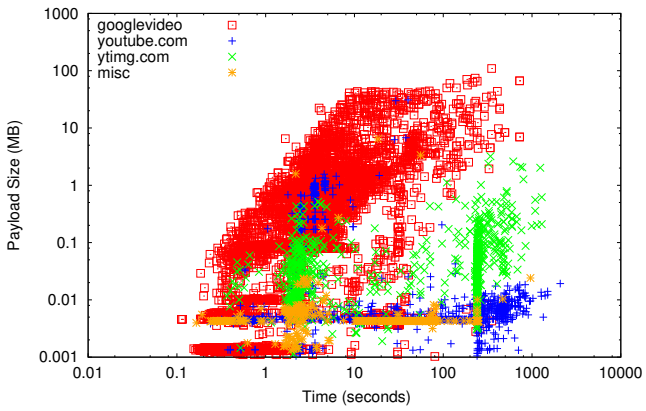


Figure 7: Payload Size versus Duration for HTTPS-Based YouTube Flows.

Table 1: SNIs from YouTube Domain

pattern	count	freq (%)
*r(\d+)—sn-*googlevideo.com*	11682	77.64
**.youtube.com* or **.youtue.be*	1754	11.66
**.ytimg.com*	758	5.04
redirector.googlevideo.com	620	4.12
manifest.googlevideo.com	108	0.72
youtube-nocookie.com and etc.	42	0.31
without SNIs	77	0.51
Total	15046	100.00

Table 1 summarizes the SNIs from the HTTPS-based YouTube flows. From the table, flows with SNIs like *r(\d+)—sn-*googlevideo.com* are likely carrying video content, while flows with SNIs like **.youtube.com* provide customized YouTube Web pages. Flows with SNIs like *ytimg*[2] store thumbnail images and similar objects used by YouTube's video players. Flows with SNIs like *manifest.googlevideo.com* provide manifest files for video clips, and likely do not contain video segments.

Figure 8 compares key flow statistics for HTTPS-based YouTube flows, broken down by the same SNIs. The "misc" group includes flows with SNIs matching *redirector.googlevideo.com*, *manifest.googlevide.com*, or *youtube-nocookie.com*, and are unlikely to carry video content.

Figure 8(a) depicts the cumulative distributions of the flow durations. The x-axis is the duration time in seconds and the y-axis is the cumulative distribution. Note the x-axis is in logscale. From the figure, flow duration by itself is not a good differentiator for video flows since the duration of googlevideo.com flows (which actually are videos) is even smaller than ytimg.com flows (which actually are just images). Because ytimg.com stores icons and thumbnail images for YouTube web pages, ytimg.com and youtube.com flows only terminate when subscribers close their Web browsers or player apps. Thus, ytimg.com flows live even longer than short video flows.

Figure 8(b) depicts the downlink volume of HTTPS-based YouTube flows. The x-axis is the volume in KBytes and the y-axis is the cumulative distribution. Note the x-axis is in logscale. As expected, googlevideo.com flows are larger than other flows – 18% of googlevideo.com flow are larger than 1 MByte. However, 31% of googlevideo.com flows are smaller than 5 KBytes – presumably, flows so small are likely only carrying a preview image or some video meta data rather than actual video content.

Figure 8(c) shows the downlink throughput of HTTPS-based YouTube flows. The x-axis is the throughput in Kbps and the y-axis is the cumulative distribution. Note the x-axis is in logscale. From the graph, the throughput of googlevideo.com flows is much higher than the others. This suggests that throughput might be a good candidate to detect videos. Note, the vertical *brown* dashed line at 1.5 Mbps in the figure marks the bitrate of a YouTube video encoded at 480p [9]. About 25% of googlevideo.com flows yield a throughput more than 1.5 Mbps. One tier-1 carrier rate-limits all video flows to 1.5 Mbps as its video optimization solution [14]. In this case, if their subscribers' video players cannot automatically adapt to lower quality segments, 25% of YouTube video flows would be punished by their traffic policing policy, and the subscribers would experience long buffering times with multiple stalls.

Figure 8(d) depicts the average per packet Transport

[2]ytimg is an acronym for "YouTubeIMaGe."

layer payload length for HTTPS-based YouTube flows. The x-axis is the average payload length and the y-axis is the cumulative distribution. From the graph, googlevideo.com flows consist of large packets – 60% of googlevideo.com flows have a payload of 1000+ bytes per packet. This suggests per packet payload length might be a good metric for video detection of encrypted flows.

5.4 QUIC-based YouTube Traffic

QUIC was designed to provide security protection equivalent to TLS/SSL and reduce connection latency, while still providing bandwidth estimation to avoid congestion.[3] However, at the time of the study, only Chrome-based browsers and Google servers provide services over QUIC. Thus, AppDetect classifies UDP flows to/from Google servers on port 443 as QUIC traffic. Since the QUIC header does not contain any information similar to SNIs, AppDetect uses reverse DNS to divide QUIC traffic into two groups: i) *QUIC (youtube)* – QUIC traffic from YouTube servers, and ii) *QUIC (misc)* – QUIC traffic from other Google servers. Our offline observations show QUIC (misc) carries performance statistics on Chrome browsers.

Figure 9(a) and Figure 9(b) compare the flow duration and volume of QUIC traffic with other UDP-based traffic: DNS and UDP (misc) which is not DNS or QUIC. In Figure 9(a), the x-axis is the flow duration in seconds and in Figure 9(b), the x-axis is the flow volume for both uplink and downlink in KBytes. In both graphs the y-axis is the cumulative distribution.

As Figure 9(a) shows, 40% of UDP (misc) flows are effectively zero seconds in that these flows contain only one packet. Since about half of these single packet UDP flows have either source port or destination port as 0, we suspect these flows are probes for DDoS attacks.[4].

From Figure 9(b), QUIC can be distinguished from other UDP flows based on volume, but doing so in real-time is a challenging task for middle-boxes since the total volume is only known at the end of the flow. Note also that since these total volumes are significant, this suggests that the focus on treating and classifying mobile traffic should not only focus on the TCP protocol [12, 30].

6. IDENTIFY VIDEO TRAFFIC

Text-based matching rules during HTTP and TLS/SSL handshakes can reveal some flow content types [21, 23], but not all flows matching the rules transmit actual video content. As Figure 7 shows, some HTTPS flows with matched SNIs may not carry video segments: the flows are either too small (less than 10 KBytes) or too short (less than 5 seconds) for video. Moreover, SNI checking is not reliable – it cannot detect video flows from previously unseen SNIs (i.e., SNIs that are not in AppDetect's dictionary), nor can it be used for video flows without an SNI field (e.g., QUIC). Last, basic matching approaches are prone to deliberate exploitation – malicious subscribers can access any content (not just video) at no cost[5] to their data plan by setting up a proxy and modifying HTTP or HTTPS headers [14] to appear as an alternate flow type.

Thus, we analyze video flows and use this analysis as motivation for a new algorithm to detect video flows which does not rely on SNIs.

6.1 High Definition (HD) Video Detection without SNIs

Figure 10 depicts the relationship between downlink throughput and average TCP payload length for HTTPS-based YouTube flows longer than 10 seconds. These "long" video flows are triaged into three categories: *unlikely HD Video*, *might-be HD Video*, and *likely HD Video*.

The *unlikely* flows consist of small packets (an average packet payload length less than 450 byte)s, and small throughputs (less than 128 Kbps), making it unlikely they carry video content.

The *HD Video* flows consist of large packets with high throughput (more than 300 Kbps). 20% of such flows even have throughputs higher than 1.5 Mbps. It is likely these flows carry video content.

The *might-be* flows contains two types of flows: i) video flows experience low throughput (less than 300 Kbps); and ii) flows with small packets, which may not be video (e.g., commercial banners and preview thumbnail images). Because HTTPS flows are encrypted, AppDetect cannot determine the exact content even using commercial deep-packet inspection engines. However, considering the high capacity of LTE networks, the radio link may not be the bottleneck for the *might-be* flows, anyway, so it is reasonable for middle-boxes not to rate limit them.

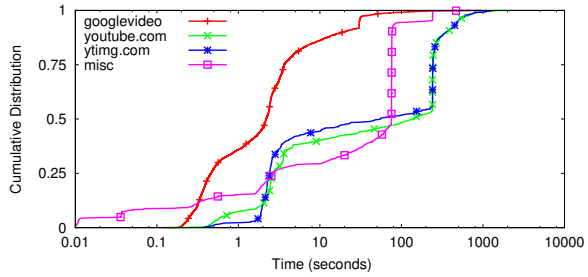
Based on the above observations, we propose a video detection algorithm, shown in Algorithm 1, based on Transport Layer payload length (l) and flow rate (r). The reason to use Transport Layer payload length is that packets without Transport Layer loads do not carry any application data (e.g. TCP SYN/SYN-ACK and ACKs), instead serving as control packets.

Note, Line 19 calculates the payload length using a simple average. It could be replaced with an exponentially weighted moving average [20] to reduce the effects packets that are markedly smaller or larger than the average. However, our preliminary studies show the

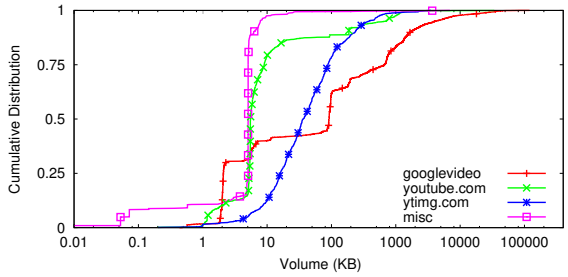
[3] `https://en.wikipedia.org/wiki/QUIC`

[4] `https://isc.sans.edu/forums/diary/Port+0+DDOS/17081/`

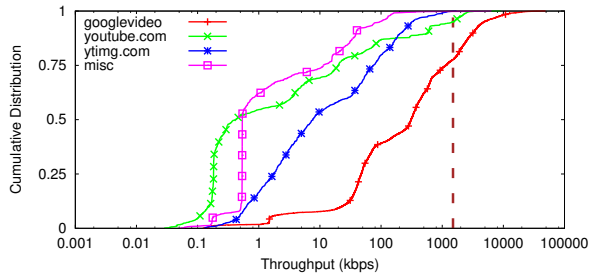
[5] Some U.S. wireless providers do not count video traffic towards a subscribers data plan.



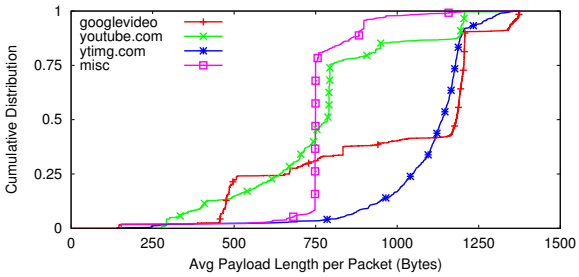
(a) Distribution of Duration of HTTPS-Based YouTube Video Flows



(b) Distribution of Downlink Volume of HTTPS-Based YouTube Video Flows

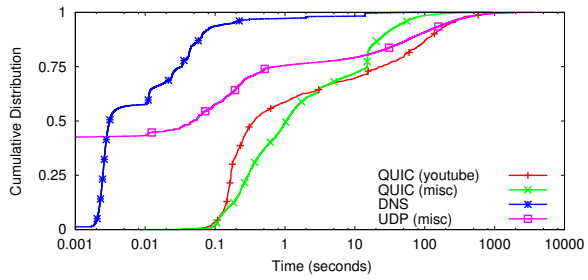


(c) Distribution of Downlink Throughput of HTTPS-Based YouTube Video Flows

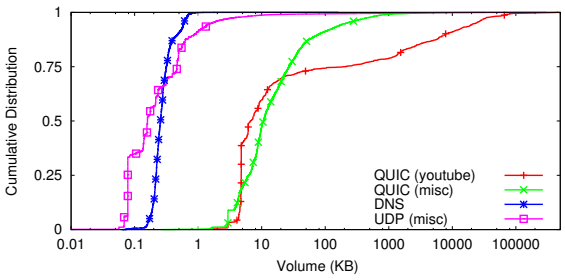


(d) Distribution of Downlink Average Packet Length of HTTPS-Based YouTube Video Flows

Figure 8: HTTPS-Based YouTube Video Flow Analysis



(a) Distribution of Duration of UDP Flows



(b) Distribution of Volume (uplink and downlink) of UDP Flows

Figure 9: UDP Flow Analysis

Table 2: Symbols in Algorithm 1

Thresholds	Description	Defaults
T_v	threshold for video segments	60 KB
R_v	rate threshold for video	300 Kbps
R_a	rate threshold for non-video	128 Kbps
L_v	pkt length threshold for video	900 B
L_a	pkt length threshold for non-video	450 B

simple average of payload length works quite well.

Table 2 summarizes the thresholds used in Algorithm 1 along with their default values.

Video Segments Threshold Video clips contain at least one video segment [22] and their volume is not small. Thus, T_v is used to filter out flows which are too small to carry a video segment, and is set to the minimum segment size for videos stored by a content provider. For our study, the default value for T_v is chosen to be 60 KBytes, close to the size of 2-second long video segment of a YouTube clip [9, 19]. Another reason to introduce T_v is to differentiate the initial buffering phase of a video streaming session. A video player is able to start the downloaded video only after its initial play-out buffer is filled. Thus, reducing initial buffering time with TCP acceleration [28] can improve the QoS of video streaming (i.e., providing a short "time to first picture"). However, the playout rate of a video stream is limited by its codec [6, 22], thus providing a reasonable throughput limit after the initial buffering phase. Thus, the QoS requirement for the initial buffer phase is different than the QoS requirement during steady-

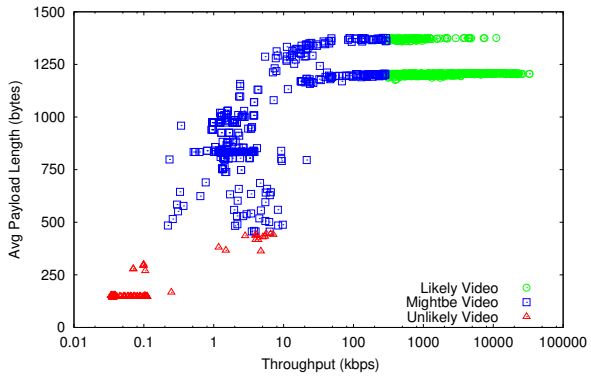


Figure 10: Throughput and Average Payload Length for HTTPS-Based YouTube Flows Longer than 10 Seconds

Algorithm 1 Video Flow Detection

variables
  c: number of pkts with transport layer payload.
  l: transport layer payload length of pkt p.
  t_e: elapsed time from 1st pkt.
  $\bar{L}$: average length of transport layer payload.
  S: cumulative length of transport layer payload.
  R: application layer throughput.
end variables
for each down link packet p in flow id **do**
  if $is\_from\_video\_server(p) = false$ **then**
    return $false$
  end if
  $l \leftarrow payload\_len(p)$
  if $l = 0$ **then** ▷ no application level data
    return $current\_type(id)$
  end if
  $c \leftarrow c + 1$
  $S \leftarrow S + l$
  $\bar{L} \leftarrow S/c$ ▷ avg payload length
  $R \leftarrow S/t_e$
  if $(S \geq T_v) \cap (R \geq R_v) \cap (\bar{L} \geq L_v)$ **then**
    return $true$ ▷ likely HD video flow
  else if $(S < T_v) \cap (R < R_a) \cap (\bar{L} < L_a)$ **then**
    return $false$ ▷ unlikely HD video flow
  end if
  return $mightbe$ ▷ might-be HD video flow
end for

state playout. Therefore, T_v can help middle-boxes to differentiate the initial buffering phase from the steady streaming phase.

Rate Thresholds After the initial buffering phase, the throughput of video flows lies within a certain, well-known range [6, 22]. Therefore, Algorithm 1 uses a video rate threshold (R_v) to detect HD video flows – if a flow's rate is higher than R_v, it is likely to be a high quality video. In this study, the default value of R_v is 300 Kbps, which is the encoded rate of a 240p video on YouTube. R_a is used to detect low rate media (e.g., voice) traffic. Due to the popularity of social networks, some apps convert short video clips into animated GIFs for distribution to friends. These low quality videos often have low bit rates, too. In this study, we set R_a's default value to 128 Kbps, which is the highest audio rate supported by YouTube. Note, voice traffic should be prioritized because voice is sensitive to delay as well as jitter [6, 20].

Packet Payload Length Thresholds Because of the size of video content, the majority of packets in a video flow have lengths close to the MTU settings [20]. Algorithm 1 uses two thresholds to help differentiate video traffic: *i*) if a flow's average Transport layer payload length is greater than L_v, the flow might be a video flow; *ii*) if a flow's average payload length is smaller than L_a, the flow may not be a video flow. In this study, L_v and L_a are 900 and 450 bytes, respectively ($\frac{2}{3}$ and $\frac{1}{3}$ of the default MTU size without IP and Transport layer headers).

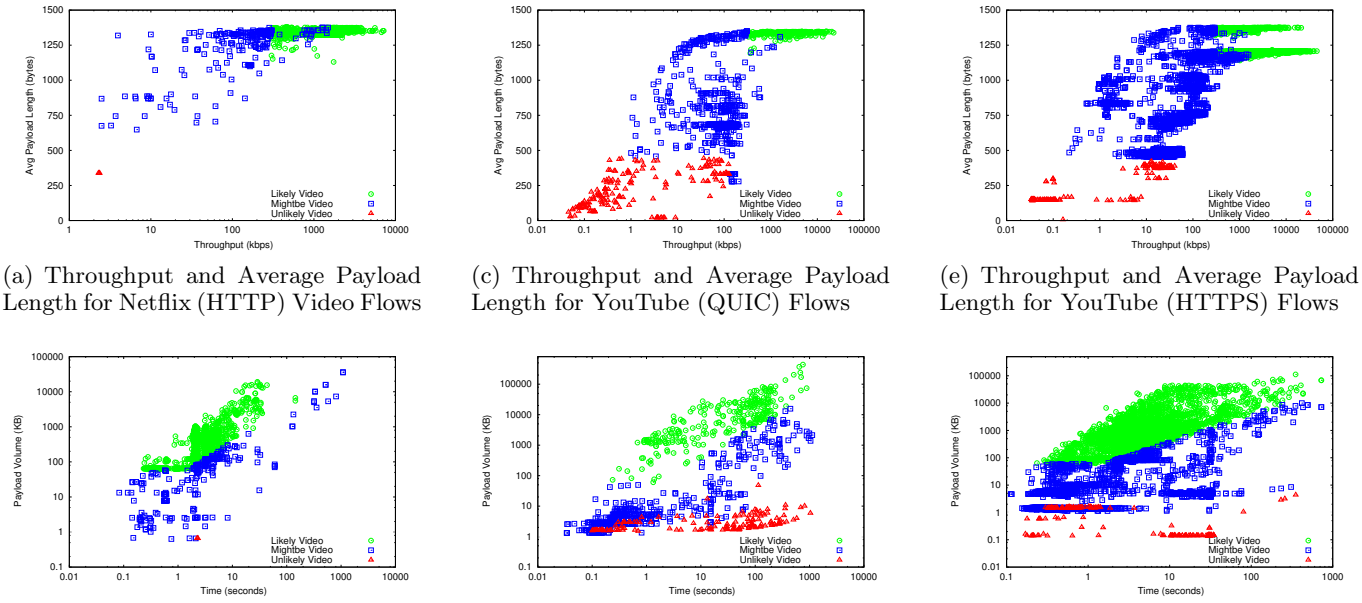
6.2 Results

Table 3 and Figure 11 summarize the results of Algorithm 1 on Netflix (HTTP), YouTube (QUIC), and YouTube (HTTPS) flows. Note, there are two possible misclassification cases: *i*) some high volume, long duration flows might be misclassified as "might-be" video because their throughputs are less than $R_v = 300Kbps$, e.g., the blue squares in the top right area of Figure 11(b). *ii*) Some flows with large packets and high data rates might be misclassified as "might-be" video because their cumulative payload sizes are smaller than 2 second video segments ($T_v = 60KB$). Case *i* cold be caused by the throughput calculation – our offline tests reveal some gaps between different video segments, and the measured throughput might be lower than actual video encoding rate. One reason to explain case *ii* is that subscribers could abandon watching a video at the beginning of streaming – our offline tests reveal 7% of the YouTube HTTPS flows end with TCP RSTs from clients, an indicator the subscribers have prematurely terminated the video.

Table 3: Video Detection Results for Algorithm 1

Categories	Netflix (HTTP)		YouTube (QUIC)		YouTube (HTTPS)	
	#	%	#	%	#	%
Likely	2842	90.3	272	19.4	5625	48.1
Might-be	302	9.6	973	67.0	5766	49.4
Unlikely	4	0.0	190	13.5	291	2.5
Total	3148	100.0	1399	100.0	11682	100.0

7. FUTURE WORK



(a) Throughput and Average Payload Length for Netflix (HTTP) Video Flows

(c) Throughput and Average Payload Length for YouTube (QUIC) Flows

(e) Throughput and Average Payload Length for YouTube (HTTPS) Flows

(b) Duration and Downlink Volume for Netflix (HTTP) Video Flows

(d) Duration and Downlink Volume for YouTube (QUIC) Flows

(f) Duration and Downlink Volume for for YouTube (HTTPS) Flows

Figure 11: Algorithm Results for Netflix (HTTP), YouTube (QUIC) and YouTube (HTTPS) Flows

The future plan is to remove reverse DNS checks (Lines 10 to 12) from Algorithm 1. This may cause some non-video traffic (e.g., such as large file downloading) to be classified (and, hence, treated) as video. However, from the perspective of radio link utilization, such misclassification might not be harmful. File transfer applications are not delay sensitive – subscribers sometimes run large file downloads overnight. So, treating large download flows during congestion may not degrade user QoE, while saving radio link utilization and, hence, supporting more concurrent users without additional hardware investment.

Future plans also include enhancements to AppDetect with performance analysis abilities – e.g., adding analysis and statistics for TCP retransmission, segment loss, and duplicate ACKs. Due to the volatile nature of radio conditions, it is often difficult for field engineers to reproduce and trouble-shoot performance issues reported by customers. Thus, AppDetect enhanced with performance analysis abilities may be able to detect performance issues before customers even file complaints. Besides, more analytic services for AppDetect will be developed using data mining technologies and the data collected by the AppDetect. These analytic services can provide intelligence for network planning, marketing, and even QoS enhancement strategies.

8. CONCLUSIONS

This paper presents a novel real-time application detection framework – *AppDetect* – that collects flow statistics and traffic information on middle-boxes (e.g., a router) over mobile networks without leaking customer privacy. Different from other approaches, AppDetect does not rely on capturing, offline processing, or deep packet inspection of application payloads. Moreover, AppDetect can be deployed with any Linux-based inline box or performance enhancing proxy (PEP). The classification results AppDetect produces can be used for many purposes such as dynamic traffic shaping, QoS enhancement, real-time data mining and even long-term radio network planning. Complementing AppDetect, we propose a real-time video detection algorithm that works for encrypted flows, allowing middle-boxes to manage increasingly important video traffic.

In addition to our methodological contributions, we also investigate the cross-layer traffic statistics and show the analysis results from a large carrier's mobile network serving the South Central geographic area of the United States (e.g., Oklahoma). Our traffic analysis includes IPv4/6, HTTP/S and newly emerging QUIC. Based on our analysis, IPv6 and HTTPS are becoming increasingly prevalent. QUIC composes over 60% of UDP traffic and warrants attention since it greatly increases UDP volumes. From the application points of view, Google, Facebook, and Apple are the top 3 content providers while YouTube alone accounts for over 20% of the total traffic. In addition to providing a detailed snapshot of current (late 2016) commercial mobile network use, the results from our traffic analysis should be useful for industrial or academic researchers doing simulation, emulation and modeling of traffic for LTE networks.

9. REFERENCES

[1] V. K. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z.-L. Zhang. Unreeling Netflix: Understanding and Improving Multi-CDN Movie Delivery. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, March 2012.

[2] N. Brownlee and kc Claffy. Understanding Internet Traffic Streams: Dragonflies and Tortoises. *Communications Magazine, IEEE*, 40(10):110–117, 2002.

[3] J. But, U. Keller, D. Kennedy, and G. Armitage. Passive TCP Stream Estimation of RTT and Jitter Parameters. In *Proceedings of the 30th IEEE Conference on Local Computer Networks(LCN)*, Sydney, Australia, November 2005.

[4] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson. BBR: Congestion-Based Congestion Control. *ACM Queue*, 14, September-Octobor, 2016.

[5] Cisco System Inc. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update. Technical report, Cisco System Inc., February 2016.

[6] M. Claypool, R. Kinicki, and C. Wills. Treatment-Based Traffic Signatures. In *Proceeding of IMRG (IETF Internet Measurement Research Group) Workshop on Application Classification and Identification (WACI)*, Cambridge, MA, October 2007.

[7] N. Dukkipati, M. Mathis, Y. Cheng, and M. Ghobadi. Proportional Rate Reduction for TCP. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, 2011.

[8] T. Flach, P. Papageorge, A. Terzis, L. Pedrosa, Y. Cheng, T. Karim, E. Katz-Bassett, and R. Govindan. An Internet-Wide Analysis of Traffic Policing. In *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*, pages 468–482, August 2016.

[9] Google Inc. Live encoder settings, bitrates, and resolutions: YouTube Help, August 2016.

[10] A. Gupta and R. K. Jha. A survey of 5G network: architecture and emerging technologies. *IEEE access*, 3, 2015.

[11] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. A Close Examination of Performance and Power Characteristics of 4G LTE Networks. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, 2012.

[12] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck. An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance. In *ACM SIGCOMM Computer Communication Review*, SIGCOMM '13, 2013.

[13] J. Iyengar and I. Swett. QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2, June 2015.

[14] A. M. Kakhki, F. Li, D. Choffnes, A. Mislove, and E. Katz-Bassett. BingeOn Under the Microscope: Understanding T-Mobile's Zero-Rating Implementation. In *Proceedings of Internet-QoE Workshop*, August 2016.

[15] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel Traffic Classification in the Dark. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 229–240, Philadelphia, PA, August 2005.

[16] J. Khalife, A. Hajjard, and J. Diaz-Verdejo. Performance of OpenDPI in Identifying Sampled Network Traffic. *Journal of Networks*, 8(1):71–81, January 2013.

[17] M. Kühlewind, S. Neuner, and B. Trammell. On the State of ECN and TCP Options on the Internet. In *International Conference on Passive and Active Network Measurement*, March 2013.

[18] L. Le, J. Aikat, K. Jeffay, and F. D. Smith. Differential Congestion Notification: Taming the Elephants. In *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP)*, pages 118–128, Berlin, Germany, October 2004.

[19] S. Lederer. Optimal Adaptive Streaming Formats MPEG-DASH & HLS Segment, April 2015.

[20] F. Li, M. Claypool, and R. Kinicki. Treatment-based traffic classification for residential wireless networks. In *2015 International Conference on Computing, Networking and Communications (ICNC)*, February 2015.

[21] F. Li, A. M. Kakhki, D. Choffnes, P. Gill, and A. Mislove. Classifiers Unclassified: An Efficient Approach to Revealing IP Traffic Classification Rules. In *Proceedings of the 2016 ACM on Internet Measurement Conference*, IMC '16, 2016.

[22] J. Martin, Y. Fu, N. Wourms, and T. Shaw. Characterizing Netflix bandwidth consumption. In *2013 IEEE 10th Consumer Communications and Networking Conference (CCNC)*, pages 230–235, December 2013.

[23] A. Molavi Kakhki, A. Razaghpanah, A. Li, H. Koo, R. Golani, D. Choffnes, P. Gill, and A. Mislove. Identifying Traffic Differentiation in Mobile Networks. In *Proceedings of the 2015 ACM*

Conference on Internet Measurement Conference, pages 239–251, October 2015.

[24] D. Moore, K. Keys, R. Koga, E. Lagache, and K. Claffy. The CoralReef Software Suite as a Tool for System and Network Administrators. In *Proceedings of the 15th USENIX Conference on System Administration (LISA)*, pages 133–144, San Diego, CA, 2001.

[25] C. Palazzi, G. Pau, M. Roccetti, S. Ferretti, and M. Gerla. Wireless Home Entertainment Center: Reducing Last Hop Delays for Real-Time Applications. In *Proceedings of the ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE)*, page 67, Hollywood, CA, 2006.

[26] J. Roskind. QUIC: Quick UDP Internet Connections Multiplexed Stream Transport over UDP, November 2013.

[27] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. Class-of-Service Mapping for QoS: a Statistical Signature-based Approach to IP Traffic Classification. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, Taormina, Sicily, Italy, October 2004.

[28] J. Snellman. Mobile TCP Optimization: Lessons Learned in Production. Technical report, Telco. Networks, August 2015.

[29] H. Wu. *ARMOR - Adjusting Repair and Media Scaling with Operations Research for Streaming Video*. PhD thesis, Worcester Polytechnic Institute, May 2006.

[30] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman. Identifying Diverse Usage Behaviors of Smartphone apps. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference (IMC'11)*, November 2011.

[31] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the Characteristics and Origins of Internet Flow Rates. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 309–322, Pittsburgh, PA, August 2002.