MoVIE: A Measurement Tool for Mobile Video Streaming on Smartphones

Sina Keshvadi*
University of Calgary
Calgary, Alberta, Canada
sina.keshvadi1@ucalgary.ca

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

ABSTRACT

Mobile video streaming is becoming increasingly popular. In this paper, we describe the design and implementation of a cross-platform measurement tool called MoVIE (Mobile Video Information Extraction) for video streaming on mobile devices. MoVIE is a client-side traffic analyzer that studies smartphone video streaming from different viewpoints. It collects information about network-level packet traffic, transport-layer flows, and application-level video player activities. Then it identifies relationships within the collected data to make mobile video streaming activities transparent. MoVIE is an open-source tool with a graphical user interface. In addition to network traffic measurement, MoVIE supports objective Quality of Experience (QoE) evaluation of video streaming. These features make MoVIE a powerful tool for network traffic measurement, multimedia streaming studies, and privacy analysis. We illustrate MoVIE's capabilities with a small case study of streaming 360° videos.

CCS CONCEPTS

• **Networks → Network measurement**; • **Information systems → Multimedia streaming**.

KEYWORDS

Network Traffic Measurement, Video Streaming, Smartphone

ACM Reference Format:
Sina Keshvadi and Carey Williamson. 2020. MoVIE: A Measurement Tool for Mobile Video Streaming on Smartphones. In *Proceedings of the 2020 ACM/SPEC International Conference on Performance Engineering (ICPE '20), April 20–24, 2020, Edmonton, AB, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3358960.3375796

1 INTRODUCTION

Over the past decade, smartphones have become an integral part of our lives. The emerging mobile web supported by WiFi and 4G/5G cellular networks allows anytime and anywhere Internet access from hand-held devices such as smartphones. In the first quarter of 2019, the traffic from hand-held devices (excluding tablets) represented 48.7% of global web traffic [24].

Among all Internet traffic consumption, video streaming has the biggest share [2]. Everything from entertainment, sports, businesses, educational institutions, and governments are increasingly using video streaming services. During recent years, video-on-demand services such as Hulu, Netflix, Vimeo, and YouTube have experienced explosive growth in terms of the number of users and videos. Indeed, video delivery is shifting from traditional TV broadcasting to online video streaming, and Internet traffic is dominated by such videos. A recent report from Cisco [2] predicts that global mobile traffic will increase 7-fold by 2022, and mobile video streaming traffic will grow 11-fold, reaching nearly 79% of the world's total mobile traffic. This growth in mobile traffic will be a heavy burden on mobile operators.

Another ongoing trend that is feeding this tremendous growth is Head-Mounted Displays (HMDs). Devices such as Oculus Rift, HTC Vive, and Google Cardboard have become widely available and affordable for the public. These devices enable viewers to watch immersive 360° videos with even higher bandwidth demands. Major multimedia streaming providers such as YouTube and Facebook now support uploading and viewing these 360° videos, which users can watch by connecting their smartphones to the HMDs. However, this technology demands huge bandwidth for streaming and also the part of the video that is not in the user's viewport is considered as wasted bandwidth [27].

By increasing popularity in video streaming, this market has welcomed many different providers. In this competitive circumstance, the video providers collect information about the user and video streaming to improve their services. Besides, many of these providers cover part of their costs through advertising and data analytics. This questions the user's privacy. Recently users are more concerned about their privacy and security when they interact with Internet services [26].

In this context, the characterization of video streaming on mobile devices can provide a comprehensive understanding of video streaming behaviours. Network measurement tools on their own lack visibility into the video player activities, and existing video quality measurement tools do not provide adequate insight into the network traffic and privacy issues generated by mobile devices.

To tackle these issues, we have designed and implemented a tool called MoVIE, which uses multiple observational viewpoints [1] to extract information about mobile video streaming. It studies video streaming from device, network protocol, and video player viewpoints. Our tool has a graphical user interface that allows researchers to have a multi-layer view of video streaming.

To illustrate the MoVIE functionality, we use it to study 360° video streaming on smartphones. To the best of our knowledge, there are no tools that investigate video streaming activities on

ICPE '20, April 20–24, 2020, Edmonton, AB, Canada
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-6991-6/20/04...$15.00
https://doi.org/10.1145/3358960.3375796

smartphones using multiple coordinated viewpoints. Overall, the main contributions of this paper are summarized as follows:

- We design and implement MoVIE, a video measurement tool that provides multiple simultaneous views of streaming traffic.
- We propose a new mechanism to identify relationships between packets, flows, and video player activities from generated traffic.
- We analyze the streaming of 360° videos from a mobile web player viewpoint. MoVIE provides objective Quality of Experience (QoE) metrics of video streaming. In our case study, we evaluate two popular 360° video providers from different viewpoints.
- MoVIE is open-source, portable to any operating system, and easy to use and customize. We provide references to the MoVIE website and source code.

The rest of this paper is organized as follows. Section 2 presents an overview of the tool, followed by the detailed design and implementation in Section 3. Section 4 presents an initial case study, in which we use MoVIE to analyze an Android device when it plays 360° videos from two different video streaming providers. Section 5 presents related work. Section 6 concludes the paper.

2 TOOL OVERVIEW

This section presents an overview of the MoVIE tool, which intercepts and analyzes all network traffic into and out of a smartphone during video streaming.

We aim to develop a user-friendly and easy-to-customize tool to explore video streaming traffic, measure objective video streaming QoE metrics, and provide visibility into streaming traffic. The MoVIE tool is designed with the following features:

- **Visibility**: MoVIE captures all incoming/outgoing Internet traffic from the smartphone during the video streaming. It records all video streaming activities such as the transmitted TCP/UDP packets, HTTP/HTTPS requests and responses, streaming status, and video buffering events. MoVIE provides a multi-level view of video streaming network traffic, from the network packet level to an application-level view.
- **QoE Measurement**: Video streaming users expect a Quality of Experience (QoE) that is not always well-reflected by traditional network-layer QoS metrics such as packet loss, delay, and jitter. QoE metrics are classified into objective and subjective metrics [21]. Objective QoE metrics, such as playback delay, video quality switches, and the number of stalls, are metrics that can be quantified with a measurement tool. MoVIE provides the objective QoE metrics for video streaming. Subjective metrics, such as Mean Opinion Score, must be collected directly from users.
- **Privacy View**: Smartphones involve sensitive information for personal conversations, bank transactions, online purchasing, tax payment, and so on. In addition, smartphones exploit sensors like GPS, camera, and microphone to provide better services, but increase the potential risk of privacy violation. Recent studies show that people are concerned about the privacy of themselves and their children. MoVIE provides a full view of the generated connections, and it shows the tracking and advertisement flows. It provides a detailed report of what data is collected and where it is sent during the interactions with streaming websites.
- **Traffic Mapping**: In this paper, we propose a novel and accurate mechanism to relate all transmitted packets to network flows and video streaming status. In this way, MoVIE can report network statistics such as bitrate, packet rate, average packet size, and connection type for every single flow, with all flows related to application-level video player activities.
- **Comparability**: MoVIE can process up to two video streaming traces simultaneously, which is convenient for comparing video providers, video codecs, and video players. This feature makes MoVIE a powerful tool to compare video streaming from different devices, operating systems, providers, and codecs.
- **Customizability**: MoVIE is an open-source tool that visualizes user interactions and enables researchers to view network flows, captured packets, multimedia events, general statistics, graphs, and so on. MoVIE is portable and can run on all desktop operating systems. The measurement report is in HTML5 format, which is supported by any modern browser and OS. To make MoVIE an easy-to-customize tool, we: i) publish the source code on GitHub; ii) design a website to explain the tool and illustrate the experimental setting; iii) and provide several use cases with reports and raw captured data.

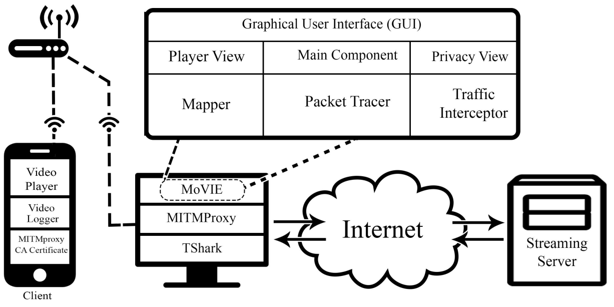


Figure 1: Structural overview of the MoVIE tool for mobile video streaming analysis

3 TOOL DESIGN AND IMPLEMENTATION

This section discusses the design and implementation of MoVIE, which captures all network traffic, request/response flows, and video player events. Based on the captured data, MoVIE then provides a graphical view of video streaming traffic.

Figure 1 illustrates the architecture of MoVIE, which consists of seven components: *Traffic Interceptor*, *Packet Tracer*, *Player View*, *Privacy View*, *Mapper*, *Main*, and *Graphical User Interface*. We now discuss each component in more detail.

3.1 Traffic Interceptor

MoVIE characterizes all Internet traffic generated by the mobile device during video streaming. It starts with an application-layer view of the activities generated by the Web browser. To achieve this, we use MITMproxy [3] to transparently intercept all HTTP/HTTPS traffic between the mobile device and the Internet. MITMproxy sits between the two parties, and is able to intercept all transmitted traffic to provide a coarse-grained view of all network activities involving the smartphone. Since most mobile apps use TLS as the default protocol for secure communication [17], we need to decrypt this traffic. MITMproxy can decrypt the TLS traffic if the mobile device is configured to trust MITMproxy. Specifically, this requires the MITMproxy CA certificate to be installed on the mobile device. The *Traffic Interceptor* collects information such as URL, host name, GET/POST requests, connection types, duration, request/response timestamps, plus sender and receiver IP addresses and port numbers, and sends them to the *Main* component. The example below shows several lines of a flow response in JSON format:

```
{"Method":"GET",  "Status_Code":"200",
"Content_Type":"video/webm", "Content_Length": "1912",
"Client_IP":"172.17.20.22", "Client_Port":"57924",
"Server_IP":"157.240.3.23", "Server_Port":"443",
"Host":"video-sea1-1.xx.fbcdn.net", "User-Agent":
"Mozilla/5.0 (X11; Linux x86-64) AppleWebKit/537.36
(KHTML, like Gecko)Chrome/71.0.3578.98 Safari/537.36",
"TLS-Established":"True",
"ClientInitiatedTime":"1552886540.546299"}
```

3.2 Packet Tracer

We use TShark, a terminal-oriented version of Wireshark[1], to capture full packet traces of Internet traffic generated by mobile video streaming. TShark is a widely-used protocol analyzer, providing a network-level view of traffic, similar to Wireshark. The raw packet trace data can be saved in JSON, XML, and PCAP file formats.

The output from the *Packet Tracer* is sent to the *Main* component for analysis. The example below shows some TCP packet information in JSON format:

```
    {"frame.time":"Jan 24, 2019 06:26:31.514545631 MST",
     "frame.time_absolute":"1548336391.514545631",
     "frame.time_relative":"5.724449491",
     "frame.number":"41", "frame.len":"74",
     "frame.protocols":"eth:ethertype:ip:tcp",
     "ip.version":"4", "ip.proto":"6",
     "tcp.stream":"1", "tcp.window_size":"29200"}
```

3.3 Player View

MoVIE uses video player activities and events to measure video quality metrics. It also exploits this information to determine the relationships between video player events and network traffic generated by the mobile device.

We developed an Android app to collect and parse video properties [10], video player activities, and player events from video player. The video log contains title, URL, audio/video codecs, duration, resolution, play starting time, video buffering events, video resolution changes, and so on.

The output of this component is sent to the *Main* component for analysis. The example below shows some video player log entries in JSON format:

```
    {"time":0,
    "key":"origin_url","value":"https://www.youtube.com/"}
    {"time": 0.010999999940395355,
    "key": "frame_title", "value": "YouTube"}
    {"time": 0.1459999978542328,
    "key": "url", "value":
     "blob:https://www.youtube.com/f169cece-5961-401c-a2ae"}
    {"time": 3.623999997973442,
    "key": "pipeline_state", "value": "kStarting"}
    {"time": 990.527000002563,
    "key": "found_audio_stream", "value": true}
    {"time": 990.5300000011921,
    "key": "audio_codec_name", "value": "opus"}
```

[1]https://www.wireshark.org

3.4 Privacy View

This component investigates the captured flows to detect potential data leaks. It receives flows from the Traffic Interceptor component and uses EasyList[2] to detect advertisement and tracking flows. EasyList has a list of all international tracking and advertisement URLs, and is the primary filtering list used by ad blockers. The list is created and updated weekly by the ad-block community[3] and supports many different geographical regions. This component extracts a list of potential tracking and advertising flows from the set of all flows observed.

3.5 Mapper

MoVIE exploits multiple observational viewpoints [1] for its analysis of network traffic, flows, and video player activities. Mapping the network packets captured by Wireshark/TShark to the flows collected from MITMproxy, and relating this information to video player activities, is an essential phase of MoVIE.

We need to identify the origins of each flow, for given video streaming activities, and which packets correspond to those flows. Prior work [18] has shown that previous mapping approaches in many situations fail to map the passive network traffic to the corresponding mobile apps. In the following, we propose our own novel mapping mechanism. In addition to mapping packets to the corresponding apps, it maps the captured network traffic to each responsible flow and video player activity.

In the following five subsections, we describe the step-by-step details of our solution for this problem.

3.5.1 *Flows:*
We use the header field of each generated flow to group Internet connections. The `User-Agent` field identifies the software component that originates the flow request. For example, web browsers use the following User-Agent template [4, 19]:

```
[Name]/[version]   ([system   and   browser   information])
[platform] ([platform details]) [extensions]
```

Using this format, an Android Mobile `User-Agent` generated from a Samsung Galaxy S9 smartphone might appear as follows:

```
Mozilla/5.0 (Linux; Android 8.0.0; SM-G960F Build/R16NW)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.84
Mobile Safari/537.36
```

By exploiting this information, we can separate Web browser traffic from other generated flows. For multimedia traffic that includes video and audio streams, iOS devices use `Apple Core Media/[v]` and Android devices use `Stagefright` as the `User-Agent` field. In addition, we use the `Host` and `ResolvedAddress` fields of flows to identify the host name and IP address of the destination server. In the next step, we group related flows into sessions.

3.5.2 *Sessions:*
A session might involve multiple flows for initiating, maintaining, and terminating that session. These flows have several shared attributes, such as times at which the client and the server initiated their (application-layer) communication. In addition, each (transport-layer) flow in a session has its own timestamps for Client and Server TCP/TLS handshake, as well as request and response timestamps.

[2]https://easylist.to/easylist/easylist.txt
[3]http://adblock.mozdev.org

We group all flows into sessions based on their common fields. Each session contains a list of flows, host name, user agent, source and destination IP addresses and port numbers, and client/server connection establishment time. In the next step, we analyze the captured packets from Wireshark/TShark.

3.5.3 TCP/UDP Streams: All packets in a Wireshark trace are automatically grouped in TCP/UDP streams. A stream is a sequence of packets transmitted between two transport-level endpoints in one session. As a result, the packets of a session have the same value for the `tcp.stream` or `udp.stream` field.

To find the destination host of each stream, we use the `Host` field. For HTTP connections, one packet in the stream contains the `Host` and `User-Agent` fields from the HTTP request header. The `Host` field indicates the name of the stream destination. For HTTPS connections, the destination host name is located in the `ssl.handshake.extensions_server_name` field of the SSL layer in one of the packets of that stream.

We group all packets according to their stream index. Each TCP stream has its own packet list, destination host name, user agent, source and destination IP addresses and port numbers, and packet timestamps. We group UDP streams in a similar way. For UDP streams that include DNS packets, we use the `dns.qry.name` field to find the host name.

3.5.4 Mapping Packets to Flows: For each session identified by MITMproxy, we observe two separate streams with the same host name, source/destination IPs, and ports in the Wireshark trace. The reason for this is that Wireshark and MITMproxy run on the PC, and not the smartphone itself. As a result, each TCP session appears twice: once from the mobile device to MITMproxy, and again from MITMproxy to the destination server. Fortunately, the timestamps of the flows differ slightly, making it easy to distinguish them.

We assign packets in each stream to the flows in the corresponding session. The timestamps for requests and responses of each flow are used to find the corresponding packets. This is doable since there is a gap between the timestamps of packets from one flow to the next one. To the best of our knowledge, we are the first to provide an exact mapping of packets to flows. In the next step, we map the video activities to flows.

3.5.5 Mapping Video Activities to Flows: In the final step of the mapping process, we separate audio/video flows from other flows, and use the timestamps of player activities and events to find the corresponding flows. This completes the in-depth look at our *Mapper* component.

3.6 Main Component

The *Main* component is the central part of MoVIE that manages and coordinates all the other components. It receives a flow list from the *Traffic Interceptor*, a packet list from the *Packet Tracer*, and the video streaming activities from the *Player View* component. The *Mapper* component assigns the connections based on all captured information. Then the *Main* component analyzes the data to determine the video streaming metrics. It calculates the video streaming QoE parameters, statistics of packets and flows, categorizes data, and so on. Finally, the *Main* component invokes the *Graphical User Interface* to provide a visual representation of the results.

3.7 Graphical User Interface (GUI)

The MoVIE tool represents video streaming analysis in a graphical view. The multi-level visual representation in the form of tables and graphs helps researchers gain better insight into the video streaming behaviour. This GUI represents relationships and interactions between the video player, the network, and the video provider. In addition, network statistical reports such as throughput, flow sizes, packet counts, and many more are represented in tables and graphs that make them easy to explore.

The output of MoVIE is generated in HTML5 format which offers several advantages. First, it is easy to use, and easy to export the data for other purposes. Second, it is portable and only needs a browser to explore the outputs. Last but not least, MoVIE is an open-source tool, with source code available for customization. We implemented a library to export data in HTML5 format to facilitate report customization.

4 CASE STUDY

In this section, we illustrate the capabilities of MoVIE by using it to study video streaming on a mobile device. We first describe our experimental setup, and then use our tool to study 360° video streaming from YouTube and Facebook.

4.1 Experimental Setup

We set up an environment that streams video sessions in a mobile Web browser and collects packet traces, flow traces, and video player activities. All video streaming was done on a Nexus 6 smartphone (Quad-Core 2.7 GHz CPU, 3 GB RAM) running Android 7.0, with video streaming performed using Chrome 72.0 as the Web browser. MoVIE ran on a separate PC (Core i7, 8-core, 3.6 GHz CPU, 8 GB RAM) running Ubuntu 18.04 Linux. Both devices used WiFi for communication. The PC was used to capture all incoming and outgoing network flows and packet traffic from the mobile device.

To intercept the network traffic, we used MITMproxy 4.0 on the PC to collect network flows. TShark 6.2.2 was used on the PC to capture full packet traces of the Internet activities generated by the smartphone. Our Android app was installed on the mobile device, and used to parse the player log and send a JSON version of video player activities to the MoVIE.

We prepared the mobile device by performing a factory reset to ensure that other software or previous experiments do not impact our experiments. In addition, we updated the OS and pre-installed apps to the latest versions. We cleared the browser cache and history before each video streaming session. Then we installed the MITMproxy CA Certificate on the device. During each test, we configured the Chrome browser on the smartphone to send all network traffic through the MITMproxy to collect HTTP/HTTPS flows. At the same time, TShark was running on the PC to capture network traffic at the packet level.

We streamed a single 360° video called “Lion Cub” from the National Geographic Channel, using two different popular 360° video streaming platforms: YouTube and Facebook. Evaluating the same video content with two different streaming platforms helps in understanding the differences in their services.

Once the network was set up, we started the data collection tools on the PC and the video streaming on the mobile device. At the end of video streaming, our video logger application sent the video player activities to MoVIE for further processing. After all network traffic was complete, we used MoVIE to analyze the data. All captured network data, video streaming log, and results are available from our project Web site [10].

4.2 Measurement Results

MoVIE's report is divided into four parts: packet level, flow level, video player level, and comparison. The first part provides a detailed packet-level report of network traffic generated by the mobile device. The second part presents a flow-level and privacy analysis of the network traffic. The third part reports video player activities and QoE metrics. The final part is for comparing two different video streaming sessions. This could be used to compare streaming from video providers, video services, video codecs, video encoders, video players, and devices.

Here we illustrate each part in detail. Later, we compare 360° video streaming from YouTube and Facebook.

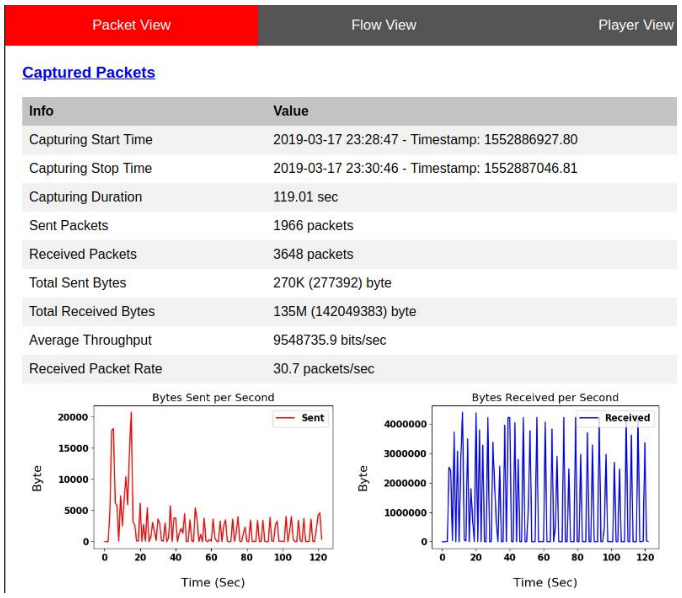
4.2.1 Packet-Level View: The first part of the tool provides a packet-level report of captured network traffic. As shown in Figure 2, this report includes a tabular view of network traffic statistics, and graphs[1] of sent/received network traffic. In addition, the whole packet trace is available in sortable/searchable form for further interactive exploration. The format includes detailed information such as timestamp, source/destination IP, ports, packet length, type, and other header fields for each individual packet.

This part of the tool is useful for studying video streaming behaviour. For instance, the graph of received bytes in Figure 2 indicates that this video streaming uses an in-memory buffering method. It starts by preloading a certain amount of video and audio in advance, to prevent video stalls.

4.2.2 Flow-level View: This section presents a flow-level study of the generated HTTP/HTTPS flows. First, it reports the statistics about the type and the number of generated flows. Then it illustrates them in graphs. For example, Figure 3 shows the distribution of flow types and sizes, as well as a time-series graph of the flows generated per second during the streaming. This graph shows (as expected) a large number of flows generated at the start of the experiment. This is because the smartphone sends an HTTP request to fetch a Web page, and the HTTP response includes links to several resources such as HTML, scripts, style sheets, images, and others, that must be requested to complete the page loading.

As we expected, the video flows contributed most of the sent/received data size. In addition, the number of video flows are nearly four times more than the number of audio flows. This could be because YouTube uses more video flows to cover a 360° sphere. In another experiment[2], we found that the number of audio and video flows are equal in streaming regular videos from YouTube.

In addition, detailed flow information is listed in a sortable and searchable table. For every single flow, it shows the URL, method (GET/POST/other), status code, content length, content type, source and destination IP addresses, ports, request/response times, and



Captured Packets

Info	Value
Capturing Start Time	2019-03-17 23:28:47 - Timestamp: 1552886927.80
Capturing Stop Time	2019-03-17 23:30:46 - Timestamp: 1552887046.81
Capturing Duration	119.01 sec
Sent Packets	1966 packets
Received Packets	3648 packets
Total Sent Bytes	270K (277392) byte
Total Received Bytes	135M (142049383) byte
Average Throughput	9548735.9 bits/sec
Received Packet Rate	30.7 packets/sec

Figure 2: Packet-level view of YouTube streaming in MoVIE

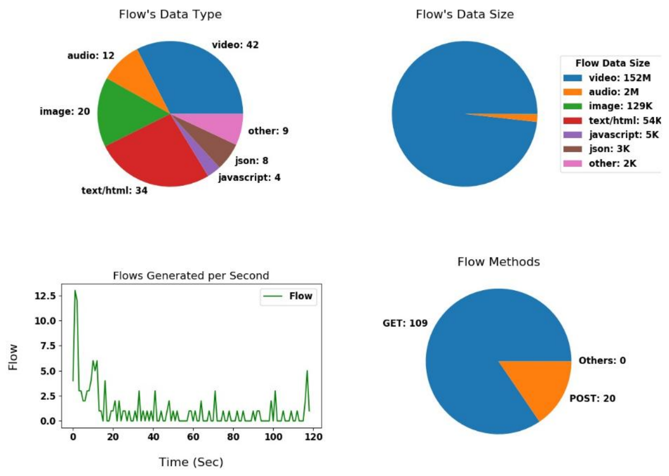


Figure 3: Flow-level view of MoVIE: the pie charts show types and sizes of flows; the graph shows flow arrivals during the streaming experiment

other fields. The user can also drill down to see the packet list for each particular flow. In the example shown here, we observed that YouTube uses HTTP/1.1 for audio/video streaming and HTTP/2.0 for other content types. The HTTP/2.0 Server Push feature allows a server to proactively send resources to a client before the client requests them.

Finally, MoVIE maps the generated flows to the EasyList entries. For this example, it recognized 7 flows as advertisements or tracking flows. Since MoVIE provides a detailed list of all flows, we are able to investigate the generated connections. Here, all ad flows were related to the Google ad services like *googleads.g.doubleclick.net.*

[1]For space reasons, we show only part of the tool functionality here. For instance, the packet level has eight graphs including sent/received packets/bytes, pdf, and CDF charts. More details are available from our project Web site [10].
[2]The experiment's data is available in the MoVIE's website [10].

4.2.3 *Player-level View:*
In this part of the GUI, MoVIE gives detailed information about the streaming events. First, it provides general info about the video such as video/audio codecs, resolution, and duration. Then it reports the QoE metrics of the streaming that includes startup time, number of quality switches, and number of audio/video rebuffering events. Startup time is the time taken to download enough chunks to begin playback. The number of quality switches is calculated by counting the total number of times that the video resolution is changed. Buffering is used to store video chunks that have been downloaded but not yet played. If the player does not find sufficient new data in the buffer, it causes a pause during the playback.

As seen in Table 1, YouTube uses the *Opus* audio codec and the *VP9* video codec for 360° videos. The playback start time was 1.2 seconds. In this particular experiment, we did not observe any video quality switches or rebuffering events.

In addition, as seen in Table 2, this part has a full detailed report for a comprehensive view of video streaming events. The timeline-based report contains all video player activities from fetching the URL to the end of the video streaming. It reports events for URL retrieval, finding video and audio codecs, determining video and audio decoders, pipelining, video and audio buffering, resolution, playtime, timestamps for resolution changes, and all the other playback activities.

MoVIE provides a detailed view of the Google Chrome video player status during the video streaming. In general, the Google Chrome player works as follows. First, the player gets *frame url* and *frame title*. Then it seeks for the audio stream and extracts its *audio codec name*. Finding the *video stream* and *video codec name* are the next steps. Then it recognizes the audio and video *decoders*. After that, it extracts the configuration such as *samples_per_second*, *bytes_per_frame*, and *codec_delay* for audio file. It does the same for video files and extracts information such as *format*, *resolution*, and *visible rect*. Then it fills the audio and video buffers. Before playing the video, it checks the *for_suspended_start* value of its player. Finally, it starts the *PLAY* event. After playing video, it may experience audio/video re-buffering or changing the video quality. *Pause*, *Stop*, and *Destroyed* events happen when the user pauses, stops, or closes the player.

4.2.4 *Comparison:*
The last part of the tool facilitates comparison of two video streaming experiments. In this study, we compared streaming from YouTube and Facebook. This comparison uses selected results from the foregoing sections.

Table 3 shows that YouTube uses *Opus/VP9* as its audio/video codecs, respectively, while Facebook uses *AAC/H.264*. In this example, YouTube streamed 135 MB of content, while Facebook streamed only 82 MB for the same video. Interestingly, Facebook experienced 10 quality switches, 4 audio rebuffering events, and 3 video rebuffering events, while YouTube had none. The playback start time of Facebook (0.72 s) was faster than YouTube (1.23 s). Facebook used 512 flows, while YouTube used only 129 flows. Among these flows, YouTube requested 42 video and 12 audio flows, while Facebook requested 192 video flows and no audio flow. The main reason for the latter difference is that Facebook uses the FB 360 Encoder[1] to

[1]https://facebook360.fb.com/spatial-workstation

Table 1: Player-level view of MoVIE, with QoE metrics

Player	**Value**
Title	YouTube
Frame URL	https://www.youtube.com/watch?v=sPyAQQklc1s&amp
Streaming URL	blob:https://www.youtube.com/f169cece-5961-401c-a2ae-da5d11322ee4
Audio Codec	opus
Video Codec	vp9
First Resolution	1440x2560
Video Duration	270.581 sec
Playing Duration	114.305362 sec
Pause Time	115.535713 sec
Stop Time	116.505313 sec
QoE Metrics	
Player Start Time	1.230351 sec
Audio Rebuffering	0 times
Video Rebuffering	0 times
Quality Switches	0 times

Table 2: MoVIE reports detailed video player activities of the Google Chrome player

Player Time	**Event**	**Value**
0.000000	origin_url	https://www.youtube.com/
0.000009	frame_url	https://www.youtube.com/watch?v=sPyAQQklc1s
0.000011	frame_title	YouTube
0.000013	surface_layer_mode	kOnDemand
0.000146	url	blob:https://www.youtube.com/f169cece-5961-401c-a2ae-da5d11322ee4
0.000176	info	ChunkDemuxer: buffering by DTS
0.990527	found_audio_stream	True
0.990530	audio_codec_name	opus
1.036293	found_video_stream	True
1.036298	video_codec_name	vp9
1.037603	audio_decoder	FFmpegAudioDecoder
1.037622	info	Selected FFmpegAudioDecoder, config: codec: opus, samples_per_second: 48000, bytes_per_frame: 8, seek_preroll: 80000us, codec_delay: 312
1.037746	video_decoder	VpxVideoDecoder
1.037759	info	Selected VpxVideoDecoder, config: codec: vp9, profile: vp9 profile0, coded size: [2560,1440], rotation: 0°
1.037773	pipeline_state	kPlaying
1.039323	audio_buffering_state	BUFFERING_HAVE_ENOUGH
1.050495	height	1440
1.050495	width	2560
1.060295	video_buffering_state	BUFFERING_HAVE_ENOUGH
1.230131	pipeline_buffering	BUFFERING_HAVE_ENOUGH
1.230131	for_suspended_start	False
1.230351	event	PLAY

Table 3: Comparison of YouTube and Facebook streaming of "Lion Cub" video

Info	**YouTube Traffic**	**Facebook Traffic**
Original URL	www.youtube.com	www.facebook.com
video_codec_name	vp9	h264
audio_codec_name	opus	aac
Resolution	1440x2560	1184x2960
Video Duration	270.581 sec	270.67508 sec
Playing Duration	114.305362 sec	111.009935 sec
Received Packets	3648 (pkt)	8334 (pkt)
Received Packet Rate	30.65 (pkt/sec)	69.82 (pkt/sec)
Sent Packets	1966 (pkt)	8123 (pkt)
Sent Packet Rate	16.52 (pkt/sec)	68.06 (pkt/sec)
Received Bytes	135M	82M
Sent Bytes	270K	785K
Throughput	9M	5M
Number of Flows	129	512
QoE Metrics		
Play Start Time	1.230351 sec	0.715850 sec
Audio Rebuffering	0 times	4 times
Video Rebuffering	0 times	3 times
Quality Switches	0 times	10 times

combine video with multiple supported audio formats when uploading 360° videos. Another interesting observation is how Facebook responded to the video player during quality switches. Facebook created and destroyed the video player 7 times during 10 quality changes, while YouTube used just one instance of the video player throughout the streaming.

5 RELATED WORK

There have been several prior works that have presented tools to measure video quality metrics. PVQT[1] is a video quality measurement tool to evaluate and compare video encoding with objective quality measurements such as PSNR, Delta, MSAD, MSE, and so on. Lie et al. [13] proposed EvalVid-RA, a framework for quality measurement of the transferred video over a simulated network. In addition to frame-level quality measurement, it measures QoS metrics of the underlying network, such as loss, delay, and jitter. Although these tools are good for analysis of video quality, they do not support streaming traffic measurement.

Because the public demand for privacy requires end-to-end encryption, providers face challenges to identify QoE to improve their services. Dimopoulos et al. [5] proposed a framework to detect video streaming QoE metrics from encrypted network traffic. First, they studied video streaming from a web proxy viewpoint, including more than 390,000 non-encrypted video sessions from a large provider with 10M users. Then they analysed the video sessions to find a methodology to extract QoE metrics from network traffic. They tried to detect stall, resolution, and representation fluctuations. For evaluation of their framework, they performed a controlled experiment to detect QoE of unencrypted and encrypted video sessions. The results showed that the proposed framework can identify QoE metrics with 78% to 93.5% accuracy for unencrypted traffic, and 76% to 91.8% accuracy for encrypted traffic. However, they just use YouTube sessions to design and evaluate their framework. Predicting QoE metrics in other video streaming platforms, or even when YouTube changes its video delivery methodology, is left for future work.

[1]http://www.compression.ru/index_en.htm

Joumblatt et al. [8] developed a predictive framework that uses network performance metrics to identify user dissatisfaction with video streaming. First, they merged the user-level feedback with the low-level network metrics. Then they applied supervised machine learning techniques to build non-linear and linear SVM predictors. Although the proposed framework predicts user's dissatisfaction with acceptable accuracy, the proposed mechanism is limited by its requirement to capture and understand user's feedback and behaviour.

There are several client-side tools that have been implemented to run on user devices and measure QoE parameters more accurately. Depending on the data collection methods, these tools could be categorized into passive and active. Passive tools capture streaming data from videos that are watched by an active user, while active tools generate artificial video streaming. YoMo [23] is a passive client-side measurement tool that consists of a Java application and a Firefox plugin to measure QoE metrics of YouTube videos. It constantly monitors the playout buffer status of the YouTube application and predicts the QoE metrics. YoMo does not have any correlation with the video characteristics, and the accuracy depends on the bandwidth. Pytomo [9] is an active measurement tool that crawls YouTube to measure the network latency metrics, the QoE metrics, and the CDN information of videos as if they are being viewed by a user.

Recently, there have been some studies where mobile apps have been built to perform passive or active measurements to gather network traces. However, some of these apps have limitations due to their operating system capabilities, memory requirements, or a lack of programming libraries. DynoDroid [14] and PUMA [7] modify the device OS to track smartphone traffic at runtime. Some drawbacks of these approaches are that they require device rooting, and void the owner's warranty [18]. Furthermore, dynamic analysis causes significant runtime overheads in terms of memory and bandwidth usage, which may reduce smartphone performance and produce misleading results. For example, running the monitoring app alongside video streaming, where the monitoring app requires memory for runtime analysis and cloud data storage, could negatively impact video player activities such as rendering and buffering. Our tool focuses on the network level, and does not interfere with the mobile traffic. Our approach thus has advantages in terms of memory, performance, accuracy, and traffic visibility.

Beyond the domain of video streaming, there are some mobile security tools that provide transparency into mobile network traffic. PATHspider [11] is an open-source active measurement tool that provides protocol transparency. PATHspider performs a controlled comparison between two different protocols or protocol extensions. Meddle [16] is a framework for enhancing transparency in mobile networks. It combines a virtual private network (VPN) with middleboxes to provide an experimental platform that enables users to control mobile traffic. It exploits VPNs to tunnel network traffic through the Meddle server, from mobile devices to the target machine where the researchers can control the network flows. However, this approach has overhead in terms of power consumption and network latency. Also, there is a security problem, since the mobile data goes through the third party server.

Ren et al. [18] developed ReCon, a machine-learning-based privacy tool that intercepts network traffic to detect PII leaks. It enables a user to have control over the transmitted data. They conducted a security measurement of popular mobile apps available in Android, iOS, and Windows Phone. They used Meddle [16] on the flows generated from devices. In a manual test, they interacted with 100 popular apps. In an automated test, they used a Monkey app to test 1000 popular apps. Monkey is a command-line tool to perform regression testing on application by sending pseudo-random streams of keystrokes, touches, and gestures to a device. The results indicated that the information leaked by an app varied across operating systems. Besides, they observed that some applications leak information through SSL encryption. Although these tools provide visibility of mobile usage, they cause memory and processing overhead, increase battery consumption, and violate the warranty.

6 CONCLUSION AND FUTURE WORK

In this paper, we presented MoVIE, a video streaming measurement tool that provides visibility into network traffic from mobile devices. It is an open-source and cross-platform solution to study video streaming behaviour. MoVIE enables video streaming researchers to have a comprehensive view of flows and network traffic generated by a mobile device, and also video player activities. We illustrate MoVIE's functionality with a small case study of 360° video streaming via Facebook and YouTube.

Our case study shows that MoVIE is a useful tool for mobile video streaming studies. The source code and a demo are available from our project Web site [6, 10]. Our ongoing work is extending the tool to support iOS devices, other QoE metrics, and other network data sets for mobile video streaming.

MoVIE currently uses the Google Chrome browser as the web-based video player. It uses Chrome's properties to extract video player status. However, MoVIE can provide similar functionalities (except video player view), with other browsers like Safari and Firefox. To deal with this, we aim to add machine learning techniques to MoVIE to predict video quality, stall, and delay of video streaming. As we see in the case study, MoVIE is able to analyze and compare up to two video streaming sessions. However, extending MoVIE to analyze more video streaming events requires more memory to load and analyze packet trace files.
In our future work, we plan to make MoVIE an online tool to store and process video streaming in the cloud. In addition, we will use MoVIE to study and characterizate different video streaming techniques. Since MoVIE is an open-source tool, other contributors can adapt it to study mobile network usage.

ACKNOWLEDGEMENTS

Financial support for this work was provided by Canada's Natural Sciences and Engineering Research Council (NSERC). The authors thank the anonymous ICPE reviewers for their constructive feedback and suggestions on an earlier version of this paper.

REFERENCES

[1] M. Arlitt and C. Williamson, "Understanding Web Server Configuration Issues", *Software: Practice and Experience*, Vol. 34, No. 2, pp. 163-186, February 2004.
[2] Cisco. "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017-2022 White Paper".
[3] A. Cortesi, M. Hils, T. Kriechbaumer, and contributors, "MITMproxy: A free and open source interactive HTTPS proxy", 2010-, https://MITMproxy.org/ [V4.0].
[4] Device Atlas, "The Complete Guide to User Agents", https://deviceatlas.com
[5] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring Video QoE from Encrypted Traffic", *Proceedings of the 2016 Internet Measurement Conference*, pp. 513-526, Santa Monica, California, USA, November 2016.
[6] GitHub, MoVIE Software Tool, https://github.com/Keshvadi/movie.
[7] S. Hao, B. Liu, S. Nath, W. Halfond, and R. Govindan, "PUMA: Programmable UI-automation for Large-scale Dynamic Analysis of Mobile Apps", *Proceedings of MobiSys*, pp. 204-217, Bretton Woods, New Hampshire, USA, June 2014.
[8] D. Joumblatt, J. Chandrashekar, B. Kveton, N. Taft, and R. Teixeira, "Predicting User Dissatisfaction with Internet Application Performance at End-Hosts", *Proceedings IEEE INFOCOM*, pp. 235-239, Turin, Italy, April 2013.
[9] P. Juluri, L. Plissonneau, and D. Medhi, "Pytomo: A Tool for Analyzing Playback Quality of YouTube Videos", *Proceedings of International Teletraffic Congress* (ITC), San Francisco, USA, pp. 304-305, September 2011.
[10] S. Keshvadi, MoVIE Software Tool, https://www.cpsc.ucalgary.ca/~sina.keshvadi1/movie.
[11] I. Learmonth, B. Trammell, M. Kuhlewind, and G. Fairhurst, "PATHspider: A Tool for Active Measurement of Path Transparency", *Proceedings of Applied Networking Research Workshop*, pp. 62-64, Berlin, Germany, July 2016.
[12] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over HTTP dataset", *Proceedings of the 3rd Multimedia Systems Conference (MMSys)*, pp. 89-94, Chapel Hill, North Carolina, USA, February 2012.
[13] A. Lie and J. Klaue, "EvalVid-RA: Trace-Driven Simulation of Rate Adaptive MPEG-4 VBR Video", *Multimedia Systems*, Vol. 14, No. 1, pp. 33-50, June 2008.
[14] A. Machiry, R. Tahiliani, and M. Naik, "Dynodroid: An Input Generation System for Android Apps", *Proceedings of the Joint Meeting on Foundations of Software Engineering*, pp. 224-234, St. Petersburg, Russia, August 2013.
[15] Net Market Share, "Browser market share", www.netmarketshare.com.
[16] A. Rao, J. Sherry, A. Legout, A. Krishnamurthy, W. Dabbous, and D. Choffnes, "Meddle: Middleboxes for Increased Transparency and Control of Mobile Traffic", *Proceedings of ACM CoNEXT Student Workshop*, pp. 65-66, Nice, France, December 2012.
[17] A. Razaghpanah, R. Nithyanand, N. Vallina-Rodriguez, S. Sundaresan, M. Allman, C. Kreibich, and P. Gill, "Apps, Trackers, Privacy and Regulators: A Global Study of the Mobile Tracking Ecosystem", *Proceedings of Network and Distributed System Security Symposium (NDSS 2018)*, pp. 1-15, San Diego, CA, USA, February 2018.
[18] J. Ren, A. Rao, M. Lindorfer, A. Legout, and D. Choffnes, "ReCon: Revealing and Controlling PII Leaks in Mobile Network Traffic", *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pp. 361-374, Singapore, Singapore, June 2016.
[19] RFC 7231, Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, IETF, June 2014.
[20] M. Siekkinen, E. Masala, and T. Kamarainen, "A First Look at Quality of Mobile Live Streaming Experience: the Case of Periscope", *Proceedings of the 2016 Internet Measurement Conference*, pp. 477-483, California, USA, November 2016.
[21] L. Skorin-Kapov, M. Varela, T. Hobfeld, and K. Chen, "A Survey of Emerging Concepts and Challenges for QoE Management of Multimedia Services", *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, Vol. 14, No. 2s, pp. 1-29, May 2018.
[22] K. Spiteri, R. Sitaraman, and D. Sparacio, "From theory to practice: improving bitrate adaptation in the DASH reference player", *Proceedings of the 9th ACM Multimedia Systems Conference*, pp. 123-137, Amsterdam, Netherlands, June 2018.
[23] B. Staehle, M. Hirth, R. Pries, F. Wamser, and D. Staehle, "YoMo: A YouTube Application Comfort Monitoring Tool", *New Dimensions in the Assessment and Support of Quality of Experience for Multimedia Applications*, Vol. 6, Tampere, Finland, 2010.
[24] Statista - Global mobile data traffic from 2017 to 2022. https://www.statista.com/statistics/271405/global-mobile-data-traffic-forecast.
[25] The Wonder Shaper 1.4. https://github.com/magnific0/wondershaper.
[26] J. Wang, B. Amos, A. Das, P. Pillali, N. Sadeh, and M. Satyanarayanan, "A Scalable and Privacy-Aware IoT Service for Live Video Analytics", *Proceedings of the 8th ACM on Multimedia Systems Conference*, pp. 38-49, Taipei, Taiwan, June 2017.
[27] M. Xiao, C. Zhou, Y. Liu, and S. Chen, "OpTile: Toward Optimal Tiling in 360-degree Video Streaming", *Proceedings of the 25th ACM international conference on Multimedia*, pp. 708-716, California, USA, October 2017.
[28] M. Zink, R. Sitaraman, and K. Nahrstedt, "Scalable 360° Video Stream Delivery: Challenges, Solutions, and Opportunities", *Proceedings of the IEEE*, Vol. 107, No. 4, pp. 639-650, April 2019.