Open Video Datasets over Operational Mobile Networks with MONROE

Cise Midoglu
Simula Research Laboratory
cise@simula.no

Mohamed Moulay
IMDEA Networks Institute
mohamed.moulay@imdea.org

Vincenzo Mancuso
IMDEA Networks Institute
vincenzo.mancuso@imdea.org

Özgü Alay
Simula Metropolitan Center for Digital Engineering
ozgu@simula.no

Andra Lutu
Telefonica Research
andra.lutu@telefonica.com

Carsten Griwodz
University of Oslo
griff@ifi.uio.no

ABSTRACT

Video streaming is a very popular service among the end-users of Mobile Broadband (MBB) networks. DASH and WebRTC are two key technologies in the delivery of mobile video. In this work, we empirically assess the performance of video streaming with DASH and WebRTC in operational MBB networks, by using a large number of programmable network probes spread over several countries in the context of the MONROE project. We collect a large dataset from more than 300 video streaming experiments. Our dataset consists of network traces, performance indicators captured during the streaming sessions, and experiment metadata. The dataset captures the wide variability in video streaming performance, and unveils how mobile broadband is still not offering consistent quality guarantees across different countries and networks, especially for users on the move. We open source our complete software toolset and provide the video dataset as open data.

CCS CONCEPTS

• **Networks** → **Mobile networks**; *Network experimentation*; • **Information systems** → Multimedia streaming;

KEYWORDS

Dynamic Adaptive Streaming over HTTP (DASH), Mobile Broadband (MBB) Networks, Video on Demand (VoD), WebRTC

ACM Reference Format:
Cise Midoglu, Mohamed Moulay, Vincenzo Mancuso, Özgü Alay, Andra Lutu, and Carsten Griwodz. 2018. Open Video Datasets over Operational Mobile Networks with MONROE. In *MMSys'18: 9th ACM Multimedia Systems Conference, June 12–15, 2018, Amsterdam, Netherlands.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3204949.3208138

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

ACM ISBN 978-1-4503-5192-8/18/06…$15.00
https://doi.org/10.1145/3204949.3208138

1 INTRODUCTION

Video streaming is one of the most prominent applications running over the Internet today. Forecasts indicate that video traffic will constitute over 80% of global data traffic by 2021 [4]. Two of the key techniques for standardized video delivery are Dynamic Adaptive Streaming over HTTP (DASH) and WebRTC. DASH is a client-server based approach used for adaptive on-demand streaming over HTTP, with various available rate adaptation algorithms [8], where WebRTC is a solution that enables web browsers to communicate with each other, supporting voice, video and data communication in a peer-to-peer fashion with no need for servers, plugins or additional software [20].

The popularity of DASH and WebRTC video services is growing rapidly in Mobile Broadband (MBB) networks, where the available bandwidth offered by Mobile Network Operators (MNOs) is constantly increasing. However, although increased bandwidth is expected to enhance the performance of the delivery, users of video services still report unreliable Quality of Experience (QoE) [19]. Therefore, understanding how different streaming strategies work in MBB networks is crucial, and services with better and certified QoE guarantees need to be designed for the constantly increasing number of mobile customers.

Analyzing the performance of different applications over MBB networks in an *empirical* manner is of paramount importance, since new services are developed continuously and do not wait for approximated models and network optimization studies before going live. Indeed, they go to the market and are sold without waiting for MNOs to adjust [17]. It is notoriously difficult to obtain systematic data from end-users through repeatable measurements and, at the same time, collect proper information on the conditions under which these measurements run [9, 16, 18]. To overcome this problem, we have designed, developed and deployed a dedicated hardware platform in the context of the Measuring Mobile Broadband Networks in Europe (MONROE) project [1]. MONROE runs measurement probes from an end-user perspective (from the access network) by relying on regular commercial mobile subscriptions. MONROE nodes are currently operational in 7 European countries, from where it is possible to monitor more than 15 MNOs.

In this paper, we focus on Video on Demand (VoD) streaming in MBB networks and empirically assess streaming performance by leveraging MONROE nodes from stationary and mobile vantage points. We evaluate the performance of HTTP through a regular

DASH implementation, and the performance of Real-time Transport Protocol (RTP) by using WebRTC in video streaming mode. We also compare the performance of different DASH rate adaptation algorithms, and different MNOs.

Experiments running on the MONROE platform are designed as Docker containers, in which necessary measurement tools are packaged, and all of them are available as open source [15]. In this work, we develop the containers `MONROE-AStream` [5] and `WebStreamer` [6]. The former implements DASH with 3 alternative rate adaptation algorithms, and the latter implements WebRTC using HTTPS. We show that our containers enable monitoring and assessing DASH and WebRTC performance over MBB from stationary and mobile devices in a repeatable and controllable manner.

MONROE captures the variable end-user experience from various applications *in the wild*, in different countries and for different MNOs, with a rich dataset that is continuously updated with new measurements. This dataset enables the continuous monitoring of how services and networks evolve over time. The contribution described in this work is a part of the MONROE dataset, and includes the DASH and WebRTC measurement results, jointly with a generous quantity of *metadata* provided by the platform. Our video dataset has results from over 300 individual experiments with various log files, and more than 500K lines of metadata, including GPS, sensor and modem information. The MONROE video dataset is publicly available under [7].

2 DATA COLLECTION

In this section, we describe how our dataset is generated.

2.1 MONROE Platform

MONROE [1, 14] is a European transnational open platform, and the first open access hardware-based platform for independent, multi-homed, and large-scale MBB measurements on commercial networks. The platform comprises a set of 150 nodes, both mobile (e.g., operating in delivery trucks and on board public transport vehicles, such as trains or buses) and stationary (e.g., volunteers hosting nodes in their homes). Each node is multi-homed to 3 different MNOs. MONROE is currently operational in Italy, Norway, Spain, Sweden, Portugal, Greece and the UK. All software components used in the platform are open source and available online [15].

User access to the platform resources is through a web portal, which allow authenticated users to use the MONROE scheduler to deploy their experiments. This enables exclusive access to nodes (i.e., no two experiments run on a node at the same time). Experiments running on the platform use Docker containers (light-weight virtualized environment) to provide agile reconfiguration.

The designed DASH and WebRTC experiments can be run easily on MONROE with different configuration parameters, and measurement results from operational MBB networks can be obtained in real time. We provide templates with default parameters to run the containers directly on the MONROE scheduler interface, so that both experiments can be run with a single click.

Metadata collection. Since MONROE does not involve real users (which usually entail privacy protection restrictions), rich metadata collection, including geo-temporal tagging, is possible. MONROE nodes generate metadata passively and continuously: each node is instrumented to gather information relating to its MNOs, such as location, signal strength, and link technology.

At the node side, metadata distribution is implemented in a publish/subscribe pattern using ZeroMQ (ZMQ).[1] Metadata entries are generated in a single-line JavaScript Object Notation (JSON) format, where every entry is labeled with a "topic" field. The metadata subscriber module subscribes to all the topics, writing JSON entries to files in a special file system location. A synchronization process transfers these files to the MONROE server when no other active, periodic, or user-defined experiment is running. In this way, metadata from all MONROE nodes is collected and stored centrally.

The metadata stream is also available for experiments during their execution. Experimenters can have their containers subscribe to any of the metadata topics, and use them either during runtime or store them together with experiment results. The former allows for monitoring and reacting to events such as interface reconnections or link technology/signal strength change for each MBB interface at run-time, where the latter facilitates the joint post-processing of measurement data and metadata pertinent to a given experiment.

2.2 DASH Experiments

We design the MONROE-AStream experiment for conducting DASH measurements over operational MNOs using the MONROE platform. The overall framework consists of 5 steps and is described in Figure 1 as a flow diagram.

First, a video is selected for streaming. This video can be from any source (e.g. YouTube). The DASH representations of the video are downloaded/generated and the retrieved segments are used for Media Presentation Description (MPD) generation. An existing MPD file can also be used.

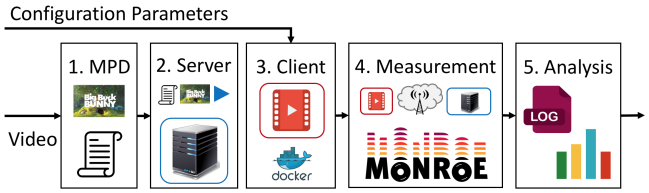


Figure 1: MONROE-AStream framework.

Then, the MPD file and the segments are uploaded to the server. The original AStream server script is modified to accommodate for the new MPD file and the location of different video representations. The server is configured to listen to a desired port. It is also possible to use an existing DASH server, which is able to respond to client requests regarding the selected video, and compatible with its corresponding MPD file.

The MONROE-AStream client consists of the AStream core with an additional patch for reading different MPD formats [2], configuration and logging functionality for MONROE-related parameters, wrapped in a MONROE-compatible Python script. The client container and the server code can run both within or outside of the MONROE platform, allowing for measurements in different setups.

The final step of our experiment framework is the analysis of output files, for which our toolset includes parsing and plotting

[1] ZMQ distributed messaging: http://zeromq.org

scripts. The experiment container is provided as open source within an Experiment as a Service (EaaS) framework, under [5].

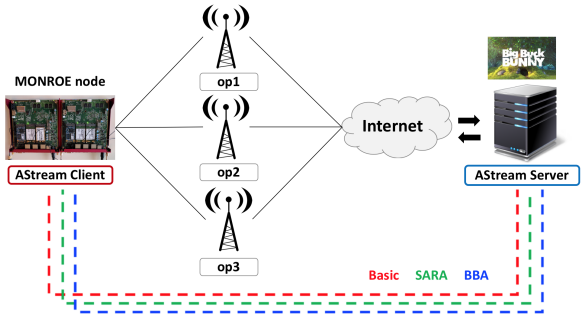


Figure 2: MONROE-AStream measurement setup.

Figure 2 depicts our experiment scenario where the MONROE platform is used as the measurement environment in Step 4. We aim to investigate the performance of different rate adaptation algorithms over different MNOs. The rate adaptation algorithms we consider are: Basic, Segment Aware Rate Adaptation (SARA) [11], and Buffer Based Adaptation (BBA) [10].

2.3 WebRTC Experiments

The main idea for the MONROE-WebRTC experiment is to test video streaming using WebRTC for mobile users. Figure 3 outlines the streaming setup. When an experiment is scheduled on a set of MONROE nodes, each node runs the `WebStreamer` container, which makes an HTTPS link available to watch the video stream coming from the node. We use a Chrome browser acting as the WebRTC client in our lab. The WebRTC client is connected from a computer wired to a well provisioned gigabit link so that the receiver is not the bottleneck. The video stream produced by the MONROE node goes through a cellular uplink, traverses the Internet and then accesses the network of our lab. Therefore, the network bottleneck experienced in a WebRTC streaming session is the MBB network used by the MONROE node.

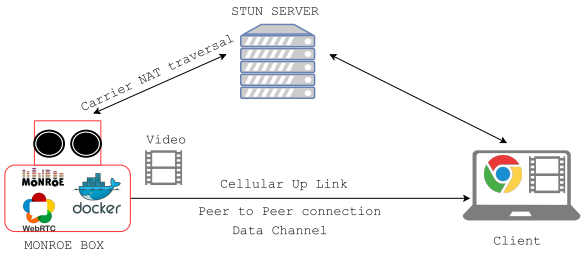


Figure 3: MONROE-WebRTC video streaming setup.

The video to stream is stored locally in the Docker container, with a resolution of 854x480 and a constant frame rate of 24 fps. The duration of the video is 9 minutes but can be looped for several hours, according to the duration of the scheduled experiment. The video streaming configuration parameters include the following:

- The possibility to use a local or online video by passing an RTSP link.
- Picking the preferred port for communication.
- Using a specific STUN server in case the client is behind a firewall.
- Tweaking resolution and duration of the video to stream.

At the client side, the video stream is received by means of Google Chrome, which also measures and collects streaming statistics in JSON format. The resulting dump contains Peer-to-Peer Connection status information between the MONROE node and the WebRTC client in our lab, in addition to updates and data statistics that can be easily accessed, e.g., by loading a specific Chrome page.[2]

The client logs contain, per each individual stream, the timing and headers of packets received as well as the timing of various internal events such as received frames, losses, bitrate, delay, jitter and other metrics that will be discussed in Section 3.

3 DATASET DETAILS

Our video dataset includes data and metadata relative to hundreds of experiments with stationary and mobile MONROE nodes in Italy, Norway, Sweden and Spain. This is a live dataset, so the dataset keeps growing. Experiment results included in the dataset so far have been collected within a period of eight months, and we are currently running more experiments to keep following the performance evolution in the monitored locations (for stationary nodes) and paths (for mobile MONROE nodes mounted on buses and trains). Our open dataset can be found under [7] and includes a `README` file explaining the organization of each directory (MONROE-AStream and MONROE-WebRTC).

Here, we provide a list of parameters that are available in the MONROE dataset relevant to video measurements. These come from the MONROE-AStream and WebStreamer experiments, and the MONROE metadata collection process.

3.1 MONROE Metadata

Table 1 illustrates the metadata "topics", which are streamed to subscriber entities within MONROE nodes using ZMQ, and the relevant experiments running regularly on the platform.

Table 1: MONROE Metadata topics and experiments.

Class	Type	Examples
Node	Sensor	CPU temperature
Node	Probe	Load, memory usage
Node	Event	Power up, reboot
Device	GPS	GPS coordinates
Device	Modem	RSSI, link technology, CID, IP
Experiment	RTT	Ping RTT
Experiment	Tstat	TCP statistics
Experiment	Bandwidth	Nettest TCP throughput

Metadata includes network parameters such as Received Signal Strength Indicator (RSSI), Reference Signal Received Power (RSRP), Reference Signal Received Quality (RSRQ), cell identifiers, link technology, node location and speed via GPS, node working parameters such as CPU temperature and load, and node events (watchdogs).

In our open dataset, we include samples from the `node.sensor` and `device.modem` classes for the duration of MONROE-AStream experiments, and samples from the `device.gps` class for the WebStreamer experiments under mobility.

[2] chrome://webrtc-internals

3.2 MONROE-AStream Dataset

MONROE-AStream produces an experiment summary in JSON format, 3 types of detailed logs, and the streamed segments in raw form (for which the download option is a configuration parameter). These are ready to download in the form of 2 files after experiment completion: one JSON file as the summary, and one compressed archive for all logs and video segments, both with the filename stub *MONROE.EXP.ASTREAM_<video>_<adaptation>_<nodeid>_<interface>_<timestamp>*. Samples for each type of output can be found in our open dataset [7].

Experiment Summary. This is a JSON file with a single-level dictionary, generated per experiment. It lists the basic configuration parameters and platform-related information for the experiment, such as timestamp, node id, operator, container version, video name, adaptation algorithm, segment limit, and experiment tag.

Runtime Logs. These are `log` files, providing the complete command line output of the code execution as a file, generated per experiment interface. They give a detailed overview of the experiment execution with timestamps, and enable troubleshooting.

Segment Logs. These are JSON files with a 4-level dictionary, including the following fields, generated per experiment interface.

- ***playback_type:*** indicates the selected adaptation algorithm (Basic, SARA, or BBA).
- ***segment_info:*** contains an entry per each streamed segment, indicating the segment filename, bitrate, size, and time to download.
- ***playback_info:*** contains the start time (*start_time*), end time (*end_time*), number of up switches (*up_shifts*), number of down switches (*down_shifts*), duration of initial buffering in seconds (*initial_buffering_duration*), and information regarding buffering events, i.e. stalls (*interruptions*). The *interruptions* object includes the fields *count*, *total_duration*, and *events*.
- ***video_metadata:*** contains the total playback duration (*playback_duration*), and information regarding the available representations of the given video (*available_bitrates*). The latter includes an object with fields *width*, *bandwidth*, *height*, *frameRate*, and *codecs* per each representation.

Buffer Logs. These logs are Comma-Separated Values (CSV) files with event-based entries including the following fields, generated per experiment interface.

- ***EpochTime:*** unix timestamp.
- ***CurrentPlaybackTime:*** current playback time in seconds, with a granularity of segment duration (e.g. 2*s*).
- ***CurrentBufferSize:*** current buffer size in seconds.
- ***CurrentPlaybackState:*** current playback state as one of the following; `INITIAL_BUFFERING`, `PLAY`, `STOP`.
- ***Action:*** current action as one of the following; `Starting`, `Writing`, `InitialBuffering-Play`, `StillPlaying`, `Stopped`.
- ***Bitrate:*** current segment bitrate in bits per second.

Downloaded Segments. It is possible to download the streamed segments at the client side in raw form, with a configuration parameter. This option gives experimenters the ability to reconstruct the streamed video for playback.

3.3 MONROE-WebRTC Dataset

WebStreamer generates two types of log files. The first one is our Docker container log, which consists of the following:

- HTTP link to connect too.
  `2018-xx-xxTxx:xx:xx your url is: https://xxxx`
- The videos you are going to watch.
  `2018-xx-xxTxx:xx:xx.xx answer:[`
  `2018-xx-xxTxx:xx:xx.xx "xx.mp4,rtsp://xxxxx"]`
- The open port listening to upcoming connections.
  `2018-xx-xxTxx:xx:xx.xx HTTP Listen at :8888`
- The constant Framerate of the video.
  `2018-xx-xxTxx:xx:xx.xx a=framerate:24.0`
- The video codec used that includes h264, VP8, and VP9.
  `2018-xx-xxTxx:xx:xx Created a data sink for the "video/H264/VP8/VP9" subsession`
- The STUN server used when establishing connection between client and host.
  `2018-xx-xxTxx:xx:xx.xx{"url" : "stun:stun.l.google.com:19302"`
  `2018-xx-xxTxx:xx:xx.xx }`
- The preferred candidate for connection.
  `body:{"candidate":"candidate:0 2 UDP 2122252542 192.xx.x.x 42479 typ host", "sdpMid":"sdparta_0","sdpMLineIndex":0}`
  `answer:1`
- The Video Content Type which is an extension used to communicate a video content type from sender to receiver of RTP video stream.
  `2018-xx-xxTxx:xx:xx.xx a=rtpmap:96 mpeg4-generic/48000/2`

When the connection is established, the WebRTC receiver starts collecting the JSON dump from Google Chrome, which is our second log. This includes different metrics: *bitrate received, frames decoded, packets lost during the session, packets received per second, video current delay in ms, frame rate received, decoded and output, jitter delay in ms, number of ACKs sent, target delay in ms.*

The MONROE-WebRTC template and the visualization tool that helps import all the metrics mentioned in CSV format can be found at [7] and [6].

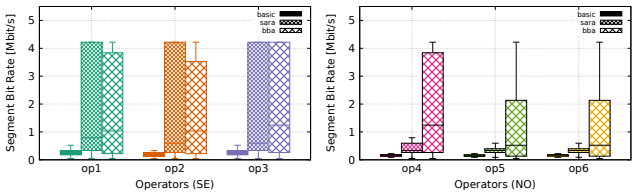
4 APPLICATIONS OF THE DATASET

The MONROE video dataset is unique in two aspects: it incorporates video measurements from *operational* MBB networks, and it provides *context information* for all measurements. The former allows for the analysis of realistic MBB scenarios, including mobility, whereas the latter allows researchers to correlate their results with metadata. As an example, the performance of a rate adaptation algorithm may be related to the underlying network conditions during its execution, such as signal coverage.

MONROE-AStream and MONROE-WebRTC experiment results can be used for a number of purposes. For instance, it is possible to evaluate the performance of different MNOs, or the performance of different DASH rate adaptation algorithms [12]. It is also possible to compare performance under mobility with stationary scenarios [13]. In the following, we present an analysis regarding MNO and rate adaptation algorithm performance. Note that we do not report the name of the MNOs in our results because we do not intend to carry on an explicit comparison between MNOs.[3]

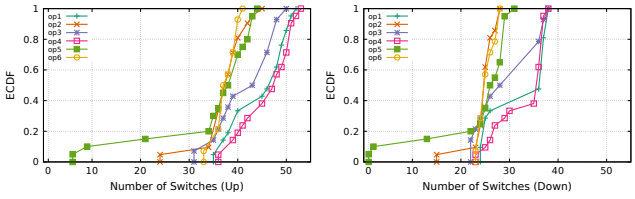
MONROE-AStream. We use the DASH server from ITEC with the readily available MPD file for the BigBuckBunny video from their open *MMSys* dataset (20 representations), selecting 2*s* duration for segments [3]. We run the MONROE-AStream client on stationary nodes in Norway and Sweden, using 6 operational MBB networks, with 3 rate adaptation algorithms (Basic, SARA, and BBA) with an option to download the streamed video segments.

[3]The MNO names and exact GPS coordinates of the nodes used in the experiments are made available as metadata in the dataset.



(a) Segment Bit Rate (Sweden) (b) Segment Bit Rate (Norway)

Figure 4: MONROE-AStream segment bitrate from experiments run on stationary nodes in Sweden and Norway. Results are split according to operator and adaptation algorithm, per country.



(a) Up-Switches (b) Down-Switches

Figure 5: MONROE-AStream number of quality switches per operator, from experiments run on stationary nodes in Sweden and Norway (adaptation: SARA).

We specifically choose these countries since they have higher SIM quotas, allowing a larger number of measurements compared to other countries.

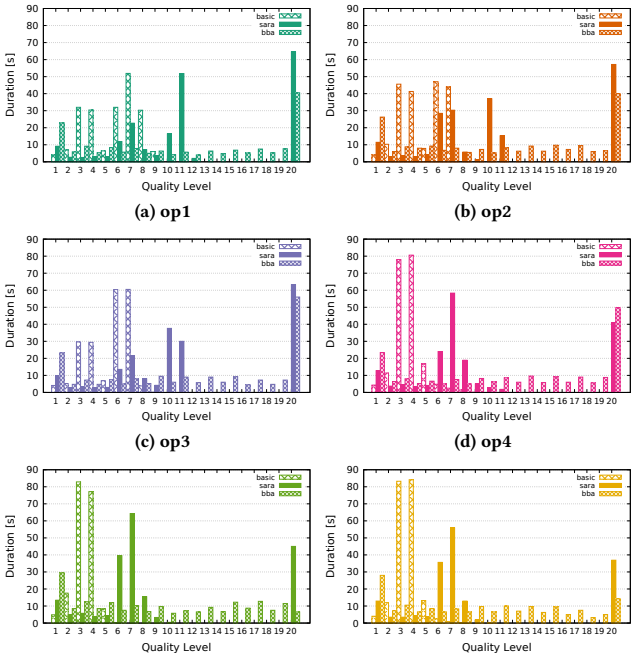
Using the MONROE-AStream experiments, it is possible to evaluate metrics such as segment bit rate, duration of stalls (buffering), number of up/down switches, and time spent at different video quality levels. It is possible to evaluate the performance of MNOs and adaptation algorithms separately, as well as compare their combined performance.

Figure 4 shows the boxplot of segment bitrate for each adaptation algorithm and operator in (a) Sweden and (b) Norway. As one of the most prominent indicators of video streaming quality, segment bitrate can be used to compare the performance of MNOs as well as adaptation algorithms. We see a higher overall video bitrate on average for nodes on Sweden when compared to nodes in Norway, and a higher bitrate on average for adaptation algorithms SARA and BBA compared to Basic.

Figure 5 shows the Empirical Cumulative Distribution Function (ECDF) of the total duration of up and down-switches for different MNOs, for a single adaptation algorithm (SARA). As an indicator of possible fluctuations in the available network throughput, as well as user experienced quality, the number of switches gives insights into the performance of video streaming as experienced by the clients. From our measurements, we observe that operators op2, op5, and op6 switch less in both directions as compared to operators op1, op3, and op4, on average.

Figure 6 shows the total time spent at each quality level for different operators. We observe for all operators that, while the Basic algorithm streams video segments mostly from the lower quality layers, SARA can stretch further to medium levels, with a jump to the highest quality after more than half of the total duration, and BBA traverses the low and medium quality levels quite quickly to spend most of the time at the highest quality level.

MONROE-WebRTC. Here, we use a sample of our dataset to demonstrate the performance analysis of WebRTC. We use WebRTC



(a) op1 (b) op2

(c) op3 (d) op4

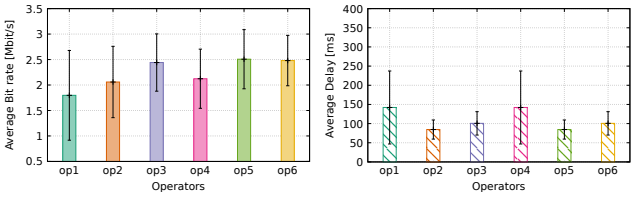
(e) op5 (f) op6

Figure 6: MONROE-AStream average time spent at each video quality level per operator and adaptation algorithm, from experiments run on stationary nodes in Sweden and Norway. Results are split according to operator (op1-3 in Sweden, op4-6 in Norway).

video data for a few stationary nodes across Sweden and Norway, and for three mobile nodes located in regional trains in Norway, in the Oslo metropolitan area and its surroundings. The video statistics we show here are the ones coming from the logs built by means of Chrome WebRTC internals, at the receiver side. We also use MONROE metadata generated at the MONROE nodes to complement and interpret the video data.

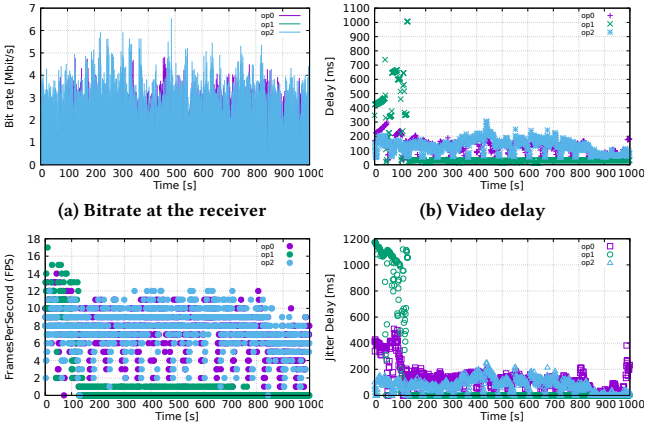
For experiments with stationary nodes, we select WebRTC streamers located aboard MONROE nodes with good MBB connectivity. Specifically, we consider WebRTC streams using as uplink an MBB connection under good 4G coverage. In Figure 7, we outline the average bitrate and average delay experienced during a connection. Error bars in the figure indicate the standard deviation. A total number of 6 nodes were used for measuring 6 different MNOs. We split results according to the country of origin of the streams in the experiment, which is either Sweden or Norway, whereas the receiver was located in all cases in Madrid, Spain. We further split the results per MNO. As shown in the figure, the bitrate was consistently high. The figure also shows that the one-way delay observed during the video streaming sessions was acceptable and, in most of the cases, well below 150 ms, which is the upper limit beyond which a human being starts noticing the delay and gets annoyed.

For experiments with mobile streamers, we selected MONROE nodes traveling on trains. Here we report an example of three time series obtained from three trains moving in and around Oslo in the same time interval. For this mobile case, Figure 8 depicts instantaneous values of bitrate received, video frame rate reconstructed at destination, and video delay and jitter experienced by the receiver. The figure shows that performance under mobility can be relatively good, at least under good cellular coverage. However, cellular



(a) Average bitrate (b) Average delay

Figure 7: MONROE-WebRTC average performance for the experiments run from stationary MONROE nodes in Sweden (op1-3) and Norway (op4-6). Results are split according to operator and country of origin of the WebRTC streams.



(a) Bitrate at the receiver (b) Video delay

(c) Frames per second at the receiver (d) Video jitter delay

Figure 8: MONROE-WebRTC performance time series for an experiment with streamers located aboard trains moving in the region of Oslo, Norway, using different MBB operators (one stream per train, different operator per stream).

coverage in mobility is not always available in the observed area. By analyzing the metadata associated with the traces reported in Figure 8, we have noticed that one of the trains used for the experiment progressively moved from the city center to the outskirts of Oslo, while the other trains stayed within the city, moving between multiple stations in Oslo. The streams from the trains remaining in the city experienced almost always good performance, albeit with high variability. On the other hand, the stream from the train moving outside of Oslo shows poor performance at the beginning of the experiment, when the stream suffered a huge degradation in bitrate and a significant increase in delay and jitter, and later suffers an almost complete connection drop (almost no delay reports were received, and only sporadic frames were reproduced). This example leads to the conclusion that current MBB networks are not ready yet to fully support WebRTC and similar video services everywhere on the move.

5 CONCLUSIONS

In this paper, we have described the MONROE platform and two containers developed on top of this platform to study the performance of video streaming in operational MBB networks. Using `MONROE-AStream` and `WebStreamer`, we investigated the effect of different protocols and algorithms on video streaming performance, and evaluated several operator networks in multiple countries. The measurements so far unveiled that, while mobile operators still have to strive for offering consistent quality guarantees, especially for users on the move, both DASH and RTP can provide a relatively stable performance over mobile networks when the users are stationary.

Our results are provided as an open dataset, which comprises not only video-related quality metrics, but also rich metadata coming from the MONROE platform. These are useful to interpret the *context* of the measurements in terms of MBB parameters, and allow for recreating certain scenarios. Our complete software toolset for experimentation and data analysis, including the Docker containers implementing DASH [5] and WebRTC [6], are also provided as open source. Being completely open source, our work can be extended and reused with minor effort.

ACKNOWLEDGMENTS

This work is funded by the EU H2020 research and innovation programme under grant agreement No. 644399 (MONROE), and by the Norwegian Research Council project No. 250679 (MEMBRANE). The work of V. Mancuso was supported by the Ramon y Cajal grant (ref: RYC-2014-16285) from the Spanish Ministry of Economy and Competitiveness.

REFERENCES

[1] Ö. Alay, A. Lutu, M. Peón-Quirós, V. Mancuso, T. Hirsch, K. Evensen, A. Hansen, S. Alfredsson, J. Karlsson, A. Brunström, A. Safari Khatouni, M. Mellia, and M. Ajmone Marsan. 2017. Experience: An Open Platform for Experimentation with Commercial Mobile Broadband Networks. *Mobicom* (2017).
[2] AStream. 2018. https://github.com/pari685/AStream[commit:aadcfe63]
[3] ITEC Big Buck Bunny. 2018. http://www-itec.uni-klu.ac.at/ftp/datasets/mmsys12/BigBuckBunny/
[4] Cisco. 2017. *Cisco Visual Networking Index: Forecast and Methodology, 2016–2021.* Technical Report.
[5] MONROE-AStream Container. 2018. https://github.com/acmmmsys/2018-MONROE-astream/
[6] MONROE-WebStreamer Container. 2018. https://github.com/acmmmsys/2018-MONROE-webstreamer/
[7] MONROE Video Dataset. 2018. https://doi.org/10.5281/zenodo.1188410
[8] S. Hu, L. Sun, C. Gui, E. Jammeh, and I. H. Mkwawa. 2014. Content-Aware Adaptation Scheme for QoE Optimized DASH Applications. *GLOBECOM* (2014).
[9] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z.-M. Mao, S. Sen, and O. Spatscheck. 2013. An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance. *SIGCOMM* (2013).
[10] T. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. 2014. A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. *SIGCOMM* (2014).
[11] P. Juluri, V. Tamarapalli, and D. Medhi. 2015. SARA: Segment Aware Rate Adaptation Algorithm for Dynamic Adaptive Streaming over HTTP. *ICCW* (2015).
[12] C. Midoglu, K. Kousias, C. Griwodz, and Ö. Alay. 2017. Evaluation of DASH Rate Adaptation Algorithms in Operational Mobile Networks (poster). *TMA PhD School* (2017).
[13] M. Moulay and V. Mancuso. 2018. Experimental Performance Evaluation of WebRTC Video Services over Mobile Networks. *INFOCOM CNERT* (2018).
[14] M. Peón Quirós, V. Mancuso, V. Comite, A. Lutu, Ö. Alay, S. Alfredsson, J. Karlsson, A. Brunstrom, M. Mellia, A. Safari Khatouni, and T. Hirsch. 2017. Results from Running an Experiment as a Service Platform for Mobile Networks. *Mobicom WiNTECH* (2017).
[15] MONROE Project Repository. 2018. https://github.com/MONROE-PROJECT/
[16] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, S. Venkataraman, and J. Wang. 2013. A First Look at Cellular Network Performance during Crowded Events. *SIGMETRICS* (2013).
[17] Technavio. 2016. *Technavio Global Video Services Market 2016–2020.* Technical Report.
[18] N. Vallina-Rodriguez. 2017. Illuminating the Third Party Mobile Ecosystem with the Lumen Privacy Monitor. *FTC PrivacyCon* (2017).
[19] C. Wang, D. Bhat, A. Rizk, and M. Zink. 2017. Design and Analysis of QoE-Aware Quality Adaptation for DASH: A Spectrum-Based Approach. *TOMM* (2017).
[20] A. Zeidan, A. Lehmann, and U. Trick. 2014. WebRTC Enabled Multimedia. *WTC* (2014).