DASH QoE Performance Evaluation Framework with 5G Datasets

Raza Ul Mustafa
University of Campinas (UNICAMP)
Brazil
razaul@dca.fee.unicamp.br

Md Tariqul Islam
University of Campinas (UNICAMP)
Brazil
tariqsaj@dca.fee.unicamp.br

Christian Rothenberg
University of Campinas (UNICAMP)
Brazil
chesteve@dca.fee.unicamp.br

Simone Ferlin
Ericsson AB
Sweden
simone.ferlin@ericsson.com

Darijo Raca
University of Sarajevo, Sarajevo, BiH
draca@etf.unsa.ba

Jason J. Quinlan
University College Cork, Ireland
j.quinlan@cs.ucc.ie

***Abstract*—Fifth Generation (5G) networks provide high throughput and low delay, contributing to enhanced Quality of Experience (QoE) expectations. The exponential growth of multimedia traffic pose dichotomic challenges to simultaneously satisfy network operators, service providers, and end-user expectations. Building QoE-aware networks that provide run-time mechanisms to satisfy end-users' expectations while the end-to-end network Quality of Service (QoS) varies is challenging, and motivates many ongoing research efforts. The contribution of this work is twofold. Firstly, we present a reproducible data-driven framework with a series of pre-installed Dynamic Adaptive Streaming over HTTP (DASH) tools to analyse state-of-art Adaptive Bitrate Streaming (ABS) algorithms by varying key QoS parameters in static and mobility scenarios. Secondly, we introduce an interactive Jupyter notebook and Binder service providing a live analytical environment, which processes the output dataset of the framework and compares the relationship of five QoE models, three QoS parameters (RTT, throughput, packets), and seven different video KPIs.**

***Index Terms*—5G, QoE , QoS, ABS algorithm, DASH**

I. Introduction

5G is expected to support significantly high bandwidth content with speeds in excess of 10 GB/s, very low (i.e. 1-millisecond) end-to-end over-the-air latency, real-time information transmission, and lower network management operation complexity [1]. The key challenge of streaming video in 5G is soothing the juxtaposition of the increased growth of multimedia traffic and user satisfaction. On average, multimedia users spend six hours a day watching different streaming content[1]. Furthermore, the recent coronavirus (COVID-19) pandemic has dramatically increased the amount of video streaming in 2020 [2].

The impact of end-user QoE for multimedia traffic ultimately depends on underlying network-level Quality of Service (QoS) performance. QoE represents the user perception on the quality of a provided service whereas QoS relates to network quality indicators (e.g., latency, packet loss).

In HTTP Adaptive Streaming (HAS), the choice of the Adaptive Bitrate Streaming (ABS) algorithm plays a significant role in end-user satisfaction [3]. In recent years, the goal of many ABS algorithms is to provide interrupt-free videos and hence provide maximum achievable video quality. These ABS algorithms works on the principal by calculating network condition and utilize the maximum resources thus provide better video quality during a video session. Comparing different ABS algorithms is a non-trivial task, some algorithms focus on smooth streaming, resulting in lower bitrate and fewer quality switching. Other algorithms aim is to provide high quality content, utilizing more network resources, irrespective to the number of stalls (freezing). Ultimately, the main goal of all ABS algorithms is to provide best the QoE to end-users.

With the exponential growth of mobile data and smart devices, the investigation of 5G QoE in terms of video quality assessment has become a research focus both in industry and academia. Video perceived quality in 5G network is critical thus various methods have been used to optimize video delivery over 5G networks such as video compression and better resource utilization [4], [5]. In 5G/future networks QoE management is crucial as the estimation and resource allocation for better video quality should be completed quickly. Although 5G networks are still at conceptual stage, it is necessary to understand the correlation between ABS behaviour, its metrics for QoE and network-level QoS.

The contributions presented in this paper are divided into two phases: *Phase 1* presents a multi-user reproducible framework containing (i) godash - an ABS video player [6], (ii) Caddy - a WSGI web server hosting DASH video content, (iii) Mininet-Wifi - a wireless network emulation environment [7], (iv) Scripts - Bash scripts to apply the 5G bandwidth values sampled from the 5G traces [2] at run-time;

[1]https://www.nielsen.com/us/en/insights/report/2018/q2-2018-total-audience-report/

[2]5G traces taken from publicly available dataset that contains throughput, channel and context information for 5G networks: https://github.com/uccmisl/5Gdataset

978-3-903176-31-7 © 2020 IFIP

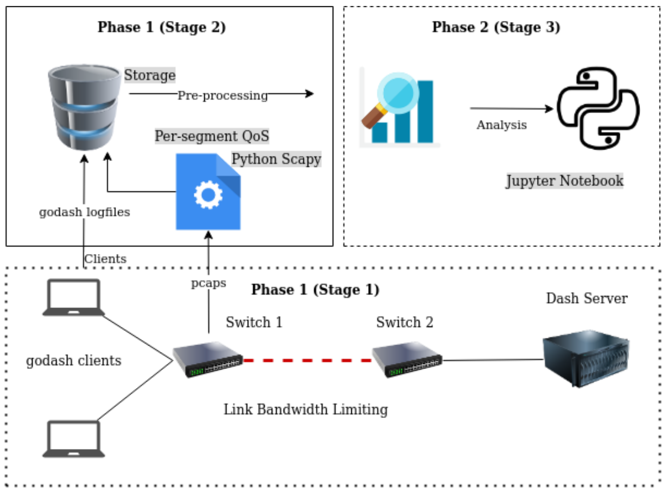


Fig. 1: Phase 1 (Stage 1), DASH streaming environment, Phase 2 (Stage 2), godash logfiles and per-segment QoS processing, Phase 2 (Stage 3), Jupyter notebook interacting with the processed dataset

and Python scripts to process the per segment QoE/QoS logs created during experimentation.

Phase 2 receives as input the processed QoE/QoS dataset from the first phase and demonstrates an interactive Jupyter notebook to analyse the ABS algorithm with specific objective QoE KPIs, per-segment QoS features and the output of five QoE models Claey [8], Dunamu [9], Yin [10], and Yu [11] and ITU-T Rec. P.1203 standard [12], [13] (mode 0 considering metadata only, bitrate, frame rate, and resolution). The framework uses the pre-installed ABS algorithms provided by godash [6]. The available algorithms are categorised as: Rate-based — Conventional [14] and Exponential, Buffer-based — Logistic [15] and BBA [16], and Hybrid — Arbiter+ [17] and Elastic [18].

The rest of the paper is structured as follows: Section II presents background and related work. Section III describes the proposed framework in Phase 1 followed by the dynamic analysis of Phase 2 in Section IV. The experimental use case is presented in Section V. Section VI concludes our paper and discuses some future work.

II. BACKGROUND AND RELATED WORK

In adaptive streaming, video content is split into multiple segments, typically with an individual segment duration of between 2 to 20 seconds. Each segment is then encoded with a different video bitrate. The ABS algorithm decides on the quality of the segments to be downloaded based on the network's available resources. To ease access to video content on the associated webserver, a Media Presentation Description (MPD) file is created. This MPD file contains general information such as clip length, segment duration, DASH video profile, but more importantly the MPD file contains metrics specific to each of the video representation available. Each representation represents a different quality level determined by video resolution, average encoding bitrate, thus offering an easy mechanism to permit the player adapt video quality for the user. Once the player downloads the MPD file, the ABS algorithm can adjust quality by selecting the most appropriate segment for each video time period.

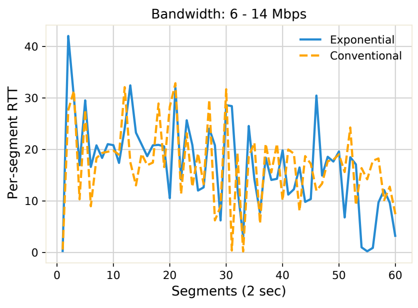
The ABS algorithms are divided into three major categories i) rate-based [19], buffer-based [3] and hybrid-based [17]. In rate-based, a decision is made on the delivery rate of the previously downloaded segments. Buffer-based algorithms monitor the state of the playback buffer, while in hybrid both playback buffer and delivery rate are considered for the choice of the next segment.

Many studies have been carried out to find the key indicators for better video quality such as TCP slow-start [20] and "ON-OFF" status of HAS players [21]. Similarly, Saamer et.al, [22] evaluated two major commercial players for their findings (Smooth Streaming, Netflix) and one open source player (OSMF). Several QoE key factors have been identified such as how long a video streaming player take to converge to maximum bitrate, what happens when two adaptive video players compete for available bandwidth on a bottleneck link. The authors also point out how the adaptive streaming perform with respect to live content. [23] provides a comprehensive comparative study of state-of-art ABS algorithms. Authors have concluded that buffer-based ABS shows better QoE as compared to rate- and hybrid-based algorithms. In another study that evaluated both objective and subjective QoE, but the authors only consider throughput based algorithms [24].

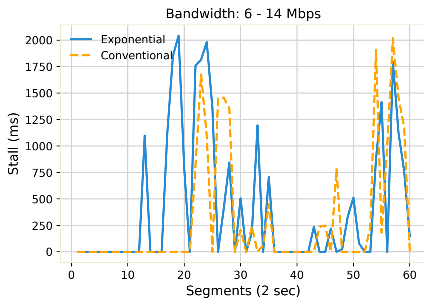
We have found that many studies that exist in the literature lack a comprehensive comparison of HAS algorithms. Also, many ABS algorithms are limited in their functionality as the authors have not released their framework for reproducibility. Additionally, in comparison to previous studies, much attention has been given to QoE evaluation's rather than state-of-art per-segment QoS to QoE mapping. The QoS to QoE mapping is necessary to deliver more evidence-based higher quality video content through understanding how limited network resources can impact quality of experience of end-users. To fill this gap, we provide a flexible framework for analysis of DASH videos considering many combinations of real 5G traces as mentioned in Section I. The framework is equipped with many other options such as the ability to change the video content for streaming, a range of different ABS algorithms to compare QoS to QoE metrics, and a rich set of different evaluations scenarios.

III. PHASE 1 - STREAMING FRAMEWORK

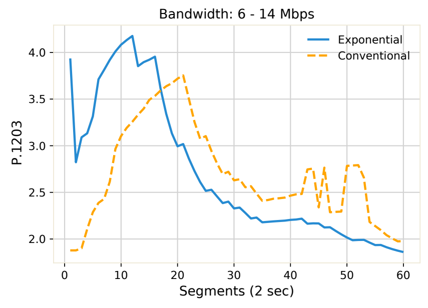
We present a re-producible DASH framework supporting the evaluation of six state-of-art ABS algorithms through the emulation of ten different real 5G traces to stream DASH videos. The provided tools process seven objective Key Performance Indicators (KPIs), five QoE models output (P.1203, Yin, Yu, Duanmu, Clae), and three per-segment QoS features extracted from trace files (RTT, throughput, packets). The



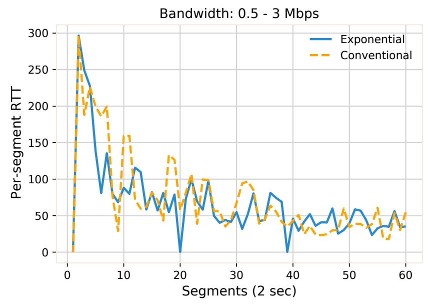
(a) RTT, case a)



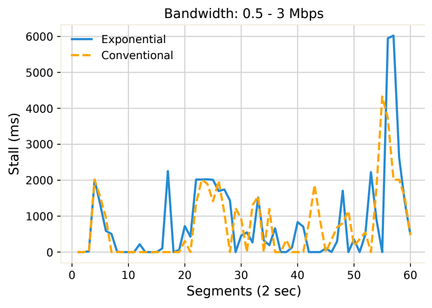
(b) Stall, case a)



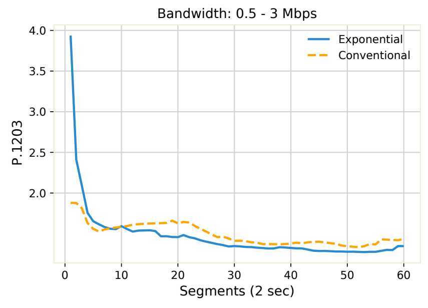
(c) P.1203, case a)



(d) RTT, case b)



(e) Stall, case b)



(f) P.1203, case b)

Fig. 2: Exponential - Conventional: case a) and b); QoS (RTT), stalls, and QoE (P.1203) score per video segment for 60 video segments

TABLE I: *Conventional*: *godash* log file of first 5 video segments, Case *Mobility* (6-14) Mbps

Seg_#	Algorithm	Seg_Dur	Codec	Width	Height	FPS	Play_Pos	RTT	P.1203	Clae	Duanmu	Yin	Yu
1	conventional	2000	H264	320	180	24	0	25.025	1.878	0.000	51.077	-5760.485	0.240
2	conventional	2000	H264	320	180	24	2000	78.83	1.878	0.480	46.477	-11520.970	0.24
3	conventional	2000	H264	384	216	24	4000	12.09	1.9	0.417	46.898	718.545	0.286
4	conventional	2000	H264	512	288	24	6000	16.86	2.106	0.314	47.826	1097.122	0.404
5	conventional	2000	H264	640	360	24	8000	74.93	2.287	0.302	48.77	1863.42	0.54

TABLE II: Processed dataset first 5 video segments of 2s for case (6-14) Mbps using Conventional ABS algorithm

Total_Users	Host	Segment	Stall	Bitrate	Buffer	RTT	Throughput	Packets	P.1203	Clae	Duanmu	Yin	Yu
2	1	1	0	8	2000	0.14	7443037.97	2	1.87	0	51.07	-5760.48	0.24
2	1	2	0	329	4000	27.65	240702.88	30	1.87	0.48	46.47	-11520.97	0.24
2	1	3	0	720	4643	31.39	280181.47	64	1.9	0.41	46.89	718.54	0.28
2	1	4	0	1408	5212	10.33	465851.21	117	2.10	0.31	47.82	1097.12	0.40
2	1	5	0	1191	5277	27.68	325186.14	104	2.28	0.30	48.77	1863.42	0.54

framework encompasses a DASH streaming environment, the pre-processing of network, video client logs and associated scripts. For ease of use, the framework includes a Virtual Machine (VM) [25] with all software and dependencies installed as shown in Figure 1. The VM provides all tools and the environment needed to stream DASH content in a multi-user realistic 5G network. Currently, the VM showcases a single combination of mobility, host competition, and link bandwidth parameters to run the `Mininet-WiFi` emulated topology, collect `godash` log(s), pcap file(s), and process the raw video logs and network data. However, the framework is versatile and can be easily modified to accommodate additional DASH algorithms, 5G traces, etc. The proposed framework provides a convenient mechanism to generate multimedia traffic processed data. Video instructions on the framework's use within the VM are available online [26]. Note that we have released all remaining code [3] used for processing the dataset for reproducibility. The computational scripts and utilities are already available in the VM.

[3]https://github.com/sajibtariq/dashframework

IV. Phase 2 - Dynamic Analysis Demo

In Phase 2, we provide the processed dataset collected by running ten different combinations of real 5G traces across six state-of-art ABS algorithms. We assess the impact of concurrent video streaming clients (1,...,2), (1,...,3); with all of the clients streaming from the same server.

The processed dataset generated in Phase 1 is imported and examined using a `Jupyter` notebook, a well-known Web-based interactive environment for data analyses. For ease of use, the framework integrates the provided VM with `JupyterLab`[4]. For each DASH video-segment, the `Jupyter` notebook analyses the impact of five QoE models (P.1203, Clae, Duanmu, Yin and Yu), seven video client objective KPIs (arrival time (ms), delivery time (ms), stall (ms), delivery rate of network (Kbps), segment size (bytes), bitrate (Kbps) and buffer level (s) after the segment was just downloaded, and three QoS features (derived from packet captures).

Note that the data and the notebook hosted on `Github` are in read-only (static) mode. However, a live `Binder`[5] service is available in our `GitHub` repository [27], allowing interaction with the read-only notebook in an executable dynamic environment. In addition, we provide a video demonstration on how to use the environment [26]. In the video, we showcase how to modify the VM generated `Jupyter` static notebook as an interactive notebook with the Binder service and how to visualize changes in the data.

V. Experimental Use Case

Figure 1 gives an overview of the two phases divided into three stages: Phase 1 (Stage 1): data acquisition from the network interface and `godash` player; Phase 1 (Stage 2): data pre-processing from `godash` and the network and Phase 2 (Stage 3): where we use a `Jupyter` notebook to analyse and visualise the processed dataset, as shown in Figure 3.

We begin in the VM with Phase 1 (Stage 1) - logs generation. In this stage, each experiment is performed using the `Mininet-WiFi` [7] network emulator as shown in Figure 1, using the setup detailed in Figure 3. To emulate the HTTP streaming video, we use a lightweight DASH compatible video streaming tool called `godash` [6] at the host node(s) and `Caddy`, a `WSGI` web server, hosting a popular 2-second segment duration x264 animated video titled *Sintel*[6], sourced from a publicly available 4K DASH video dataset [28].

To simplify dataset generation for the article, we asses the impact of 2 and 3 concurrent clients streaming from the same server. The network bandwidth values are based on the 5G trace parameters [29]. We select ten combinations of Mobility and Static (in Mbps); Mobility — (0.5 - 3), (6 - 14), (38 - 10) and (29 - 10); Static — (0.5 - 6), (8 - 57), (4 - 7.6), (52 - 0.5), (70 - 20) and (72 - 9)[7]. Note that the bandwidth during each experiment is changed in real-time between Switch 1 and Switch 2 link after every 4 seconds as shown in Figure 1 using Linux Traffic Control (TC) and Hierarchical Token Bucket (HTB) [30]. A python script is used to collect per-run pcap by `tcpdump`[8]. Later, python Scapy package is used to get per-segment QoS features from pcap.

Table I illustrates an example of a `godash` log file for a single client in the Mobility (driving) scenario using (6 to 14) Mbps, with each line representing per segment metrics for the conventional ABS algorithm. Detailed information on each feature and ABS algorithms is available in `godash` [31].

In Phase 1 (Stage 2), a Python script is used to fetch per segment QoS metrics (RTT, Throughput and Packets) from the pcap files. We merged the QoS metrics and godash logfiles output as a single CSV dataset (example presented in Table II). The first two columns present network context for each experiment, i.e, (Total_Users, Host) indicated as total users competing for video stream and host number. The next column has Segments followed by three video KPIs (Stall, Bitrate and Buffer level) of each corresponding segment. The QoS features extracted from pcap traces of each segment is indicated as (RTT, Throughput, Packets) and finally, the five QoE models provided by `godash`.

Figure 2 takes information from both Table I and Table II, and depicts the RTT, stalls and P.1203 score per video segment for 60 video segments, i.e., $60 \times 2s = 120s$ or 2 minutes of video, for both the conventional and exponential algorithms in the Mobility (driving) scenario using (6 to 14 Mbps) and (0.5 to 3 Mbps) bandwidth. Note that in the evaluation of the QoS impact on QoE in Figure 2, the length of the video file in the experiment is 2 minutes, 2 hosts competing for video stream considering 5G dynamic cases. The bandwidth combinations we select gradually increases from lower to upper limit and 1st user experience is shown in Figure 2.

It is important to note in Figure 2 that both rate-based algorithms select wrong segments to stream, which causes the stalls to appear frequently and ultimately lowering the QoE as in our case P.1203. We can also observe that Conventional has slightly better QoE when compared to Exponential. The QoS feature (RTT) for 60 segments is similar in both cases (6-14), (0.5-3) Mbps, as presented in Figure 2 (a) and (d). Exponential experiences more peaks of stalls see Figure 2 (b) from (10 to 35) segments ultimately causing P.1203 score to converge to lower values as shown in Figure 2 (c). However, in less performing networks (0.5-3) Mbps, both algorithms experience similar stall ratio, see Figure 2 (e), as well as P.1203 scores with fewer jumps to higher scores, as shown in Figure 2 (f).

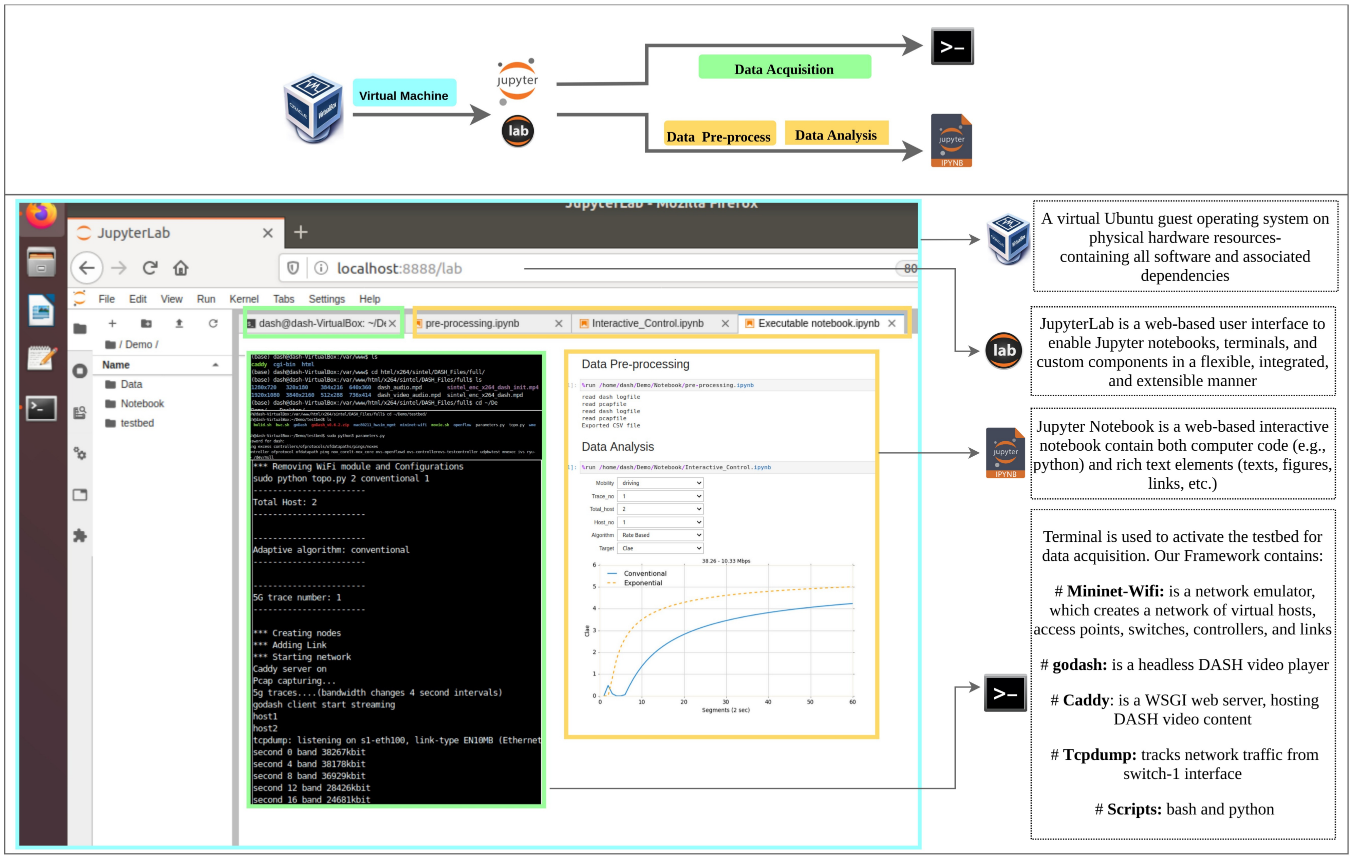
[4]https://jupyterlab.readthedocs.io

[5]https://mybinder.org

[6]http://cs1dev.ucc.ie/misl/4K_non_copyright_dataset/2_sec/x264/sintel/DASH_Files/full/sintel_enc_x264_dash.mpd

[7]10 5G real cases: https://github.com/sajibtariq/dashframework/tree/master/Testbed/5g_traces

[8]https://www.tcpdump.org



```
(base) dash@dash-VirtualBox:/var/www$ ls
caddy  cgi-bin  html
(base) dash@dash-VirtualBox:/var/www$ cd html/x264/sintel/DASH_Files/full/
(base) dash@dash-VirtualBox:/var/www/html/x264/sintel/DASH_Files/full$ ls
1280x720   320x180    384x216  640x360  dash_audio.mpd        sintel_enc_x264_dash_init.mp4
1920x1080  3840x2160  512x288  736x414  dash_video_audio.mpd  sintel_enc_x264_dash.mpd
(base) dash@dash-VirtualBox:/var/www/html/x264/sintel/DASH_Files/full$ cd ~/De
h@dash-VirtualBox:/var/www/html/x264/sintel/DASH_Files/full$ cd ~/Demo/testbed/
h@dash-VirtualBox:~/Demo/testbed$ ls
 bulid.sh  bwc.sh  goDash  goDash_v0.6.2.zip  mac80211_hwsim_mgmt  mininet-wifi  movie.sh  openflow  parameters.py  topo.py  wme

h@dash-VirtualBox:~/Demo/testbed$ sudo python3 parameters.py
sword for dash:
ng excess controllers/ofprotocols/ofdatapaths/pings/noxes
ntroller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openflowd ovs-controllerovs-testcontroller udpbwtest mnexec ivs ryu-
 /dev/null
*** Removing WiFi module and Configurations
sudo python topo.py 2 conventional 1
----------------------
Total Host: 2
----------------------


----------------------
Adaptive algorithm: conventional
----------------------


----------------------
5G trace number: 1
----------------------

*** Creating nodes
*** Adding Link
*** Starting network
Caddy server on
Pcap capturing...
5g traces....(bandwidth changes 4 second intervals)
godash client start streaming
host1
host2
tcpdump: listening on s1-eth100, link-type EN10MB (Ethernet
second 0 band 38267kbit
second 4 band 38178kbit
second 8 band 36929kbit
second 12 band 28426kbit
second 16 band 24681kbit
```

Data Pre-processing

```
%run /home/dash/Demo/Notebook/pre-processing.ipynb
read dash logfile
read pcapfile
read dash logfile
read pcapfile
Exported CSV file
```

Data Analysis

```
%run /home/dash/Demo/Notebook/Interactive_Control.ipynb
```

Mobility: driving
Trace_no: 1
Total_host: 2
Host_no: 1
Algorithm: Rate Based
Target: Clae

A virtual Ubuntu guest operating system on physical hardware resources-containing all software and associated dependencies

JupyterLab is a web-based user interface to enable Jupyter notebooks, terminals, and custom components in a flexible, integrated, and extensible manner

Jupyter Notebook is a web-based interactive notebook contain both computer code (e.g., python) and rich text elements (texts, figures, links, etc.)

Terminal is used to activate the testbed for data acquisition. Our Framework contains:

Mininet-Wifi: is a network emulator, which creates a network of virtual hosts, access points, switches, controllers, and links

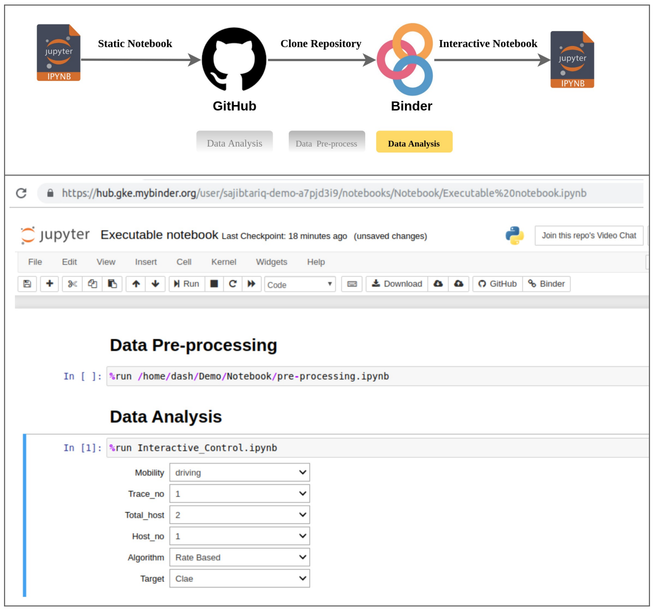
godash: is a headless DASH video player

Caddy: is a WSGI web server, hosting DASH video content

Tcpdump: tracks network traffic from switch-1 interface

Scripts: bash and python

Fig. 3: An Ubuntu 18.04 VM including a DASH streaming environment containing: `Mininet-Wifi`, `Jupyter` lab and notebook, `godash` player, `Caddy` server and DASH content, `tcpdump`, and scripts


Static Notebook
GitHub
Clone Repository
Binder
Interactive Notebook
Data Analysis
Data Pre-process
Data Analysis
https://hub.gke.mybinder.org/user/sajibtariq-demo-a7pjd3i9/notebooks/Notebook/Executable%20notebook.ipynb
jupyter
Executable notebook Last Checkpoint: 18 minutes ago (unsaved changes)
Join this repo's Video Chat
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Run
Code
Download
GitHub
Binder

Data Pre-processing

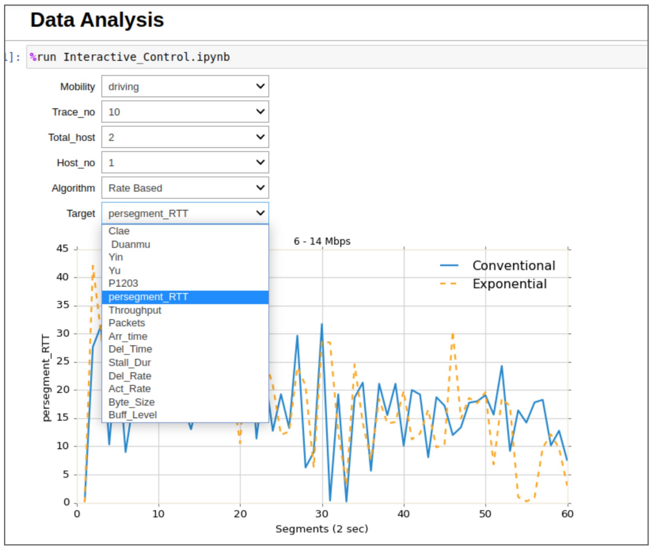
```
In [ ]: %run /home/dash/Demo/Notebook/pre-processing.ipynb
```

Data Analysis

```
In [1]: %run Interactive_Control.ipynb
```

Mobility: driving
Trace_no: 1
Total_host: 2
Host_no: 1
Algorithm: Rate Based
Target: Clae

Fig. 4: `Binder`, turns the `Github` notebook into an interactive notebook in an executable environment for data analysis


Data Analysis

```
]: %run Interactive_Control.ipynb
```

Mobility: driving
Trace_no: 10
Total_host: 2
Host_no: 1
Algorithm: Rate Based
Target: persegment_RTT

Fig. 5: First user experience (mobility) with Conventional and Exponential ABS algorithms over (6-14) Mbps. Per-segment QoS RTT on (y-axis), 60 segments on (x-axis)

Moving now to Phase 2, we use the CSV dataset and create a `Jupyter` notebook. The `Jupyter` notebook and csv dataset are uploaded from the VM to `GitHub` and through a live dynamic `Binder` service, we can interact, analyse and visualise the input dataset. To visualise your own data, the easiest option is to fork our repository [27] and upload your data to the forked version of it. Figure 4 highlights the outline and design of the Binder service, while Figure 5 illustrates some of the features that can be selected to update and revise the output plots.

VI. Conclusions

This work presents a reproducible framework which generates a QoS and QoE metric dataset for DASH experiments using different state-of-art adaptive bitrate streaming algorithms. A convenient and interactive Binder notebook is used to demonstrate live analytical environment processing the dataset output of the framework. We observe the impact of bandwidth variation on the adaptation strategies of each category of ABS algorithm (Rate, Buffer and Hybrid), discerning the relationship between network QoS metrics, video QoE models and DASH streaming KPI values. Future work includes further analysis of the impact of other QoS metrics (e.g., delay and packet loss) on HAS performance and the support of machine learning research to correlate and predict QoE based on the observed QoS features.

Acknowledgment

This research was supported by the Innovation Center, Ericsson S.A., Brazil, grant UNI.67. This document, however, does not necessarily represent Ericsson's official viewpoint. We also acknowledge the support of Science Foundation Ireland under Grant 13/IA/1892 and 13/RC/2077.

References

[1] N. Panwar, S. Sharma, and A. K. Singh, "A survey on 5G: The next generation of mobile communication," *Physical Communication*.

[2] Conviva, "Conviva's state of streaming q1 2020," 2020.

[3] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *ACM SIGCOMM Computer Communication Review*, vol. 44, pp. 187–198, ACM, 2014.

[4] J. Qiao, X. S. Shen, J. W. Mark, and L. Lei, "Video quality provisioning for millimeter wave 5G cellular networks with link outage," *IEEE Transactions on Wireless Communications*, vol. 14, no. 10, pp. 5692–5703, 2015.

[5] M. Ismail and W. Zhuang, "Statistical QoS guarantee for wireless multi-homing video transmission," in *2013 IEEE Global Communications Conference (GLOBECOM)*, pp. 4615–4620, IEEE, 2013.

[6] D. Raca, M. Manifacier, and J. J. Quinlan, "goDASH - GO accelerated HAS framework for rapid prototyping," in *Proceedings of the 12th International Conference on Quality of Multimedia Experience*, 2020.

[7] R. R. Fontes, S. Afzal, S. H. Brito, M. A. Santos, and C. E. Rothenberg, "Mininet-WiFi: Emulating software-defined wireless networks," in *2015 11th International Conference on Network and Service Management (CNSM)*, pp. 384–389, IEEE, 2015.

[8] S. Petrangeli *et al.*, "QoE-Driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming," *ACM Trans. Multimedia Comput. Commun. Appl.*, Oct. 2015.

[9] Z. Duanmu *et al.*, "A quality-of-experience database for adaptive video streaming," *IEEE Transactions on Broadcasting*, vol. 64, pp. 474–487, June 2018.

[10] X. Yin *et al.*, "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," in *2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, 2015.

[11] L. Yu *et al.*, "QoE-Driven Dynamic Adaptive Video Streaming Strategy With Future Information," *IEEE Transactions on Broadcasting*, vol. 63, pp. 523–534, Sep. 2017.

[12] W. Robitza *et al.*, "HTTP Adaptive Streaming QoE Estimation with ITU-T Rec. P. 1203: Open Databases and Software," in *9th ACM Multimedia Systems Conference*, MMSys '18, pp. 466–471, 2018.

[13] A. Raake *et al.*, "A bitstream-based, scalable video-quality model for HTTP adaptive streaming: ITU-T P.1203.1," in *Ninth International Conference on Quality of Multimedia Experience (QoMEX)*, May 2017.

[14] Z. Li *et al.*, "Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale," *IEEE Journal on Selected Areas in Communications*, vol. 32, pp. 719–733, April 2014.

[15] Y. Sani *et al.*, "Modelling Video Rate Evolution in Adaptive Bitrate Selection," in *2015 IEEE International Symposium on Multimedia (ISM)*, pp. 89–94, Dec 2015.

[16] T. Huang *et al.*, "A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, (New York, NY, USA), pp. 187–198, ACM, 2014.

[17] A. H. Zahran *et al.*, "Arbiter+: Adaptive rate-based intelligent http streaming algorithm for mobile networks," *IEEE Transactions on Mobile Computing*.

[18] L. D. Cicco *et al.*, "ELASTIC: A Client-Side Controller for Dynamic Adaptive Streaming over HTTP (DASH)," in *2013 20th International Packet Video Workshop*.

[19] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pp. 97–108, 2012.

[20] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: picking a video streaming rate is hard," in *Proceedings of the 2012 internet measurement conference*, pp. 225–238, 2012.

[21] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What happens when HTTP adaptive streaming players compete for bandwidth?," in *Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video*, pp. 9–14, 2012.

[22] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in *Proceedings of the second annual ACM conference on Multimedia systems*, pp. 157–168, 2011.

[23] T. Karagkioules, C. Concolato, D. Tsilimantos, and S. Valentin, "A comparative case study of HTTP adaptive streaming algorithms in mobile networks," in *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp. 1–6, 2017.

[24] C. Timmerer, M. Maiero, and B. Rainer, "Which adaptation logic? An objective and subjective performance evaluation of HTTP-based adaptive media streaming systems," *arXiv preprint arXiv:1606.00341*, 2016.

[25] P. 1-VM, "Download link." https://drive.google.com/drive/folders/1y4HZ7sYxzCi__yXTpAnZwMQlQy5na04b?usp=sharing, 2020.

[26] Dashframework-video, "Demonstration videos." https://drive.google.com/drive/folders/1JayDnKF8NLneIFj1nc2CLKD7UZ0AnYWM?usp=sharing, 2020.

[27] Dashframework, "Github repository." https://github.com/sajibtariq/dashframework, 2020.

[28] J. J. Quinlan *et al.*, "Multi-profile Ultra High Definition (UHD) AVC and HEVC 4K DASH Datasets," in *9th ACM MMSys Conference*.

[29] D. Raca, D. Leahy, C. J. Sreenan, and J. J. Quinlan, "Beyond throughput, the next generation: a 5g dataset with channel and context metrics," in *Proceedings of the 11th ACM Multimedia Systems Conference*.

[30] D. G. Balan and D. A. Potorac, "Linux htb queuing discipline implementations," in *2009 First International Conference on Networked Digital Technologies*, pp. 122–126, IEEE, 2009.

[31] J. O'Sullivan, D. Raca, and J. J. Quinlan, "godash 2.0-The Next Evolution of HAS Evaluation," in *2020 IEEE 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pp. 185–187, IEEE, 2020.