Automated Capture and Animated Playback of TCP Behaviour During DASH-based Content Delivery

Jonathan Kua, Grenville Armitage, Philip Branch
Centre for Advanced Internet Architectures
Swinburne University of Technology
Melbourne, Australia
{jtkua, garmitage, pbranch}@swin.edu.au

***Abstract*—Transmission Control Protocol (TCP) has been for many decades the dominant transport layer protocol that carries the bulk of all traffic across the Internet. Although TCP was traditionally used for reliable bulk transfers, recently it is also becoming the protocol of choice for streaming applications – Dynamic Adaptive Streaming over HTTP (DASH) has recently emerged as a standard for live and on-demand streaming. Netflix and YouTube employ DASH-like streaming strategies and account for more than 50% of North American traffic in 2015, representing a significant source of Internet traffic. Consumer video streams are most likely to be bottlenecked by last-mile ISP links and impacted by emerging Active Queue Management (AQM) schemes for counteracting bufferbloat. However, the interactions between different TCP algorithms, DASH traffic (within a mix of other typical traffic) and the underlying AQMs are not well understood. Experiments in a controlled testbed allow shedding more light on this issue. We propose demonstrating live DASH experiments and data visualisation with "TCP Experiment Automation Controlled Using Python" (TEACUP) – a software tool for running automated experiments and data analysis in a controlled testbed. TEACUP is used for running DASH-based experiments presented in our paper [1], hence this demonstration will also allow experiment repeatability.**

***Index Terms*—DASH, TCP, AQM, TEACUP, Experiments**

I. Introduction

Transmission Control Protocol (TCP) has been the dominant transport layer protocol that carries the bulk of all traffic across the Internet for many decades. Several TCP congestion control algorithms were developed for performance optimisation over the last few decades. Although TCP was traditionally used for reliable bulk transfers, recently it is also becoming the protocol of choice for multimedia streaming applications – Dynamic Adaptive Streaming over HTTP (DASH) has recently emerged as a standard for live and on-demand streaming [2]. Netflix and YouTube employ DASH-like video streaming strategies [3], [4] and account for more than 50% of North American traffic in 2015 [5], representing a significant source of Internet traffic. Consumer video streams are most likely to be bottlenecked by last-mile ISP links and impacted by emerging Active Queue Management (AQM) schemes for counteracting bufferbloat. However, the interactions between different TCP algorithms, DASH traffic (within a mix of other typical traffic) and the underlying AQMs are not well understood. Experiments in a controlled testbed allow shedding more light on this issue. Hence, we developed "TCP Experiment Automation Controlled Using Python" (TEACUP)[1] – a software tool for running automated experiments and data analysis in a controlled testbed.

Based on a configuration file and utilising Python Fabric[2], TEACUP can perform a series of experiments with different traffic mixes, different bottlenecks , different emulated network paths, and different host settings. For each experiment permutation, TEACUP automatically collects relevant information for post-analysis. More details on the design and implementation of TEACUP-specific testbed can be found in [6]. TEACUP also provides a number of native data analysis tasks [7] and data visualisation functions with TEAPLOT [8].

We propose demonstrating running live DASH experiments, data analysis and visualisation of experiment results with TEACUP and TEAPLOT. TEACUP is used for running and analysing our experiments presented in [1], hence this demonstration will allow for experiment repeatability.

This proposal is organised as follows: Section II describes a brief overview of TEACUP testbed network model, design and requirements. In Section III, we outline the extensions to TEACUP for DASH support. Section IV describes our proposed experiment demonstrations and equipment requirements. We offer concluding remarks in Section V.

II. TEACUP Requirements and Design

In this section, we briefly describe the design requirements of TEACUP. For more technical configuration details and advanced usage information, refer to [6], [7], [9].

A. Testbed topology

TEACUP runs experiments in a controlled testbed environment as shown in Figure 1, where one bottleneck router interconnects two experiment networks. The *experiment networks* contain hosts that can act as traffic sources and destinations. The router, all hosts and TEACUP (on a control server) are also connected to a separate *control network*. The control server runs a DHCP+TFTP server for the PXE boot setup [6]. TEACUP configures hosts before each experiment and collects

[1]The TEACUP project originated at Centre for Advanced Internet Architectures (http://caia.swin.edu.au/tools/teacup), Swinburne University of Technology, and from version 1.0 the source code is freely available on SourceForge at http://sourceforge.net/projects/teacup

[2]http://www.fabfile.org/

data from hosts after each experiment. It can also automated the assignment of hosts to either experiment network with an associated VLAN and act as a NAT gateway to enable all testbed hosts to access the public Internet if necessary.

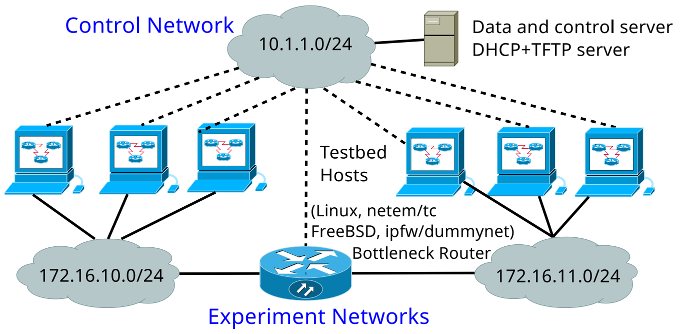

Fig. 1. TEACUP testbed overview

B. *Hosts operating systems and TCP algorithms*

TEACUP supports scenarios where sources and/or destinations are FreeBSD (NewReno, CUBIC, CAIA Delay Gradient, Hamilton Delay), Linux (NewReno, CUBIC),Windows (Compound), or Mac OS X (NewReno).

C. *Path characteristics and bottleneck AQMs*

The Linux or FreeBSD bottleneck router emulates network path characteristics by rate-shaping bandwidth limits, applying one-way delay (OWD) and packet loss rates in either direction. These conditions are applied bidirectionally, using separate delay and rate shaping stages for traffic in each direction. Hence, the path's intrinsic Round Trip Time (RTT) is 2*OWD. AQM schemes are implemented at the router with configurable buffer size. Under Linux router (netem/tc), FIFO, RED, CoDel [10], PIE [11], FQ-CoDel [12] are supported. In addition to all the AQMs supported under Linux, the FreeBSD router (Dummynet AQM v0.2.1 [13] patched), also supports FQ-PIE.

D. *Traffic generators and statistics logging*

TEACUP supports TCP bulk transfer, UDP flows (VoIP, games-like), and data centre 'incast' query-response patterns, HTTP streaming (DASH-like) traffic. TCP connection statistics are logged using SIFTR[3] (FreeBSD) and Web10G[4] (Linux). Traffic is captured using tcpdump on all interfaces and processed by Synthetic Packet Pairs (SPP) [14] to construct per-packet network-layer RTT measurements.

III. TEACUP Extensions for DASH support

In this section, we describe our extensions to TEACUP for running and analysing DASH experiments.

[3]http://caia.swin.edu.edu.au/urp/newtcp/tools.html
[4]http://www.web10g.org/

A. *Generating DASH flows*

The TEACUP DASH traffic generator starts dash.js v2.0.0[5] in Chromium[6] and Xorg[7], and starts requesting video chunks from the lighttpd[8] server (with persistent HTTP connections) that hosts the DASH dataset.

1) DASH server and dataset: DASH dataset is hosted on a regular lighttpd HTTP server in the testbed hosts. We utilise the dataset made available by ITEC-DASH [15]. The dataset[9] comprise of full-length sequences at different representations in terms of bitrates and resolutions. It is encoded and multiplexed using different chunk sizes and are made available with corresponding Media Presentation Description (MPD) files.

2) DASH client: We integrated the BSD-3 licensed dash.js DASH client into our testbed. dash.js is an initiative of the DASH Industry Forum[10] to establish a production quality framework for building multimedia players for MPEG-DASH content playback using client-side JavaScript libraries.

B. *Analysing DASH achieved rates and representation rates*

Achieved rates (AR)[11] and representation rates (RR)[12] are two important metrics for measuring DASH performance. DASH clients implement an adaptive bitrate algorithm to retrieve chunks encoded at the highest RR sustainable by recently observed network conditions. When AR is high, the client will seek to retrieve future chunks encoded at a higher RR to provide better quality of experience. When AR is low, it will seek to retrieve future chunks encoded at a lower RR to avoid playout buffer under-run. We extend TEACUP's analysis tasks to calculate ARs by using the payload lengths extracted from HTTP response headers and the chunk transfer time. RRs are extracted by parsing the client's HTTP GET requests.

IV. Experiment Demonstration

We would like to demonstrate running live DASH experiments, analysing and animating the experimental results with TEACUP. Due to logistics reasons, we will replicate our physical testbed in a virtualised environment.

A. *Technical details and configuration options*

- **Router and hosts:** The bottleneck router uses either FreeBSD or openSUSE Linux to provide a configurable bottleneck between clients and servers. End-hosts run either FreeBSD NewReno or openSUSE Linux CUBIC.

[5]http://dashif.org/reference/players/javascript/v2.0.0/samples/dash-if-reference-player/index.html
[6]https://www.chromium.org/
[7]https://www.x.org/
[8]https://www.lighttpd.net/
[9]The BigBuckBunny video content is an open-source animation video, with source quality of 1080p, 9mins 46secs long by the Blender Institute. It can be downloaded from http://www-itec.uni-klu.ac.at/ftp/datasets/DASHDataset2014/BigBuckBunny
[10]http://dashif.org/
[11]Chunk size divided by the time taken to receive it.
[12]The pre-encoded video bitrates available on the server.

- **Path characteristics:** The FreeBSD or Linux router provides rate shaping and apply specific OWD to emulate bottleneck with a specific rate and delay. We use consumer grade rates with domestic delays.
- **AQMs:** Therouter implements either traditional FIFO, Linux (CoDel, PIE and FQ-CoDel) or FreeBSD (CoDel, PIE, FQ-CoDel, and FQ-PIE) AQMs for managing the bottleneck queue. The total buffer size is set to 1000 packets for AQMs. We use AQMs with default settings (recommended by IETF Internet Drafts [11], [12], [10]).
- **DASH:** DASH clients have the option of retrieving different chunk sizes (2, 4, 10, 15 secs). Both chunk-based and byte-range requests are supported.

B. Live demonstration

Our live demonstration encompasses three main phases:

1) Experiment runs: First we set the desired DASH options, interfering traffic and emulated network conditions in a configuration file before running the experiment. The audience will be able to watch on screen how TEACUP manages the experiment setup/tear-down, and also observe how it automatically starts, runs and stops the DASH video streaming session.

2) Data analysis: We demonstrate using TEACUP's analysis tasks to analyse various TCP (throughput, congestion window, RTT, ACK sequence) and DASH (AR, RR) metrics and show the generated PDF graphs (single/comparison plots).

3) Results animation and visualisation: We further demonstrate using TEAPLOT [8] – a browser-based visualisation engine that allows 2D variable speed replay of logged TCP and system parameters over time from data logged during previous TEACUP experiments. We will use it to replay TCP and system metrics of the DASH experiments. This visual aspect will be the centre piece of our results analysis phase.

Poster display: The demonstration will also consist of an accompanying poster summarising the intended application of TEACUP for running and analysing DASH experiments; and describing TEAPLOT for data animation and visualisation.

Equipment requirements: In order to run the demonstration, we will require a table with power outlets and a monitor (with associated cabling) for larger display. We will provide the virtual machines and computing equipment (laptop).

V. Conclusions

With the resurgence of AQM schemes and the dominance of DASH-based multimedia traffic, it has become increasingly important to understand the intertwined interactions between DASH, TCP and AQMs. Hence, we extended TEACUP to allow us to run and analyse DASH experiments.

In this document, we propose demonstrating the functionality of TEACUP in running automated DASH experiments over various emulated network paths in a virtualised environment. We will also show its data analysis and animation capabilities.

This demonstration will allow us to showcase TEACUP to the academic research community; as an invaluable experimental research tool for running automated and repeatable experiments over a wide range of emulated network conditions and system settings in a controlled testbed environment. This demonstration will also allow for experiment repeatability of results presented in [1].

References

[1] J. Kua, G. Armitage, and P. Branch, "The Impact of Active Queue Management on DASH-based Content Delivery (accepted to appear)," in *41st Annual IEEE Conference on Local Computer Networks (LCN 2016)*, Dubai, United Arab Emirates (UAE), Nov. 2016.

[2] "Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats," ISO, 2012, iSO/IEC 23009-1:2012. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=57623.

[3] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao, "Youtube everywhere: Impact of device and infrastructure synergies on user experience," in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC '11, 2011, pp. 345–360.

[4] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, and W. Dabbous, "Network characteristics of video streaming traffic," in *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '11, 2011, pp. 25:1–25:12.

[5] Sandvine, "Sandvine Global Internet Phenomena Report," https://www.sandvine.com/downloads/general/global-internet-phenomena/2015/global-internet-phenomena-report-latin-america-and-north-america.pdf, 2015 (accessed May 2016).

[6] S. Zander, G. Armitage, "CAIA Testbed for TCP Experiments Version 2," Centre for Advanced Internet Architectures, Swinburne University of Technology, Tech. Rep. 150210C, 2015. [Online]. Available: {http://caia.swin.edu.au/reports/150210C/CAIA-TR-150210C.pdf}

[7] ——, "TEACUP v1.0 – Data Analysis Functions," Centre for Advanced Internet Architectures, Swinburne University of Technology, Tech. Rep. 150529B, 2015. [Online]. Available: {http://caia.swin.edu.au/reports/150529B/CAIA-TR-150529B.pdf}

[8] I. True, G. Armitage, and P. Branch, "Teaplot v0.1: A browser-based 3D engine for animating TEACUP experiment data," Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. 150828A, 28 August 2015. [Online]. Available: http://caia.swin.edu.au/reports/150828A/CAIA-TR-150828A.pdf

[9] S. Zander and G. Armitage, "TEACUP v1.0 - A System for Automated TCP Testbed Experiments," http://caia.swin.edu.au/reports/150529A/CAIA-TR-150529A.pdf, Melbourne, Australia, 29 May 2015.

[10] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Controlled Delay Active Queue Management," IETF Draft, draft-ietf-aqm-codel-03, March 2016. [Online]. Available: https://tools.ietf.org/html/draft-ietf-aqm-codel-03

[11] R. Pan, P. Natarajan, F. Baker, G. White, B. VerSteeg, M. Prabhu, C. Piglione, and V. Subramanian, "PIE: A Lightweight Control Scheme To Address the Bufferbloat Problem," IETF Draft, draft-ietf-aqm-pie-06, April 2016. [Online]. Available: https://tools.ietf.org/html/draft-ietf-aqm-pie-06

[12] T. Høeiland-Jøergensen, P. McKenney, D. Taht, J. Gettys, and E. Dumazet, "FlowQueue-Codel," IETF Draft, draft-ietf-aqm-fq-codel-06, March 2016. [Online]. Available: https://tools.ietf.org/html/draft-ietf-aqm-fq-codel-06

[13] R. Al-Saadi and G. Armitage, "Dummynet AQM v0.2 – CoDel, FQ-CoDel, PIE and FQ-PIE for FreeBSD's ipfw/dummynet framework," Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. 160418A, 18 April 2016. [Online]. Available: http://caia.swin.edu.au/reports/160418A/CAIA-TR-160418A.pdf

[14] S. Zander and G. Armitage, "Minimally-Intrusive Frequent Round Trip Time Measurements Using Synthetic Packet Pairs," in *The 38th IEEE Conference on Local Computer Networks (LCN 2013)*, October 2013.

[15] S. Lederer, C. Müller, and C. Timmerer, "Dynamic Adaptive Streaming over HTTP Dataset," in *Proceedings of the 3rd Multimedia Systems Conference*, ser. MMSys '12. New York, NY, USA: ACM, 2012, pp. 89–94. [Online]. Available: http://doi.acm.org/10.1145/2155555.2155570