Volume: 10 Issue: 03 Pages: 3898-3904 (2018) ISSN: 0975-0290

What happens when adaptive video streaming players compete with Long-Lived TCP flows?

Koffka Khan
Department of Computing and Information Technology The University of the West Indies, Trinidad and Tobago, W.I
Email: koffka.khan@gmail.com
Wayne Goodridge
Department of Computing and Information Technology The University of the West Indies, Trinidad and Tobago, W.I
Email: wayne.goodridge@sta.uwi.edu.com

ABSTRACT

Competition among adaptive video streaming players severely diminishes user-QoE. When players compete at a bottleneck link many do not obtain adequate resources. This imbalance eventually causes ill effects such as screen flickering and video stalling. This scenario worsens when Long-lived TCP flows compete with the video flows. It is a known fact that adaptive video players perform poorly in the presence of Long-lived TCP flows. This work evaluates current heuristic adaptive video players at a bottleneck link in the presence of Long-lived TCP flows. Experimental setup includes the TAPAS player and emulated network conditions. The results show ELASTIC outperforms PANDA, FESTIVE and the Conventional players.

Keywords — **adaptive video streaming, bottleneck, flickering, stalling, TAPAS, ELASTIC, PANDA, FESTIVE**

Date of Submission: June 14, 2018 Date of Acceptance: Sep 28, 2018

I. INTRODUCTION

Adaptive video players are not able to get a fair share when coexisting with a TCP greedy flow [1]. TCP long-lived flows completely shut off TCP short-lived flows [21], [19], [29], and [34]. This causes performance problems for TCP short-lived flows, which generally carry interactive or delay sensitive data, such as video data/flows. TCP short-lived flows are becoming increasingly dominant in Internet traffic. This, together with competition from TCP long-lived flows, makes it an important area for adaptive video players. The outcome is the "downward spiral" effect. ON-OFF traffic patterns are the main contributor. Remedies include increasing segment size and filtering of bandwidth estimates. Examples of TCP long-lived connections are Internet chat and messaging (MSN, Skype). Device-to-device communication utilizes frequent "keepalive" messages. Devices transmit these messages periodically. Issues, such as, over-consumed network resources result. In addition, other issues occur during a TCP long-lived connection. These include TCP congestion and TCP connection recovery. Traffic features of TCP long-lived flows are specific to applications usage and their resulting characteristics.

TCP short-lived flows spend most of their time in the slow start phase. In this phase, the congestion window increases at a seemingly exponential rate. In contrast, TCP long-lived flows spend most of their time in the congestion avoidance phase. This phase utilizes the Additive Increase Multiplicative Decrease (AIMD) congestion control strategy (see Figure 1). According to [10] bandwidth sharing: TCP long-lived flows are shown to hurt TCP short-lived flows in terms of end-to-end delay and consequently throughput.

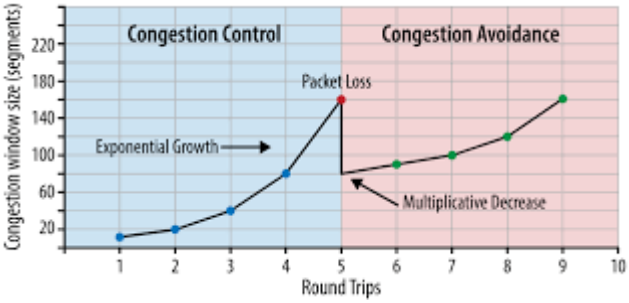

Figure 1: Congestion control and congestion avoidance. [source: https://hpbn.co/building-blocks-of-tcp/]

TCP long-lived flows occupy most of the buffer space from the sender's end point. It creates huge queuing delays for TCP short-lived flows. Consequently, TCP short-lived flows only send few packets. TCP short-lived flows share many features with TCP long-lived flows, such as self-clocking, backing off and going to time out. However, TCP short-lived flows must try to utilize as much bandwidth as possible, when coexisting with TCP long-lived flows.

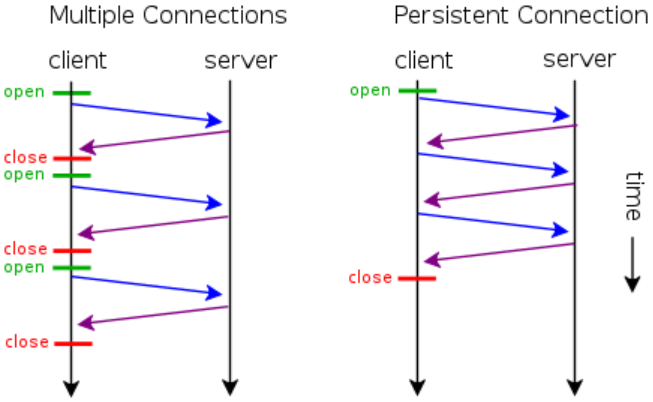

Figure 2: Multiple versus persistent connection.

Volume: 10 Issue: 03 Pages: 3898-3904 (2018) ISSN: 0975-0290

We now consider an initial scenario setting, where three TCP long-lived flows pass through a bottleneck link. TCP long-lived flows are persistent connections (see Figure 2). The TCP long-lived flows are in the slow start phase, but quickly switches to the congestion avoidance phase and performs AIMD. The maximum congestion window is limited by the bottleneck link capacity. The sender now transmits multiple TCP short-lived flows. Congestion affects TCP short-lived flows since their window evolution is subject to TCP slow start rules. Thus, the TCP short-lived flows enter their slow start phases. Their congestion windows grow exponentially. However, before congestion is met, devices time out or terminate their flows. However, for TCP long-lived flows to time out, the overall throughput of all flows (TCP short-lived and long-lived) must exceed the bottleneck link capacity. Thus, the timing out of TCP-long lived flows occur when many packets are lost from the corresponding window of data [10].

In some cases, TCP is not able to fully utilize the transport or network layer resources. This happens because applications do not produce data fast enough. They produce small amounts of data at a relatively constant rate. This results in small bursts of packets. In extreme cases, applications produce single packets less than the maximum segment size of the connection. A typical example is the Skype [9] live streaming applications. Skype transfers data over TCP at a constant rate of 32 Kbit/s. Also falling in this category are applications utilizing permanent TCP connections and sending keep-alive packets (see Figure 3) during inactive periods. An example is Bit Torrent [3], [35], [14], which exhibits this behavior during choke periods.

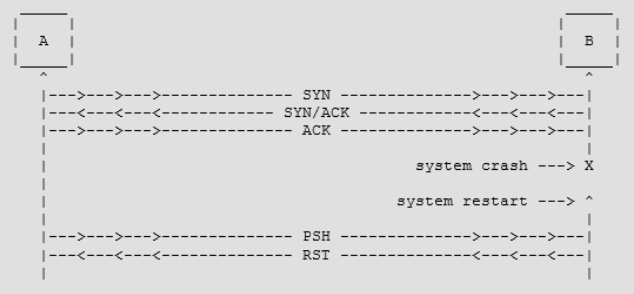

Figure 3: TCP keepalive.

In some cases, applications produce bursts of data, which become separate from each other by idle periods. Web browsing with persistent HTTP connections exhibits such behavioral characteristics. The user clicks on a link to load a web page. This causes a transfer period. He/she then, reads the page. This causes another idle period. He/she then, clicks on another link. This causes, yet another transfer period etc… These intermittent data traffic competes with video flows for bandwidth. Finally, in comparison to co-existing TCP flows, UDP flows utilize more than their fair share of the bandwidth [10] (see Figure 4).

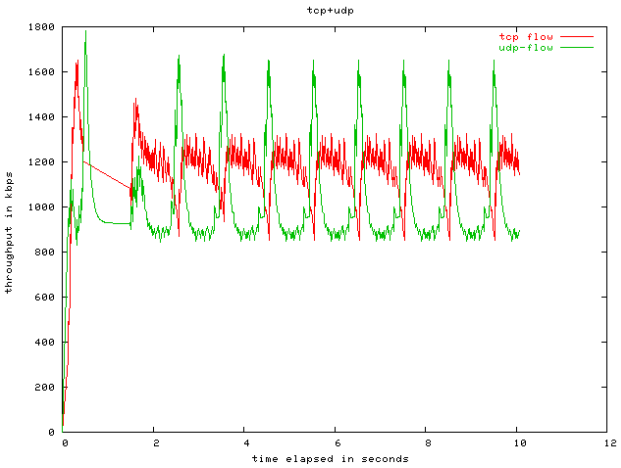

Figure 4: TCP versus UDP flow. Competition is unavoidable. When TCP backs off, UDP takes advantage and tries to control the available bandwidth; a peak result then. But immediately TCP tries to test network resources increasing its bandwidth consumption and UDP cedes and enters in a valley. This cycle repeats again and again. [source: http://web.opalsoft.net/qos/default.php?p=flows-05]

The rest of paper is organized as follows. Section 2 explores different heuristic adaptive streaming approaches including FESTIVE, PANDA and ELASTIC players. In Section 3, we look at the experimental setup of the emulations illustrated in this paper. In section 4 we give the results. Finally, we give our conclusion in section 5.

II. LITERATURE REVIEW

Chunk scheduling with stateless bitrate selection causes feedback loops, bad bandwidth estimation, bitrate switches and unfair bitrate choices [16], [28]. This paper, which portrays the FESTIVE control algorithm [16], confirms that numerous problems occur when multiple bitrate-adaptive players (adaptation over HTTP) share a bottleneck link [39]. It uncovers the fact that the feedback signal the player receives is not a true reflection of the network state because of overlaying the adaptation logic over several layers. HTTP-based video delivery issues are elucidated: (1) the granularity of the control decisions, (2) the timescales of adaptation, (3) the nature of feedback from the network and (4) the interactions with other independent control loops in lower layers of the networking stack.

FESTIVE uses an abstract player model to analyze commercial players: (1) schedule a video chunk for download, (2) select bitrate for chunk, and (3) estimate bandwidth (see Figure 5). It identifies root causes of undesirable interactions with abstract model player framework and saw the need to guide the tradeoffs between stability, fairness and efficiency. In consequence the authors created a robust video adaptation algorithm, which tried to achieve: (1) Fairness – equal allocation of network resources, (2) Efficiency – get highest bitrates for maximum user experience, and (3) Stability – avoid needless bitrate switches. The eventual contributions were a family of adaptation algorithms using the following approaches: (1) Randomized chunk scheduling: to avoid

Volume: 10 Issue: 03 Pages: 3898-3904 (2018) ISSN: 0975-0290

sync biases in network state sampling, (2) Stateful bitrate selection: to compensate between biased bitrate and estimated bandwidth interaction, (3) Delayed update: to account for stability and efficiency tradeoff, and (4) Bandwidth estimator: to increase robustness to outliers.

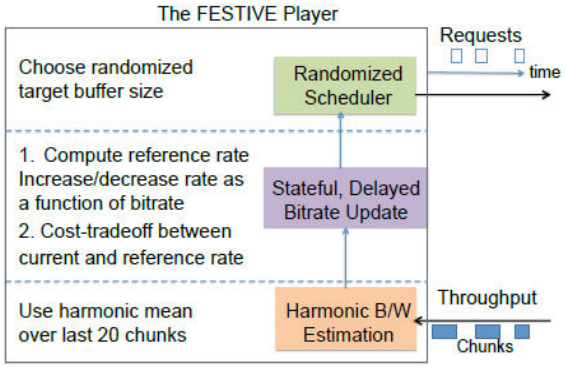

Figure 5: Overview of the FESTIVE adaptive video player. [16]

The authors in [20], who proposed the PANDA algorithm, noted that since TCP throughput observed by a client would indicate the available network bandwidth, it could be used as a reliable reference for video bitrate selection. However, this is no longer true when HTTP Adaptive Streaming (HAS) [4] becomes a substantial fraction of the total network traffic or when multiple HAS clients compete at a network bottleneck. It was observed that the discrete nature of the video bitrates results in difficulty for a client to correctly perceive its fair-share bandwidth. Hence, this fundamental limitation would lead to video bitrate oscillation and other undesirable behaviors that negatively impact the video viewing experience. They offered a design at the application layer using a "probe and adapt" principle for video bitrate adaptation (where "probe" refers to trial increment of the data rate, instead of sending auxiliary piggybacking traffic), which is akin, but also orthogonal to the transport-layer TCP congestion control.

At the beginning of each downloading step n:

1) Estimate the bandwidth share $\hat{x}[n]$ by

$$\frac{\hat{x}[n]-\hat{x}[n-1]}{T[n-1]}=\kappa\cdot(w-\max(0,\hat{x}[n-1]-\tilde{x}[n-1]+w)).$$

2) Smooth out $\hat{x}[n]$ to produce filtered version $\hat{y}[n]$ by

$$\hat{y}[n]=S(\{\hat{x}[m]:m\le n\}).$$

3) Quantize $\hat{y}[n]$ to the discrete video bitrate $r[n]\in\mathcal{R}$ by

$$r[n]=Q\left(\hat{y}[n];...\right).$$

4) Schedule the next download request via

$$\hat{T}[n]=\frac{r[n]\cdot\tau}{\hat{y}[n]}+\beta\cdot(B[n-1]-B_{\min}).$$

Figure 6: PANDA four-step model. [20]

The authors illustrate a four-step model (see Figure 6) for an HAS rate adaptation algorithm: (1) Estimating: the algorithm starts by estimating the network bandwidth that can legitimately be used, (2) Smoothing: is then noise-filtered to yield the smoothed version, with the aim of removing outliers, (3) Quantizing: the continuous is then mapped to the discrete video bitrate, possibly with the help of side information such as client buffer size [36], [12], [23] etc… (cf. Figure 7) and (4) Scheduling: the algorithm selects the target interval until the next download request. The advantages of PANDA are as follows. Firstly, as the bandwidth estimation by probing is quite accurate, one does not need to apply strong smoothing. Secondly, since after a bandwidth drop, the video bitrate reduction is made proportional to the TCP throughput reduction, PANDA is very agile to bandwidth drops.

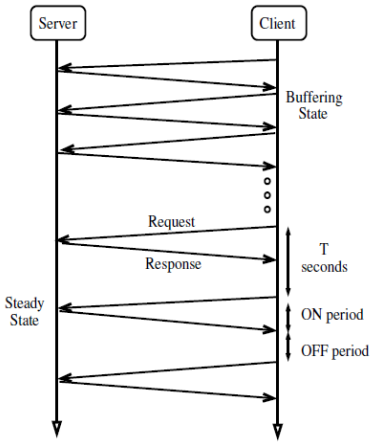

Figure 7: The request-response timing between client and server in the Buffering and Steady states. [1]

1: On segment download:
2: $\Delta T \leftarrow getDownloadTime()$
3: $S \leftarrow getSegmentSize()$
4: $d \leftarrow isPlaying()$
5: $q \leftarrow getQueueLength()$
6: $r \leftarrow h(S/\Delta T)$
7: $q_I \leftarrow q_I + \Delta T \cdot (q - q_T)$
8: **return** $Quantize(r/(d - k_p q - k_i q_I))$

Figure 8: ELASTIC Controller Pseudo-code. [7]

ELASTIC [7] proposes an approach (cf. Figure 8) that designs one controller that throttles the video level (t) to drive the playout buffer length (t) to a set-point q_T. This eliminates the ON-OFF traffic pattern. The player is always in ON phase unless (t) is the highest level and q > Q_{max} (>q_T). The basic concept is based on the playout buffer model, design a feedback control system that computes l(t) to steer q(t) to a threshold q_T. the received rate r(t), is considered as a (measurable) disturbance since it cannot be manipulated. ELASTIC provide a received video rate that oscillates around the fair share, with an increased number of video level switches. However, the main result involved long-lived TCP flows [37], where experimental evaluation showed that ELASTIC is able to get the fair share when competing with TCP long-lived flows.

Volume: 10 Issue: 03 Pages: 3898-3904 (2018) ISSN: 0975-0290

III. EXPERIMENTAL SETUP

The Controller code was written in python. TAPAS [8] an open-source Tool for rApid Prototyping of Adaptive Streaming control algorithms. TAPAS is a flexible and extensible video streaming client written in python that allows researchers to easily design and carry out experimental performance evaluations of adaptive streaming controllers without needing to write the code to download video segments, parse manifest files, and decode the video. TAPAS have been designed to minimize the CPU and memory footprint so that experiments involving a large number of concurrent video flows can be carried out. The player logs experimental data results. The TAPAS player communicates with the video server in the form of a GET request.

A virtual network is setup on the same host machine creating a custom emulation framework (see Figure 9). Our setup consists of client players, video servers, and a bottleneck link. The server resides on a Windows 10 machine. All experiments are performed on a Windows 10 client with an Intel(R) Core(TM)i7-5500U CPU 2.40GHz processor, 16.00 GB physical memory, and an Intel(R) HD Graphics processor. It serves video data to the client(s) who are on a Ubuntu operating system hosted on VMware. The virtual machine is allocated 12GB of physical memory. TAPAS is installed on Ubuntu 15.04 Linux. The TAPAS Adaptive Video Controller client makes different video segment bitrate level requests to the Apache server. TAPAS allow multiple instances of the player to be created enabling multi-client scenarios. This work involves the interaction between adaptive streaming algorithm at the controller and TAPAS players. All traffic between clients and servers go through the bottleneck, which uses VMware settings which allow bandwidth limits to be set during the experiment. TAPAS support both the HTTP Live Streaming (HLS) and Dynamic Adaptive Streaming over HTTP (DASH) format.

The ten-minute-long MPEG-DASH video sequence "Elephant's Dream"[1] is encoded at twenty different bitrates, between 46 Kbps to 4200Kbps and five different resolutions, between 320x240 to 1920x1080, is used to run the experiments (cf. Table II). The video is encoded at 24 frames per second (fps) using the AVC1 codec. Fragment duration of 2s is used and is recorded in the mpd playlist accordingly. All the DASH files (.m4s fragments and .mpd playlists) are placed on the Apache server. We implemented three client-side algorithms in the TAPAS controller. The conventional approach is present by default and is used as a baseline in which to compare against other algorithms. TAPAS is lightweight in built, thus allowing the same receiving host to run a large number of separate video player instances at the same time at different command line interfaces. Thus, it allows the multi-client scenarios which are essential to the work in this paper.

The experiment considers a bottleneck link with two total video connections. The available bandwidth is set to b = 10Mbps for the two player experiments. QoE metrics are described as follows:

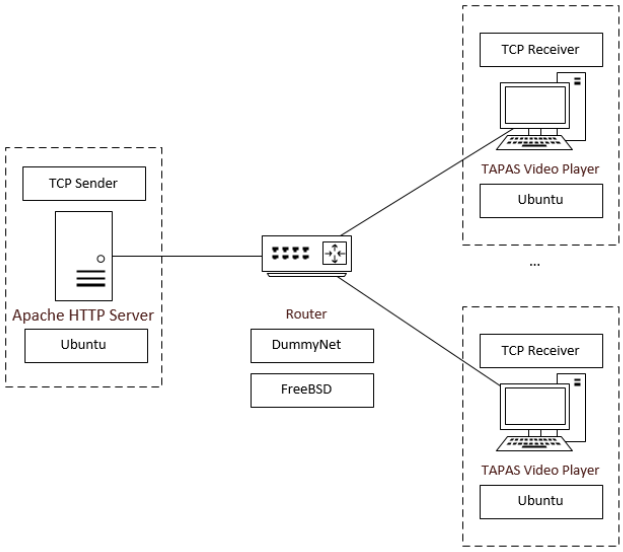

Figure 9: Network testbed setup.

i. The unfairness metric (for two players) is the average of the absolute bitrate difference between the corresponding chunks requested by each player (cf. Equation below, where p1 and p2 are player 1 and player 2, respectfully). The bitrate is the number of bits required to encode one second of playback.

$$Unfairness = Average(\sum_{i=0}^{n-1} |r_{i,p1} - r_{i,p2)}|) \quad (5)$$

ii. The utilization metric is defined as the aggregate throughput during an experiment divided by the available bandwidth in that experiment (cf. Equation below, where tp_i is the throughput at time i and bw is the experimental available bandwidth).

$$Utilization = \frac{\sum_{i=0}^{n-1} tp_i}{bw} \quad (6)$$

In the experiment (E2) the instability, inefficiency, and unfairness (different formulae used for the multi-player scenario) metrics, and re-buffering ratios is used to compare the performances of the considered algorithms.

i. Instability: The instability for player i at time t is given in Equation below, where $w(d) = k - d$ is a weight function that puts more weight on more recent samples. k is selected as 20 seconds.

$$Instability = \frac{\sum_{d=0}^{k-1} |r_{i,t-d} - r_{i,t-d-1}| * w(d)}{\sum_{d=0}^{k-1} r_{i,t-d} * w(d)} \quad (7)$$

ii. Inefficiency: The inefficiency at time t is given in Equation below. Consider N players sharing a bottleneck link with bandwidth, w, with each player x, playing a bit rate, $b_{x,t}$, at time t. A value

Volume: 10 Issue: 03 Pages: 3898-3904 (2018) ISSN: 0975-0290

close to zero implies that the players in aggregate are using as high an average bitrate as possible to improve user experience.

$$Inefficiency = \left| \frac{\sum_x b_{x,t-w}}{w} \right| \quad (8)$$

iii. Unfairness: Let $JainFair_t$ be the Jain fairness index (cf. Equation below) calculated on the average received rates, r_i, (cf. Equation below) at time t over all players. The unfairness at time t is defined as $\sqrt{1 - JainFair_t}$. A lower value implies a fairer allocation.

$$r_i = \frac{downloaded\ bytes}{time\ interval} \quad (9)$$

$$JFI = \frac{(\sum_{i=1}^{n} r_i)^2}{n \sum_{i=1}^{n} r_i^2} \quad (10)$$

iv. Re-buffering ratio: is the ratio of the time spent in re-buffering and the total playtime of the stream Equation below.

$$Re-buffering\ ratio = \frac{total\ re-buffering\ time}{experiment\ duration} \quad (11)$$

IV. RESULTS

We first present the level curves which represent the incoming bitrates of players, see Tables 1, 2 and 3. We observe ELASTIC outperforms the other players in all three experiments. This is because ELASTIC has only ON periods (no OFF periods are present) which enables it to aggressively compete for bandwidth against Long-lived TCP flows.

Table 1: Long-lived TCP flows occupying 1/3 of the bottleneck bandwidth capacity.

	FESTIVE	ELASTIC	PANDA	Conventional
Utilization	0.85	**0.87**	0.86	0.78
Unfairness	0.027	**0.020**	0.025	0.038
Re-buffering ratio	0.039	**0.035**	0.037	0.083
Instability	0.029	**0.024**	0.027	0.070
Average Quality	4.13	**4.17**	4.15	3.70

Table 2: Long-lived TCP flows occupying 1/2 of the bottleneck bandwidth capacity.

	FESTIVE	ELASTIC	PANDA	Conventional
Utilization	0.76	**0.81**	0.77	0.069
Unfairness	0.069	0.065	**0.053**	0.089
Re-buffering ratio	0.062	**0.037**	0.054	0.083
Instability	0.090	0.085	**0.056**	0.185
Average Quality	3.60	**3.85**	3.61	2.58

Table 3: Long-lived TCP flows occupying 2/3 of the bottleneck bandwidth capacity.

	FESTIVE	ELASTIC	PANDA	Conventional
Utilization	0.58	**0.66**	0.63	0.069
Unfairness	0.098	**0.074**	0.089	0.098
Re-buffering ratio	0.087	**0.070**	0.082	0.114
Instability	0.143	0.140	**0.097**	0.194
Average Quality	2.04	**2.37**	2.18	1.94

V. CONCLUSION

Competition among adaptive video streaming players severely diminishes user-QoE. When players compete at a bottleneck link many do not obtain adequate resources. This imbalance eventually causes ill effects such as screen flickering and video stalling. This scenario worsens when Long-lived TCP flows compete with the video flows. It is a known fact that adaptive video players perform poorly in the presence of Long-lived TCP flows. This work evaluates current heuristic adaptive video players at a bottleneck link in the presence of Long-lived TCP flows. Experimental setup includes the TAPAS player and emulated network conditions. The results show ELASTIC outperforms PANDA, FESTIVE and the Conventional players.

REFERENCES

[1] Akhshabi, Saamer, Lakshmi Anantakrishnan, Ali C. Begen, and Constantine 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. ACM, 2012.

[2] Akhshabi, Saamer, Lakshmi Anantakrishnan, Constantine Dovrolis, and Ali C. Begen. "Server-based traffic shaping for stabilizing oscillating adaptive streaming players." In Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, pp. 19-24. ACM, 2013.

[3] Antal, E. and T. Vinkó (2017). Modeling maxmin fair bandwidth allocation in bittorrent communities. Computational Optimization and Applications 66 (2), 383-400.

[4] Bouten, Niels, Steven Latré, Jeroen Famaey, Werner Van Leekwijck, and Filip De Turck. "In-network quality optimization for adaptive video streaming services." IEEE Transactions on Multimedia 16, no. 8 (2014): 2281-2293.

[5] Chen, Liang, Yipeng Zhou, and Dah Ming Chiu. "Smart streaming for online video services." IEEE Transactions on Multimedia 17, no. 4 (2015): 485-497.

[6] De Cicco, Luca, and Saverio Mascolo. "An adaptive video streaming control system: Modeling, validation, and performance evaluation." IEEE/ACM Transactions on Networking (TON) 22, no. 2 (2014): 526-539.

[7] De Cicco, Luca, Vito Caldaralo, Vittorio Palmisano, and Saverio Mascolo. "Elastic: a client-side controller for dynamic adaptive streaming over http (dash)." In 2013 20th International Packet Video Workshop, pp. 1-8. IEEE, 2013.

Volume: 10 Issue: 03 Pages: 3898-3904 (2018) ISSN: 0975-0290

[8] De Cicco, Luca, Vito Caldaralo, Vittorio Palmisano, and Saverio Mascolo. "TAPAS: a Tool for rApid Prototyping of Adaptive Streaming algorithms." In Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming, pp. 1-6. ACM, 2014.

[9] Dong, Y.-n. and K. Wang (2017). Fine grained classification of internet multimedia tra-cs. In Advanced Communication Technology (ICACT), 2017 19th International Conference on, pp. 668-672. IEEE.

[10] Ebrahimi-Taghizadeh, S., A. Helmy, and S. Gupta (2005). Tcp vs. tcp: a systematic study of adverse impact of short-lived tcp flows on long-lived tcp flows. In INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, Volume 2, pp. 926-937. IEEE.

[11] Ginevičius, Romualdas. "Normalization of quantities of various dimensions."Journal of business economics and management 9, no. 1 (2008): 79-86.

[12] He, Jian, Zheng Xue, Di Wu, Dapeng Oliver Wu, and Yonggang Wen. "CBM: online strategies on cost-aware buffer management for mobile video streaming." IEEE Transactions on Multimedia 16, no. 1 (2014): 242-252.

[13] Irondi, Iheanyi, Qi Wang, and Christos Grecos. "Empirical evaluation of H. 265/HEVC-based dynamic adaptive video streaming over HTTP (HEVC-DASH)." In SPIE Photonics Europe, pp. 91390L-91390L. International Society for Optics and Photonics, 2014.

[14] Ishakian, V., R. Sweha, and A. Bestavros (2017). Angelcast: Peer-assisted live streaming using optimized multi-tree construction. Computer Communications.

[15] Jarnikov, Dmitri, and Tanır Özçelebi. "Client intelligence for adaptive streaming solutions." Signal Processing: Image Communication 26, no. 7 (2011): 378-389.

[16] Jiang, Junchen, Vyas Sekar, and Hui Zhang. "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive." InProceedings of the 8th international conference on Emerging networking experiments and technologies, pp. 97-108. ACM, 2012.

[17] Juluri, Parikshit, Venkatesh Tamarapalli, and Deep Medhi. "Measurement of Quality of Experience of Video-on-Demand Services: A Survey." IEEE Communications Surveys & Tutorials 18, no. 1 (2016): 401-418.

[18] Lederer, Stefan, Christopher Müller, and Christian Timmerer. "Dynamic adaptive streaming over HTTP dataset." In Proceedings of the 3rd Multimedia Systems Conference, pp. 89-94. ACM, 2012.

[19] Li, L., K. Xu, D. Wang, C. Peng, K. Zheng, R. Mijumbi, and Q. Xiao (2017). A longitudinal measurement study of tcp performance and behavior in 3g/4g networks over high speed rails. IEEE/ACM Transactions on Networking.

[20] Li, Zhi, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C. Begen, and David Oran. "Probe and adapt: Rate adaptation for http video streaming at scale." IEEE Journal on Selected Areas in Communications 32, no. 4 (2014): 719-733.

[21] Liu, K., V. Aggarwal, Z. Shao, and M. Chen (2017). Joint upload-download tcp acceleration over mobile data networks. In Sensing, Communication, and Networking (SECON), 2017 14th Annual IEEE International Conference on, pp. 1-9. IEEE.

[22] Magharei, Nazanin, Reza Rejaie, Ivica Rimac, Volker Hilt, and Markus Hofmann. "ISP-friendly live P2P streaming." IEEE/ACM Transactions on Networking 22, no. 1 (2014): 244-256.

[23] Mansy, Ahmed, Bill Ver Steeg, and Mostafa Ammar. "Sabre: A client based technique for mitigating the buffer bloat effect of adaptive video flows." InProceedings of the 4th ACM Multimedia Systems Conference, pp. 214-225. ACM, 2013.

[24] Miller, Konstantin, Dilip Bethanabhotla, Giuseppe Caire, and Adam Wolisz. "A control-theoretic approach to adaptive video streaming in dense wireless networks." IEEE Transactions on Multimedia 17, no. 8 (2015): 1309-1322.

[25] Miller, Konstantin, Emanuele Quacchio, Gianluca Gennari, and Adam Wolisz. "Adaptation algorithm for adaptive streaming over HTTP." In 2012 19th International Packet Video Workshop (PV), pp. 173-178. IEEE, 2012.

[26] Mueller, Christopher, Stefan Lederer, and Christian Timmerer. "A proxy effect analyis and fair adatpation algorithm for multiple competing dynamic adaptive streaming over HTTP clients." In Visual Communications and Image Processing (VCIP), 2012 IEEE, pp. 1-6. IEEE, 2012.

[27] Nikmanzar, Sepideh, Akbar Ghaffarpour Rahbar, and Amin Ebrahimzadeh. "On-Demand Video Streaming Schemes Over Shared-WDM-PONs." IEEE Transactions on Circuits and Systems for Video Technology 23, no. 9 (2013): 1577-1588.

[28] Petrangeli, Stefano, Jeroen Famaey, Maxim Claeys, Steven Latré, and Filip De Turck. "QoE-driven rate adaptation heuristic for fair adaptive video streaming." ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 12, no. 2 (2016): 28.

[29] Rastegarfar, H., K. Keykhosravi, K. Szczerba, E. Agrell, L. LaComb, and M. Glick (2017). Optical circuit granularity impact in tcp-dominant hybrid data center networks. In Computing, Networking and Communications (ICNC), 2017 International Conference on, pp. 318322. IEEE.

[30] Robinson, David C., Yves Jutras, and Viorel Craciun. "Subjective video quality assessment of HTTP adaptive streaming technologies." Bell Labs Technical Journal 16, no. 4 (2012): 5-23.

[31] Seufert, Michael, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hoßfeld, and Phuoc Tran-Gia. "A survey on quality of experience of http adaptive

Volume: 10 Issue: 03 Pages: 3898-3904 (2018) ISSN: 0975-0290

streaming." IEEE Communications Surveys & Tutorials 17, no. 1 (2015): 469-492.
[32] Stockhammer, Thomas. "Dynamic adaptive streaming over HTTP--: standards and design principles." In Proceedings of the second annual ACM conference on Multimedia systems, pp. 133-144. ACM, 2011.
[33] Su, Guan-Ming, Xiao Su, Yan Bai, Mea Wang, Athanasios V. Vasilakos, and Haohong Wang. "QoE in video streaming over wireless networks: perspectives and research challenges." Wireless Networks (2015): 1-23.
[34] Sunny, A., S. Panchal, N. Vidhani, S. Krishnasamy, S. Anand, M. Hegde, J. Kuri, and A. Kumar (2017). A generic controller for managing tcp transfers in ieee 802.11 infrastructure wlans. Journal of Network and Computer Applications 93, 13-26.
[35] Tran, H. T. T., Y. Won, and J. Kim (2017). An e-cient hybrid push-pull methodology for peer-to-peer video live streaming system on mobile broadcasting social media. Multimedia Tools and Applications 76 (2), 2557-2568.
[36] Wamser, Florian, David Hock, Michael Seufert, Barbara Staehle, Rastin Pries, and Phuoc Tran-Gia. "Using buffered playtime for QoE-oriented resource management of YouTube video streaming." Transactions on Emerging Telecommunications Technologies 24, no. 3 (2013): 288-302.
[37] Wichtlhuber, Matthias, Robert Reinecke, and David Hausheer. "An SDN-based CDN/ISP collaboration architecture for managing high-volume flows."IEEE Transactions on Network and Service Management 12, no. 1 (2015): 48-60.
[38] Wu, Jiyan, Bo Cheng, Chau Yuen, Ngai-Man Cheung, and Junliang Chen. "Trading delay for distortion in one-way video communication over the internet." IEEE Transactions on Circuits and Systems for Video Technology26, no. 4 (2016): 711-723.
[39] Yin, Xiaoqi, Vyas Sekar, and Bruno Sinopoli. "Toward a principled framework to design dynamic adaptive streaming algorithms over http." In Proceedings of the 13th ACM Workshop on Hot Topics in Networks, p. 9. ACM, 2014.
[40] Yu, Hongliang, Dongdong Zheng, Ben Y. Zhao, and Weimin Zheng. "Understanding user behavior in large-scale video-on-demand systems." InACM SIGOPS Operating Systems Review, vol. 40, no. 4, pp. 333-344. ACM, 2006.
[41] Zhou, Chao, Chia-Wen Lin, Xinggong Zhang, and Zongming Guo. "A control-theoretic approach to rate adaption for DASH over multiple content distribution servers." IEEE Transactions on Circuits and Systems for Video Technology 24, no. 4 (2014): 681-694.

AUTHOR DETAILS

Koffka Khan received the M.Sc., and M.Phil. degrees from the University of the West Indies. He is currently a PhD student and has up-to-date, published numerous papers in journals & proceedings of international repute. His research areas are computational intelligence, routing protocols, wireless communications, information security and adaptive streaming controllers.

Wayne Goodridge is a Lecturer in the Department of Computing and Information Technology, The University of the West Indies, St. Augustine. He did is PhD at Dalhousie University and his research interest includes computer communications and security.