Volume: 10 Issue: 05 Pages: 4035-4044(2019) ISSN: 0975-0290

Variants of the Constrained Bottleneck LAN Edge Link in Household Networks

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 reduces user-QoE. This occurs as resource allocation becomes unfair. We refer to this as the bottleneck link problem. The resource imbalance creates severe user annoyance such as screen flickering and video freezes. However, there are numerous conditions that amplifies the problems. Some of these are well documented in the literature but we intend to highlight the major conditions at a bottleneck link in household LANs. These are time-varying bandwidth, TCP long-lived flows, players pausing, starting/re-starting and stopping and increases in player numbers. We explore these conditions and evaluate the performance of heuristic adaptive video players. ELASTIC and PANDA players are evaluated. Experimental setup includes the TAPAS player and emulated network conditions. The results show that players are well suited to specific conditions.

Keywords— **user-QoE, resource, allocation, unfair, screen flickering, video freezes, bottleneck, household, time-varying, TCP long-lived flow, TAPAS, ELASTIC, PANDA**

Date of Submission: Jan 28, 2019 Date of Acceptance: Apr 22, 2019

I. INTRODUCTION

Adaptive video streaming is very popular in today's households as so many people use video for diverse tasks, including viewing videos over the Internet, playing games and speaking to friends or family. Because of this the home router becomes a bottleneck for network traffic. We call this the constrained bottleneck link problem. Improving streaming performance when multiple devices compete at this bottleneck link has become a primary focus for many current research efforts.

However, when a player streams at a bottleneck link many different conditions are present. We have gathered a set of these conditions based on an extensive literature search. This paper tries to bring research efforts in adaptive video streaming together with a proposed standard set of experiments with a comprehensive set of conditions.

The conditions we have identified which are present at bottleneck links are time-varying bandwidth, TCP long lived flows, players pausing, starting/re-starting and stopping, players looking at different videos and number of players increasing. We call these conditions variants of the "standard" bottleneck link problem. "Standard" means the conditions we identify are not present in the streaming scenario.

In this paper we look at scenarios where players compete for bandwidth at a bottleneck link under these conditions. We look at client-based adaptive streaming heuristic methods. In these methods the client player makes the adaptation decision on which video segment to select next. They are heuristic as they try to make a "best guess" in making this decision which is not always guaranteed to be optimal.

The rest of paper is organized as follows. Section 2 explores different heuristic adaptive streaming approaches including PANDA and ELASTIC players. In Section 3, we look at the different conditions at a bottleneck link under which these adaptive video streaming players compete. Section 4 looks at the experimental setup of the emulations for the different conditions 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 [17], [31]. This paper, which portrays the FESTIVE control algorithm [17], confirms that numerous problems occur when multiple bitrate-adaptive players (adaptation over HTTP) share a bottleneck link [44]. 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.

The authors in [22], 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

Volume: 10 Issue: 05 Pages: 4035-4044(2019) ISSN: 0975-0290

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

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

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

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

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 1: PANDA four-step model. [22]

The authors illustrate a four-step model (see Figure 1) 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 [41], [13], [26] etc… (cf. Figure 2) 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.

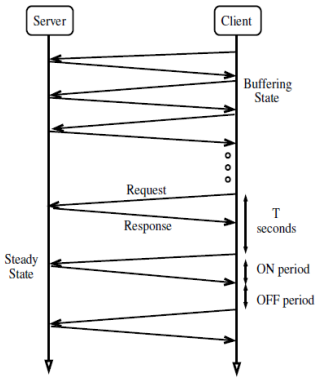

Figure 2: The request-response timing between client and server in the Buffering and Steady states. [1]

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 3: ELASTIC Controller Pseudo-code. [7]

ELASTIC [7] proposes an approach (cf. Figure 3) 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 [42], where experimental evaluation showed that ELASTIC is able to get the fair share when competing with TCP long-lived flows.

III. VARIANTS OF THE CONSTRAINED BOTTLENECK LINK PROBLEM

We group the variants of the constrained bottleneck link problem into three categories, cf. Figure 4:

- Category 1:
  - Constrained Bandwidth
  - Different Video Download
- Category 2:
  - Time-varying Bandwidth
  - Players start, stop and pause at different times
  - Increase in the number of players
- Category 3:
  - TCP long-lived flows

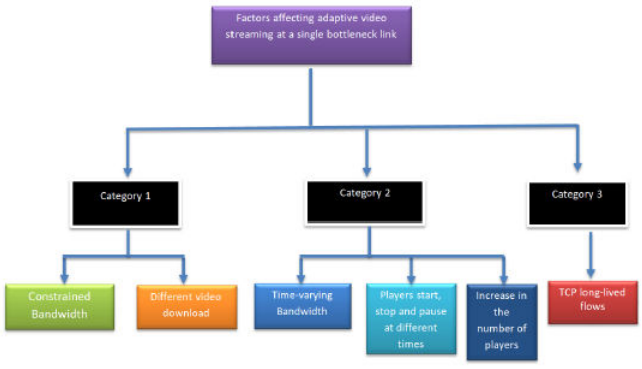

Figure 4: Variants of adaptive video streaming are placed into three categories when multiple competing players share a bottleneck link.

Volume: 10 Issue: 05 Pages: 4035-4044(2019) ISSN: 0975-0290

The categories are based on similar outcomes or effects of the conditions on streaming video. However, in the following sections we treat each condition individually.

4.1 Multi-player competition at a home bottleneck link

Multiple competing players are very common in home networks. This makes the ON-OFF problem a great nuisance to user-QoE. Thus, this paper covers adaptive players sharing a single bottleneck link at home, i.e., where the ON-OFF problem is prevalent [49], [50]. Figure 5 depicts traffic flow at a bottleneck link. This work studies these factors existing at a bandwidth constrained bottleneck link which is very common in household LANs.

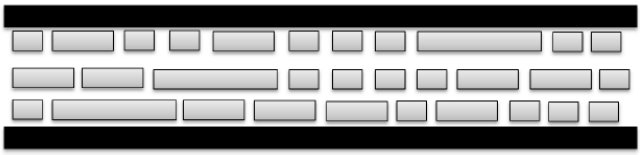

Figure 5: The bottleneck link contains different size segments for different players. High data request causes the bottleneck link to become full to capacity.

4.2 Time-varying bandwidth

In practice, available network resource for media streaming can change over time due to fluctuating link capacity in wireless networks, [40] or influence from other traffic. In an environment with stable capacity, past observations yield good estimates of future capacity. But if capacity is varying widely, estimating future capacity is much harder [48]. In general, large buffer sizes compensates network bandwidth variations. Ample buffering of too many segments guarantees a smooth video rate [21] and Figure 6.

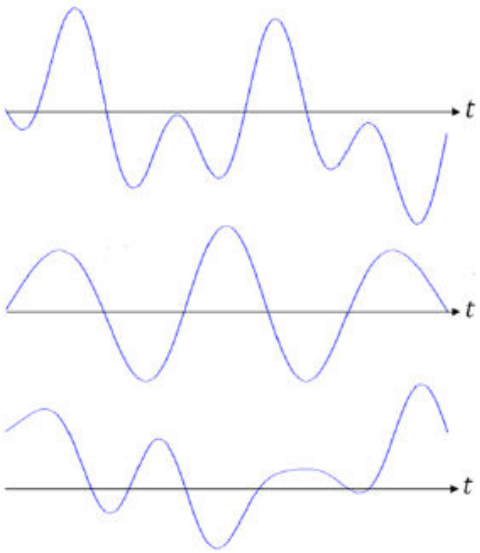

Figure 6: The bottleneck link is used by many adaptive players. Shown here are the bandwidth of three competing adaptive players using the Conventional player.

The research community utilizes the term "available bandwidth" in varying context. We adopt available bandwidth [32]. In a network path each link *j* has a certain capacity or nominal bandwidth C_j. The network interfaces in the nodes at each end of the link determine this value. The nominal bandwidth typically does not vary [11]. Usually, varying bandwidth happens within short time-scales. It is based on the link load or cross traffic Y_j=Yj(t, λ) where λ is the time resolution describing traffic fluctuations. The cross traffic rate is given by the Equation below:

$$Y_j = \frac{1}{\lambda} A_j(t-\lambda, t)$$

where A_j(t-λ, t) is the number of bits over link *j* during a time interval λ.

The time-varying bandwidth B_j=B_j(t, λ) of the link *j* is given by the following Equation:

$$B_j = C_j - Y_j$$

The bottleneck link is one of the links along the path that has the smallest available bandwidth value, see Figure 7. It determines the available bandwidth of the path. The available bandwidth is the smallest increase in traffic load from sender to receiver at time t which causes congestion at some hop on the network path [11].

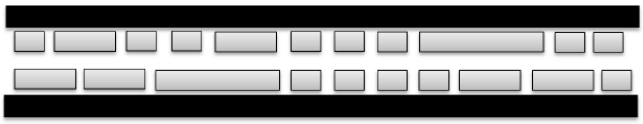

Figure 7: The bottleneck link contains the smallest value of the available bandwidth. However, when bandwidth varies the bottleneck's capacity may reduce. Note that when the bottleneck bandwidth reduces there is only two lanes of traffic. This represent a reduction from the previous three lanes capacity (the full capacity of the bottleneck link.

In addition, bandwidth can become too low or high in consecutive time intervals. Consequently, the link can become either oversubscribed, see Figure 8 or undersubscribed, see Figure 9. This causes user-QoE too suffer even further. [46] observes: drastic changes on streaming bitrates may ruin the stability of a player.

4.3 TCP long-lived flows

Adaptive video players are not able to get a fair share when coexisting with a TCP greedy flow [1], see Figure 10. TCP long-lived flows completely shut off TCP short-lived flows [24]. 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 concern for adaptive

Volume: 10 Issue: 05 Pages: 4035-4044(2019) ISSN: 0975-0290

video players. The outcome is the "downward spiral" effect, . Remedies include increasing segment size and filtering of bandwidth estimates. Examples of TCP long-lived flows 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 occurs during a TCP long-lived flow. These include TCP congestion

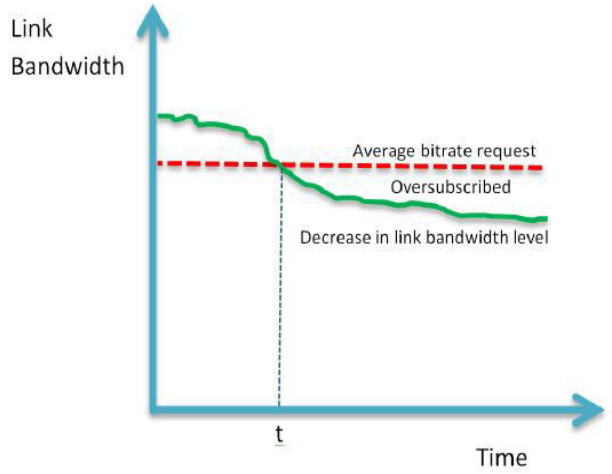

Figure 8: Starting at t seconds the bandwidth reduces and the link becomes oversubscribed. Each player requests segments larger than its current fair share portion of the bandwidth. At the end of each download request there would be unfinished segment downloads.

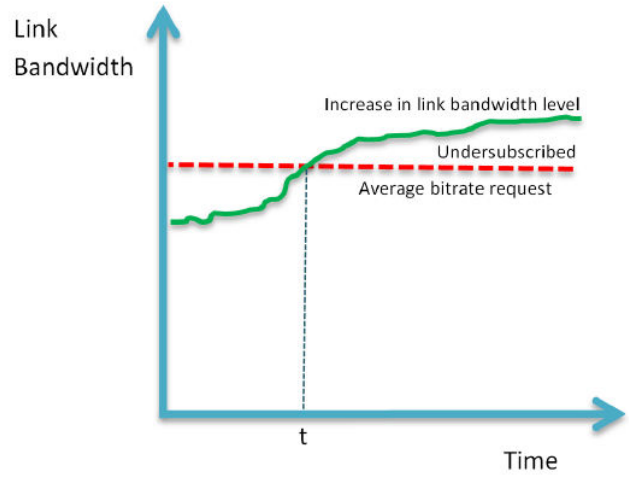

Figure 9: Initially the link is oversubscribed. At t seconds link bandwidth increases and the link becomes undersubscribed. A link utilization gap appears.

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 [51]. 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. 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. However, TCP short-lived flows must try to utilize as much bandwidth as possible when coexisting with TCP long-lived flows. TCP long-lived flows are shown to hurt TCP short-lived flows in terms of end-to-end delay and consequently throughput [10].

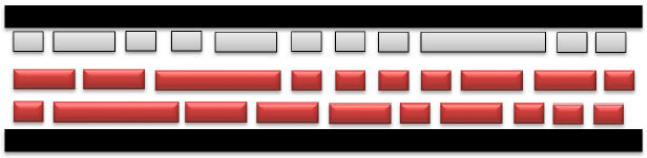

Figure 10: TCP long-lived flows (red segments) compete with video flows for bandwidth. Notice that the TCP long-lived flows can consume most of the bottleneck link bandwidth because it is a "greedy" flow.

We now consider an initial scenario setting where three TCP long-lived flows pass through a bottleneck link. 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 does 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 during inactive periods. An example is BitTorrent, [3] which exhibits this behavior during choke periods.

Some 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$\ldots$ These intermittent data traffic competes with video flows for bandwidth. These other flows may

Volume: 10 Issue: 05 Pages: 4035-4044(2019) ISSN: 0975-0290

utilize UDP as their transport protocol. However, in comparison to co-existing TCP flows UDP flows utilize more than their fair share of the bandwidth [10].

4.4 Players start, pause and stop

Most adaptive video streaming downloads assume players keep their same settings and initial states. This is a strong assumption. In reality players use different initial state and join at arbitrary times (cf. Figure 11). It is therefore essential to evaluate adaptive streaming algorithms in settings where players are different. For example, with different initial bitrate and buffer levels. Further, when players start, stop or pause there are changes in the demand for video segments.

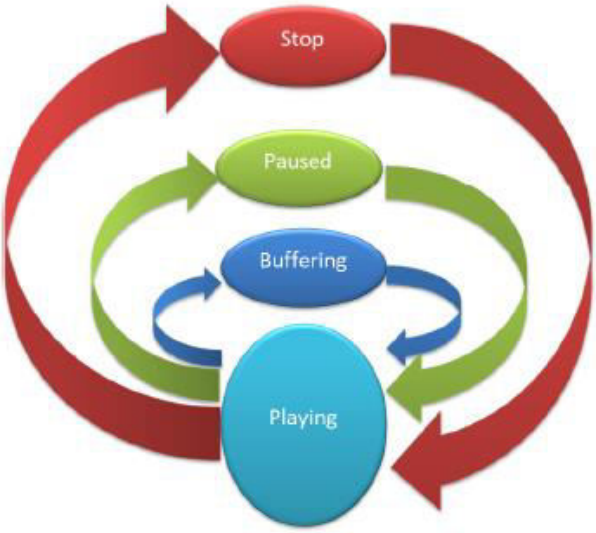

Figure 11: Players exist in different states during streaming. This study focuses on Stopping, Pausing, Buffering and Playing.

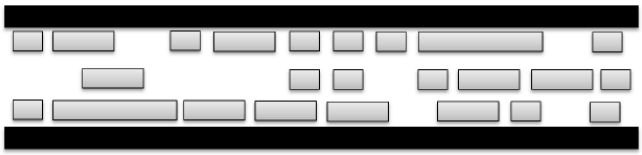

Figure 12: Gaps exist in the traffic flows when players start, stop or pause during streaming.

4.5 Players stream different videos

Members in a household express different interests. Thus, video players in a household are likely to concurrently download different videos. These videos contain different sets of qualities/bitrate levels, see Figure 12. The stability problem with multiple videos is challenging since each video possesses unique requirements, [46]. For example, a user viewing a very high quality video may over consume bandwidth starving other users. However, if the very high quality video user utilizes only what it requires and spares the rest of the bandwidth to other low quality video users (these users do not need high quality video) then the situation works out.

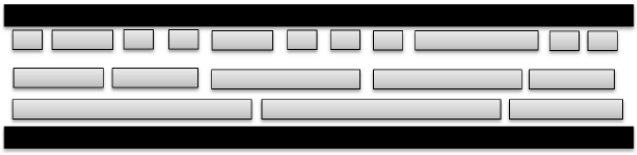

Figure 12: Different videos may have different quality bitrates. Thus, the sizes of the segments vary for the three different flows shown.

4.6 Number of players increase

The number of players connecting to a bottleneck link is a very important. A one-to-one model produces no problem once sufficient bandwidth is available for the video, see Figure 13. Internet service providers (ISPs) usually view streaming as a one-to-many model. Competing adaptive players create a problem, see Figure 14. The challenge is to provide good user-QoE for all players as the number of players increases, see Figure 15.

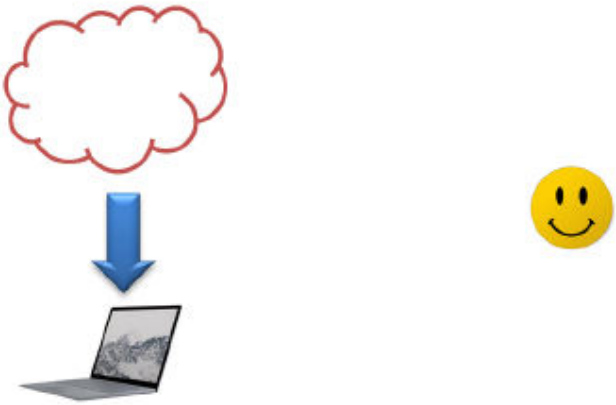

Figure 13: One adaptive video player with sufficient bandwidth obtains no difficulty in getting high quality video segments to the user.

IV. 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 (cf. Figure 16). 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 [52] in the form of a GET request.

Volume: 10 Issue: 05 Pages: 4035-4044(2019) ISSN: 0975-0290

Figure 14: Few adaptive video players with insufficient bandwidth has difficulty in getting high quality video segments to the various users.

players

Figure 15: An increasingly large number of adaptive video players with insufficient bandwidth has great difficulty in getting high quality video segments to the various users. The challenge is to get an adaptive streaming approach that will give all players in this situation sufficient quality video segments.

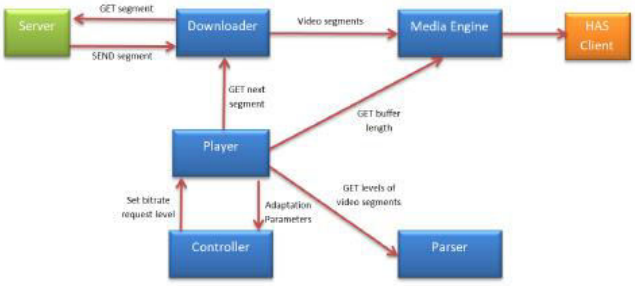

Figure 16: Adaptive Video Streaming - The player consists of four components: the Downloader, the Media Engine, the Parser and the Controller. The player sends a GET request to the server for the next video segment. The server returns the segment and it is stored in the buffer of the Downloader. The Media player gets segments to play from the Downloader. The Parser extracts the video levels from the manifest file. The Controller sets the rate value for the next segment download.

A virtual network is setup on the same host machine creating a custom emulation framework (see Figure 17). 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 = 15Mbps for the two player experiments. QoE metrics are described as follows:

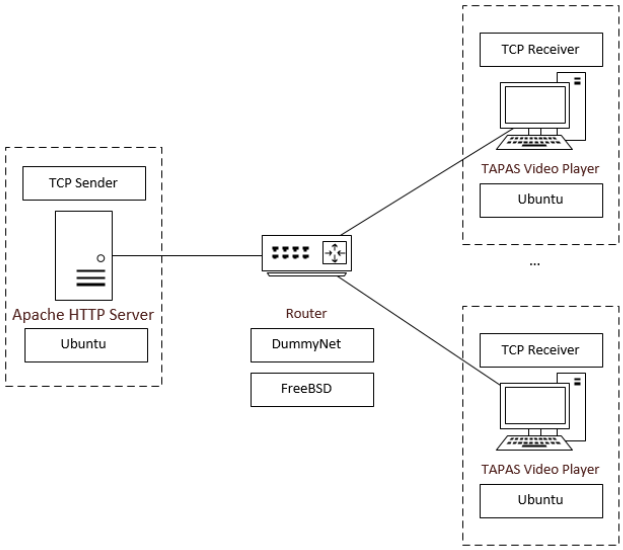

Figure 17: Network testbed setup.

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).

Volume: 10 Issue: 05 Pages: 4035-4044(2019) ISSN: 0975-0290

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

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 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.

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

V. RESULTS

We present the performance of the players under the variant conditions using an adaptive streaming approach, see Tables 1, 2, 3, 4, 5 and 6. We observe ELASTIC outperforms the other players in the single bottleneck experiment. This is because ELASTIC has only ON periods (no OFF periods are present) which enables it to capture larger amounts of bandwidth and aggressively compete against TCP long-lived flows. However, PANDA's probe and adapt mechanism gives it the ability to detect changes in bandwidth and respond quickly to these changes. Thus, PANDA outperforms ELASTIC in the time-varying bandwidth, players, start, stop and pause experiments and when number of players increases. PANDA's probe mechanism is also able to detect bandwidth usage of players with different videos and so outperforms ELASTIC. The Conventional performs worst in all experiments.

Table 1: Control experiment: single bottleneck link.

	ELASTIC	PANDA	Conventional
Utilization	0.87	0.86	0.81
Unfairness	0.035	0.017	0.033
Re-buffering ratio	0.054	0.043	0.055
Instability	0.017	0.012	0.030
Average Quality	3.94	3.86	3.51

Table 2: Time-varying bandwidth.

	ELASTIC	PANDA	Conventional
Utilization	0.78	0.81	0.74
Unfairness	0.102	0.043	0.078
Re-buffering ratio	0.040	0.038	0.056
Instability	0.022	0.022	0.026
Average Quality	3.59	3.51	3.26

Table 3: Long-lived TCP flows.

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

Table 4: Players start, stop and pause.

	ELASTIC	PANDA	Conventional
Utilization	0.81	0.83	0.79
Unfairness	0.058	0.051	0.065
Re-buffering ratio	0.087	0.070	0.095
Instability	0.051	0.043	0.076
Average Quality	7.02	6.98	6.81

Table 5: Players stream different video.

	ELASTIC	PANDA	Conventional
Utilization	0.68	0.70	0.65
Unfairness	0.0265	0.0073	0.0072
Re-buffering ratio	0.009	0.062	0.019
Instability	0.057	0.023	0.055
Average Quality	3.45	3.33	3.27

For the number of players increase experiment we increase players from 2 to 30 players. We present graphs averaged estimated bandwidth for a random two players. The Figures (cf. Figure 18, 19 and 20) below shows us that PANDA does best amongst all competing players.

Volume: 10 Issue: 05 Pages: 4035-4044(2019) ISSN: 0975-0290

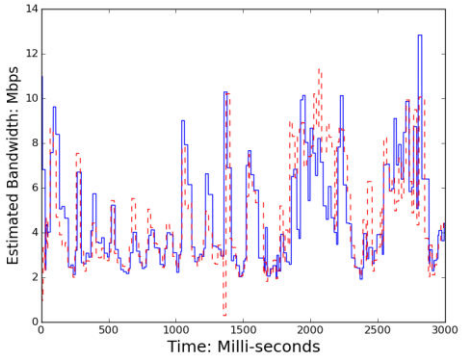

Figure 18: PANDA: Average estimated bandwidth for increasing player experiment.

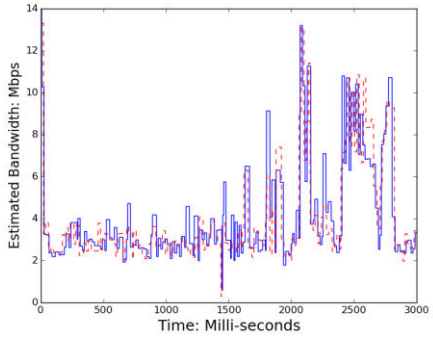

Figure 19: ELASTIC: Average estimated bandwidth for increasing player experiment.

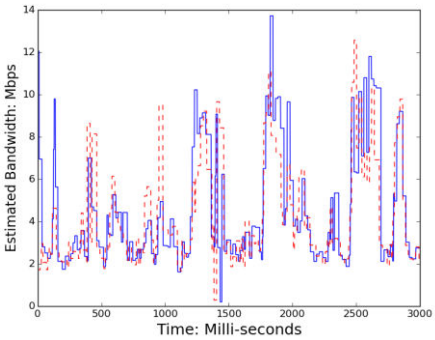

Figure 20: Conventional: Average estimated bandwidth for increasing player experiment.

VI. CONCLUSION

Competition among adaptive video streaming players severely reduces user-QoE. This occurs as resource allocation becomes unfair. We refer to this as the bottleneck link problem. The resource imbalance creates severe user annoyance such as screen flickering and video freezes. However, there are numerous conditions that amplifies the problems. Some of these are well documented in the literature but we intend to highlight the major conditions at a bottleneck link in household LANs. These are time-varying bandwidth, TCP long-lived flows, players pausing, starting/re-starting and stopping and increases in player numbers. We explore these conditions and evaluate the performance of heuristic adaptive video players. ELASTIC and PANDA players are evaluated. Experimental setup includes the TAPAS player and emulated network conditions. The results show that players are well suited to specific conditions. ELASTIC performs best when TCP long lived flows are present and in the controlled bottleneck experiment. However, PANDA did best under all other conditions. The Conventional player performs the worst.

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.

[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

Volume: 10 Issue: 05 Pages: 4035-4044(2019) ISSN: 0975-0290

Communications Societies. Proceedings IEEE, Volume 2, pp. 926-937. IEEE.

[11] Ekelin, S., M. Nilsson, E. Hartikainen, A. Johnsson, J.-E. Mangs, B. Melander, and M. Bjorkman (2006). Real-time measurement of end-to-end available bandwidth using kalman filtering. In Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP, pp. 73–84. IEEE.

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

[13] 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.

[14] 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.

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

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

[17] 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.

[18] 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.

[19] K. Khan and W. Goodridge, "S-MDP: Streaming with Markov Decision Processes," in IEEE Transactions on Multimedia. doi: 10.1109/TMM.2019.2892304

[20] Khan, Koffka, and Wayne Goodridge. "B-DASH: broadcast-based dynamic adaptive streaming over HTTP." International Journal of Autonomous and Adaptive Communications Systems 12, no. 1 (2019): 50-74.

[21] Khan, Koffka, and Wayne Goodridge. "Server-based and network-assisted solutions for adaptive video streaming." International Journal of Advanced Networking and Applications 9, no. 3 (2017): 3432-3442.

[22] 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.

[23] 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.

[24] Li, Y., Y. Wang, S. Wang, and S. Ma (2016). An adaptative bitrate algorithm for dash. In Multimedia & Expo Workshops (ICMEW), 2016 IEEE International Conference on, pp. 1–4. IEEE.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] Pavithra, K., and E. Karthikeyan. "A Study on Congestion Control Algorithms in Computer Networks." In Proceedings of the UGC Sponsored National Conference on Advanced Networking and Applications. 2015.

[35] Petrangeli, Stefano, Jeroen Famaey, Maxim Claeys, Steven Latré, and Filip De Turck. "QoE-driven rate adaptation heuristic for fair adaptive video

Volume: 10 Issue: 05 Pages: 4035-4044(2019) ISSN: 0975-0290

streaming." ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 12, no. 2 (2016): 28.
[36] Prasad, R., C. Dovrolis, M. Murray, and K. Claffy (2003). Bandwidth estimation: metrics, measurement techniques, and tools. IEEE network 17 (6), 27–35.
[37] 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.
[38] 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.
[39] Seufert, Michael, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hoßfeld, and Phuoc Tran-Gia. "A survey on quality of experience of http adaptive streaming." IEEE Communications Surveys & Tutorials 17, no. 1 (2015): 469-492.
[40] 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.
[41] 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.
[42] 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.
[43] 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.
[44] Tychogiorgos, G., A. Gkelias, and K. K. Leung (2012). Utility-proportional fairness in wireless networks. In Personal indoor and mobile radio communications (PIMRC), 2012 IEEE 23rd international symposium on, pp. 839–844. IEEE.
[45] 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.
[46] 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.
[47] 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.
[48] 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.
[49] 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.
[50] Zhang, S., B. Li, and B. Li (2015). Presto: Towards fair and efficient http adaptive streaming from multiple servers. In Communications (ICC), 2015 IEEE International Conference on, pp. 6849–6854. IEEE.
[51] 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.
[52] Zou, S., J. Huang, Y. Zhou, J. Wang, and T. He (2017). Flow-aware adaptive pacing to mitigate tcp incast in data center networks. In Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on, pp. 2119–2124. IEEE.

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.