Volume: 11 Issue: 02 Pages: 4236-4243 (2019) ISSN: 0975-0290

Collaborative Methods to Reduce the Disastrous Effects of the Overlapping ON Problem in DASH

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

The performance of today's adaptive video streaming players (DASH) is severely hindered by overlapping ON-OFF patterns that occurs during a streaming session. High switching rates, freezing and skipping annoy users resulting in poor user-QoE. This paper attempts to overcome the ON-OFF problem by keeping players aware of each other's downloads and inter-request times. Using this a player is able to better predict future player actions and reschedule their downloads to produce least conflicts with others. If a player's start segment download overlaps with the end of one or more players' current downloads within a certain time t_1 it waits until the download of others completes before starting its own download. Conversely if too many players are sharing the bottleneck link a player will hold of its download by a time t_2 enabling others to move towards completion of their downloads. This reduces the overlap of its download with other players. Both these methods help reduce competition at the bottleneck and produces a more balanced sharing of network resources.

Keywords – **DASH; overlapping; ON-OFF; streaming; user-QoE; bottleneck; link; network; resources; DASH.**

Date of Submission: Aug 30, 2019 Date of Acceptance: Sep 26, 2019

I. INTRODUCTION

Video over IP (VoIP) is becoming more and more important as we move further into the twenty-first century. The Internet is still growing rapidly and more uses are being found for video users. These include real-time online visual assistance, video learning, live event streaming, smart HDTVs, mobile phones, gaming devices, computers and visual communication among others. As the content quality is improving to meet end-user demands the bandwidth requirement for such devices is rapidly increasing. With increasing bandwidth demands and profuse video content, it is becoming likely that two or more adaptive streaming players may have to share a network bottleneck. This will result in a competition for available bandwidth. Example scenarios where this can take place are, when a number of people in the same household view similar or different videos simultaneously. Here, the domestic broadband access link is the shared bottleneck. Another instance of such competition is when many users watch the same live event (such as World Cup Soccer) online. The shared bottleneck may be an edge network link in this scenario. It has been previously observed that such competition can lead to performance issues [2] [1] [16] [13].

The concept of adaptive video streaming (see Figure 1) is based on the idea to adapt the bandwidth required by the video stream to the throughput available on the network path from the stream source to the client [1]. These algorithms can live at the server [15], at an intermediate network device [15] or at the client [16]. With client-side protocols it is the player that decides what bitrate to request for any fragment, improving server-side scalability [15]. A benefit of this protocol is that the player can control its playback buffer size by dynamically adjusting the rate at which new fragments are requested. The adaptation is performed by varying the quality of the streamed video.

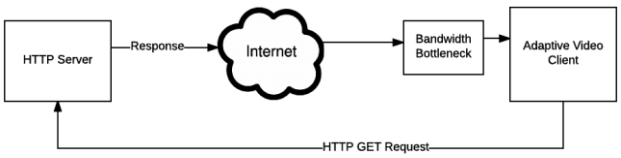

Fig. 1. DASH-based Adaptive Video Streaming

Multiple video segments constitute a video stream lasting from as little as 2 seconds to as much as having a 10 second chunk delivery rate. Segments are encoded and stored on the server in numerous quality versions, termed representations. Each version has a unique resolution, bitrate and/or quality. A client downloads segments using HTTP GET statements [4]. However, with adaptive streaming a client might request subsequent segments at different quality levels to manage varying network conditions, based on an estimation bandwidth. To do this it uses a manifest file that contains information about the video segments. Protocols and standards such as MPEG Dynamic Adaptive Streaming over HTTP (DASH) [27], Apple HTTP Live Streaming (HLS) [21], Microsoft Smooth Streaming (MSS) [6] or Adobe HTTP Dynamic Streaming (HDS) [32] typically use a media playlist that contains a list of uniform resource identifiers (URIs) that are addresses to media segments [1].

The process of determining the ideal representation for each segment to enhance the user's experience is pivotal to adaptive streaming. The controller algorithm estimates the

Volume: 11 Issue: 02 Pages: 4236-4243 (2019) ISSN: 0975-0290

network bandwidth and chooses the next bitrate level corresponding to the available network bandwidth. Variations in the available bandwidth will result in jerky playback and disruption of the video playback if the throughput falls below the bit rate requirement of the video. This is the major challenge in adaptive video streaming [15]. Selecting appropriate bitrate levels helps to maximize the user experience. Generally, higher bitrates and resolutions will give better user experience. For example, if a client approximates that there is 9.5Mb/s available in the network, it might request the server to stream video compressed to the highest video rate available, 9.5Mb/s, or the next rate below, 9.3Mb/s. If the client picks a video rate that is too high, the viewer will experience annoying re-buffering events; if they pick a streaming rate that is too low, the viewer will experience poor video quality. In both cases, the experience degrades [23] [3] [9] and user may take their viewing elsewhere. It is therefore important for a video streaming service to select the highest safe video rate [33].

To the authors review of existing literature there is no known findings of adaptive streaming players with collaborative player-to-player communications (see Figure 2) with preemptive and non-preemptive scheduling. We propose, implement and test two collaborative communication methods (CCMs) for DASH-based players. It primarily aims to obtain better fair sharing of resources in streaming environments for example a company's local area network (LAN).

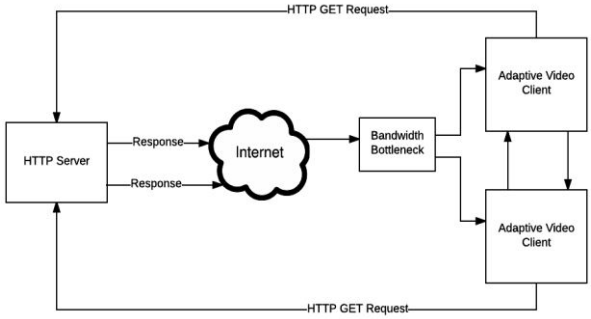

Fig. 2. Collaborative DASH-based Adaptive Video Streaming

In Section II we provide a literature review. We describe DASH-based adaptive video players. Some of these players are part of our experiments which is presented in later sections. This work provides a body of work that builds on DASH overlapping ON problem which is presented in Section III. It provides a detailed methodology on the collaborative methods on which this work is based on in Section IV. This is followed by the experimental setup in Section V. Results is given in Section VI. Finally, in Section VII we present the conclusion.

II. LITERATURE REVIEW

FINEAS (Fair In-Network Enhanced Adaptive Streaming) is proposed [24], which is capable of increasing clients' Quality of Experience (QoE) and achieving fairness in a multi-client setting. A key element of their protocol is an in-network system of coordination proxies in charge of facilitating fair resource sharing among clients. They claim that fairness is achieved without explicit communication among clients. In addition, viewers using HTTP Adaptive Streaming (HAS) without sufficient bandwidth undergo frequent quality switches that hinder their watching experience. This situation, known as instability, is produced when HAS players are unable to accurately estimate the available bandwidth. Moreover, when several players stream over a bottleneck link, their individual adaptation techniques may result in an unfair share of the channel. These are two detrimental issues in HAS technology, which is otherwise very attractive. The authors [17] describe an implementation in the form of an HTTP proxy server and show that both stability and fairness are strongly improved. In [6] several network-assisted streaming protocols which rely on active cooperation between video streaming applications and the network are explored. They use a Video Control Plane which enforces Video Quality Fairness among concurrent video flows generated by heterogeneous client devices. A max-min fairness optimization problem is solved at run-time. They compare two protocols to actuate the optimal solution in an SDN network: the first one allocating network bandwidth slices to video flows, the second one guiding video players in the video bitrate selection.

In [26] the bandwidth estimate generated at the server is used for server-side adaptive bit encoding of digital media streams. The server application measures the network bandwidth available to the individual client for TCP/IP [14] downloads of media and accordingly adjusts stream bit rate and composition to allow the client to retrieve the media stream with sufficient time margin to minimize the occurrence of underflow of client playback buffers. The root cause of the instability problem is that, in Steady-State, a player goes through an ON-OFF activity pattern in which it overestimates the available bandwidth [1]. They propose a server-based traffic shaping procedure that can considerably lower such oscillations. Their procedure is only triggered when oscillations are identified, and so the shaping rate is dynamically adjusted. This ensures that the player receives the highest available video profile without being unstable. Using HTTP for video streaming significantly increases the request overhead due to the segmentation of the video content into HTTP resources [30]. This overhead becomes even more substantial when non-multiplexed video and audio segments are deployed. The authors investigate the request overhead problem by employing the server push technology in the new HTTP 2.0 protocol. They develop a set of push strategies that actively deliver video and audio content from the HTTP server without requiring a request for each individual segment.

Chunk scheduling with stateless bitrate selection causes feedback loops, bad bandwidth estimation, bitrate switches and unfair bitrate choices [12]. This paper, which portrays the FESTIVE control algorithm, confirms that numerous problems occur when multiple bitrate-adaptive players (adaptation over HTTP) share a bottleneck link [31]. It uncovers the fact that the feedback signal the player

Volume: 11 Issue: 02 Pages: 4236-4243 (2019) ISSN: 0975-0290

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 state to analyze commercial players: (1) schedule a video chunk for download, (2) select bitrate for chunk, and (3) estimate bandwidth. It identifies root causes of undesirable interactions with abstract state player framework and saw the need to guide the tradeoffs between stability, fairness and efficiency. As a result, 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 steps: (1) Randomized chunk scheduling: to avoid 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.

The authors in [18], 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) [5] 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. The authors illustrate a four-step state 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 [29] [10] [20] etc…, 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 sensitive to bandwidth drops.

III. OVERLAPPING ON PROBLEM

In a DASH model the video is pre-encoded and stored on the server. Each video stream is broken up into segments or chunks of seconds each. The streaming process for each client is divided into sequential segment downloading steps. Variable durations between consecutive segment requests is incorporated in the model. At the beginning of downloading of each sequence two important decisions are made: (1) the video bitrate of the next segment to be downloaded, and (2) the target inter-request time. The client also determines the time it takes to download the nth segment. If the download duration is shorter than the target delay, the client has an off time or wait time. Otherwise, the download starts immediately.

The output of adaptive video players following the DASH model can be either that the next segment download starts immediately after the current download is finished (buffering mode) or that the inter-request time is set to a fixed duration which forces OFF periods (steady-state mode). The main drawback of DASH is that when there are competing video flows, the estimated bandwidth based on the observed TCP throughput during the on-intervals does not represent the fair-share bandwidth. Possible use-cases that result from improper bandwidth estimation are: (1) where competing players overestimate their fair share, they may request video representations with a higher bitrate than the fair share which leads to network congestion. When TCP detects congestion, the players in turn estimate lower bandwidth than their previous fair share estimate and select a lower video bitrate level. This environment creates a repeating oscillatory scenario and results in instability. (2) where some players overestimate their fair share while others underestimate their fair share. In this situation players may converge to a stable equilibrium but without fair allocation of bandwidth. (3) where players estimate their fair share correctly but yet the total bandwidth capacity of the network is not utilized. This occurs as players may be requesting sub-optimal video bitrate levels.

We now give various multi-player scenarios of bandwidth allocation issues that arises when ON periods overlap. Let us consider a simple model. Assume that players are in steady state. There is a request for a new segment every T seconds. There are three adaptive players sharing a bottleneck with bandwidth B. We ignore the TCP model and accept a single connection getting the entire bandwidth B. Let us assume the network bandwidth share is equal. We let network bandwidth measures at the player be b_i. Thus, for equal sharing of bandwidth the following condition holds: $b_1=b_2=b_3$. Therefore, $b_1+b_2+b_3=B$, anytime during the streaming process.

First, take the case of non-overlapping ON periods. This is where all players ON periods are non-overlapping during segment download, see Figure 1. Each player measures the bandwidth as B. Thus, $b_1+b_2+b_3>B$. Consequently, each player overestimates their bandwidth share by a factor of three. Players request higher bitrates than the channel provides. Network congestion occurs. Players now measure a smaller bandwidth, which is less than their previous estimate. They switch back to a lower video

Volume: 11 Issue: 02 Pages: 4236-4243 (2019) ISSN: 0975-0290

bitrate by requesting a lower quality segment. The up and down movement in bitrate selection creates a repeating oscillatory scenario. Instability is the outcome.

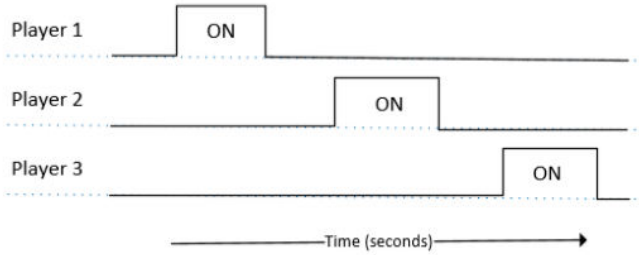

Fig. 3. Non-overlapping ON periods

In the second case, we look at ON periods that fall within each other. In multi-player streaming, the situation can arise, where the ON period of one player falls within the ON period of the other players, see Figure 2. This situation occurs if one or two player requests a segment with a low bitrate and another player requests a segment with a high bitrate. The players requesting the lower bitrate estimates a bandwidth of B/3, while the other player estimates a bandwidth that is more than B/3. This means that player three overestimates the bandwidth. This overestimation by one of the three players can still result in the three players converging to a stable equilibrium. However, the player who estimates the higher bandwidth share requests a higher video bitrate. This creates an unfair bandwidth allocation to all players. The players who request low bitrates will experience buffer underruns and poor video quality, due to flickering.

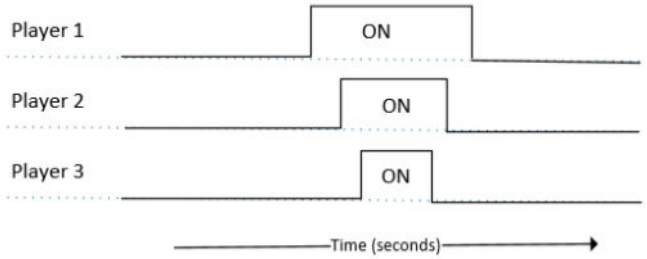

Fig. 4. ON periods within each other

For the third and final case we look at ON periods in perfect alignment. This situation occurs where ON periods of the three players align perfectly, see Figure 3. All players estimate a bandwidth of B/3. Thus, $b_1+b_2+b_3=B$. The three players estimate their bandwidth share correctly. However, bandwidth underutilization still occurs. To illustrate, suppose the video has two quality levels, q_1 and q_2. The ON periods of all three players align perfectly. This case is stable, if $b_1<B/3$, $b_2>B/3$, $b_3<B/3$ and $b_1+b_2+b_3<B$. However, all players request quality level q_1. This causes bandwidth underutilization, even though it is stable and fair. The players who obtain low bandwidth will experience buffer underruns and poor video quality, due to flickering.

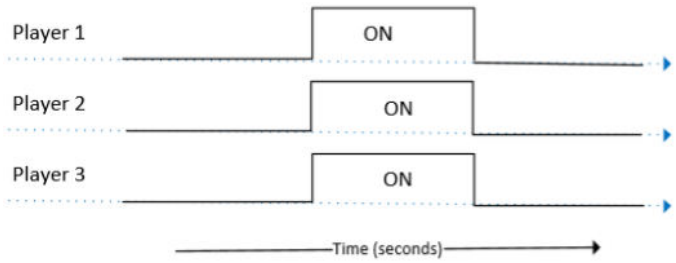

Fig. 5. Perfectly aligned ON periods

IV. COLLABORATIVE METHODOLOGY

When an adaptive video player starts a download, it broadcasts a 5-tuple to other players. This 5-tuple consists of (1) start download time, (2) requested segment download size, (3) inter-request time from last download, (4) player ID or IP address and (5) a message sequence number. During streaming this broadcast is performed by all players. Each player utilizes this information to determine (1) the approximate time the specified player would take to download the current segment, that is, the end download time, (2) the future inter-request time of the present download. This is simply the last inter-request time but could be made more accurate by taking multiple download samples for example 20. (3) the freshness of the data from other players which is obtained from the most recent broadcasted sequence numbers.

Using the 5-tuple information from other players each player performs cognitive download scheduling. It does this by using the known player start times. Having this knowledge enables a player to determine bottleneck conflicts if too many players are currently using the link. The player then adopts a strategy by waiting t seconds before it starts its download. This should reduce player contention for the bottleneck at that time stamp. Figure 6 shows an example. Here player i' has started a download and informed player i. Now player i can use this knowledge to delay its next segment download (assuming it does so with other players in the network being taken into consideration (not shown in diagram)).

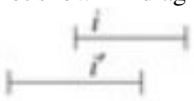

Fig. 6. Cognitive download scheduling

In addition, each player performs predictive downloading scheduling. It does this by utilizing the estimated player end times or inter-request times. A player waiting to download can use the estimated end download times or inter-request times to determine when to best start its next download. In this case instead of knowing the other players start download times the player may only determine that the other players are either still downloading or have just completed. If the other players are still downloading, then the player only has the estimated end download times. However, if the player had just finished downloaded the player can estimate the next start download time. In this way the player can delay its download by t seconds to reduce contention at the

Volume: 11 Issue: 02 Pages: 4236-4243 (2019) ISSN: 0975-0290

bottleneck link. Figure 7 shows player i downloading a video segment. Here player i' has to estimate the end download time of player i. It does this and starts it download a bit later (diagram does not show other players) to avoid network contention amongst players.

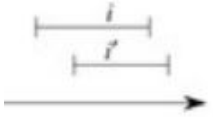

Fig. 7. Predictive download scheduling

The main constituents of an Linear Programming (LP) problem are the objective function and set of constraints. The constraints usually come from the environment, from which the pursuit of the objective becomes tangible. The environment contains principal factors (for instance restrictions, difficulties) obstructing an entity from fulling its desire or objective. The of four parts of any linear program are: (1) a set of decision variables, (2) parameters, (3) objective function and (4) a set of constraints. The contention resolution for each player is solved using a series on linear programming constraints with the optimal funcion maximizing the bandwidth. We call our solution collaborative DASH (C-DASH).

V. EXPERIMENTAL SETUP

A. Details of DASH-based Experiment Setup

A virtual network is setup on the same host machine creating a custom emulation framework. 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 [7] 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 player (cf. Figure 6). 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. Algorithms that uses the C-DASH protocol was tested and shown to work on both MPEG-DASH [28], and Apple HTTP Live Streaming (HLS) [25]. This makes it useful for video on demand (VOD) [22] and live streaming [19], for example, real-time video chats. However, the MPEG-DASH standard is used for testing in this research paper, because it makes the experiments more comparable to the ones in the research literature, for example, [8]. 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 [11]. 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.

B. QoE Metrics

i. The utilization metric [18] is defined as the aggregate throughput during an experiment divided by the available bandwidth in that experiment (cf. Equation 6, where tp_i is the throughput at time i and bw is the experimental available bandwidth).

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

ii. Instability: The instability for player i at time t is given in Equation 7, where $w(d) = k - d$ is a weight function that puts more weight on more recent samples. k is selected as 20 seconds.

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

iii. Inefficiency: The inefficiency at time t is given in Equation 8. Consider N players sharing a bottleneck link with bandwidth, w, with each player x, playing a bitrate, $b_{x,t}$, 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.

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

iv. Unfairness: Let $JainFair_t$ be the Jain fairness index (cf. Equation 10) calculated on the average received rates [18], r_i, (cf. Equation 9) at time t over all players. The unfairness at time t is defined as $\sqrt{1 - JainFair_t}$. A lower value implies a more fair allocation.

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

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

Volume: 11 Issue: 02 Pages: 4236-4243 (2019) ISSN: 0975-0290

v. Re-buffering ratio [5]: is the ratio of the time spent in
re-buffering and the total playtime of the stream Equation 11.

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

C. Videos

One of the videos (Elephant's dream) used in the experiments is shown below. Other videos are Sintel, Big Buck Bunny, The Swiss Account and Red Bull Playstreets.

TABLE I. ELEPHANT'S DREAM: VIDEO LEVELS, BITRATES AND RESOLUTIONS

Video level	Bitrate (kbps)	Resolution
l_0	46.0	320x240
l_1	91.0	320x240
l_2	131.0	320x240
l_3	180.0	480x360
l_4	222.0	480x360
l_5	261.0	480x360
l_6	328.0	480x360
l_7	382.0	480x360
l_8	523.0	854x480
l_9	594.0	854x480
l_{10}	796.0	1280x720
l_{11}	1000.0	1280x720
l_{12}	1200.0	1280x720
l_{13}	1500.0	1280x720
l_{14}	2100.0	1920x1080
l_{15}	2400.0	1920x1080
l_{16}	3000.0	1920x1080
l_{17}	3400.0	1920x1080
l_{18}	3800.0	1920x1080
l_{19}	4200.0	1920x1080

VI. RESULTS

The first experiment illustrates five players competing at a 20Mbps bottleneck link. Table 1 gives the results. C-DASH outperforms FESTIVE and the Conventional.

TABLE II. BOTTLENECK COMPETITION

	C-DASH	FESTIVE	Conventional
Utilization	0.88	0.76	0.67
Unfairness	0.053	0.079	0.194
Instability	0.163	0.250	0.310
Re-buffering	0.241	0.356	0.420
Inefficiency	0.111	0.237	0.327

The second experiment illustrates five players competing at a 20Mbps bottleneck link and stopping or pausing during the experiment. Table 2 gives the results. C-DASH outperforms FESTIVE and the Conventional.

TABLE III. VIDEO PLAYERS START, STOP AND PAUSE

	C-DASH	FESTIVE	Conventional
Utilization	0.84	0.76	0.63
Unfairness	0.098	0.127	0.216
Instability	0.178	0.232	0.378
Re-buffering	0.267	0.389	0.437
Inefficiency	0.123	0.265	0.356

The third experiment illustrates five players competing at a 100Mbps bottleneck link with increasing number of players up to 20. Table 3 gives the results. C-DASH outperforms FESTIVE and the Conventional.

TABLE IV. INCREASING PLAYERS

	C-DASH	FESTIVE	Conventional
Utilization	0.82	0.74	0.60
Unfairness	0.128	0.156	0.243
Instability	0.213	0.299	0.365
Re-buffering	0.287	0.390	0.443
Inefficiency	0.142	0.211	0.346

The fourth and final experiment illustrates five players competing at a 20Mbps bottleneck link in bandwidth varying conditions. Table 4 gives the results. C-DASH outperforms FESTIVE and the Conventional.

TABLE V. TIME-VARYING BANDWIDTH

	C-DASH	FESTIVE	Conventional
Utilization	0.77	0.72	0.59
Unfairness	0.159	0.198	0.267
Instability	0.265	0.315	0.477
Re-buffering	0.294	0.410	0.458
Inefficiency	0.180	0.274	0.398

VII. CONCLUSION

The performance of today's adaptive video streaming players (DASH) is severely hindered by overlapping ON-OFF patterns that occurs during a streaming session. High switching rates, freezing and skipping annoy users resulting in poor user-QoE. This paper attempts to overcome the ON-OFF problem by keeping players aware of each other's downloads and inter-request times. Using this a player is able to better predict future player actions and reschedule their downloads to produce least conflicts with others. If a player's start segment download overlaps with the end of one or more players' current downloads within a certain time t1 it waits until the download of

Volume: 11 Issue: 02 Pages: 4236-4243 (2019) ISSN: 0975-0290

others completes before starting its own download. Conversely if too many players are sharing the bottleneck link a player will hold of its download by a time t2 enabling others to move towards completion of their downloads. This reduces the overlap of its download with other players. Both these methods help reduce competition at the bottleneck and produces a more balanced sharing of network resources.

REFERENCES

[1] 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.

[2] Bagci, Kadir Tolga, Kemal Emrecan Sahin, and A. Murat Tekalp. "Compete or collaborate: Architectures for collaborative DASH video over future networks." IEEE Transactions on Multimedia 19, no. 10 (2017): 2152-2165.

[3] Bentaleb, Abdelhak, Ali C. Begen, and Roger Zimmermann. "SDNDASH: Improving QoE of HTTP adaptive streaming using software defined networking." In Proceedings of the 24th ACM international conference on Multimedia, pp. 1296-1305. ACM, 2016.

[4] Berners-Lee, Tim, Roy Fielding, and Henrik Frystyk. "Hypertext transfer protocol--HTTP/1.0." (1996).

[5] 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.

[6] Claeys, Maxim, Steven Latre, Jeroen Famaey, and Filip De Turck. "Design and evaluation of a self-learning HTTP adaptive video streaming client." IEEE communications letters 18, no. 4 (2014): 716-719.

[7] 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.

[8] 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.

[9] Georgopoulos, Panagiotis, Yehia Elkhatib, Matthew Broadbent, Mu Mu, and Nicholas Race. "Towards network-wide QoE fairness using openflow-assisted adaptive video streaming." In Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking, pp. 15-20. ACM, 2013.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] Khan, Koffka, and Wayne Goodridge. "Energy aware Ad-Hoc on demand multipath distance vector routing." International Journal of Intelligent Systems and Applications 7, no. 7 (2015): 50-56.

[15] 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.

[16] Khan, Koffka, and Wayne Goodridge. "S-MDP: Streaming with Markov Decision Processes." IEEE Transactions on Multimedia (2019).

[17] Kleinrouweler, Jan Willem, Sergio Cabrero, Rob van der Mei, and Pablo Cesar. "Modeling stability and bitrate of network-assisted HTTP adaptive streaming players." In Teletraffic Congress (ITC 27), 2015 27th International, pp. 177-184. IEEE, 2015.

[18] 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.

Volume: 11 Issue: 02 Pages: 4236-4243 (2019) ISSN: 0975-0290

[19] Lohmar, Thorsten, Torbjörn Einarsson, Per Fröjdh, Frédéric Gabin, and Markus Kampmann. "Dynamic adaptive HTTP streaming of live content." In 2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, pp. 1-8. IEEE, 2011.

[20] 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.

[21] Michalos, M. G., S. P. Kessanidis, and S. L. Nalmpantis. "Dynamic Adaptive Streaming over HTTP." Journal of Engineering Science & Technology Review 5, no. 2 (2012).

[22] 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.

[23] Oyman, Ozgur, and Sarabjot Singh. "Quality of experience for HTTP adaptive streaming services." IEEE Communications Magazine 50, no. 4 (2012): 20-27.

[24] 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.

[25] 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.

[26] Schmidt, Mark S., Praveen N. Moorthy, and Baozhou Li. "Server-side adaptive bit rate control for dlna http streaming clients." U.S. Patent Application 14/991,091, filed January 8, 2016.

[27] Sodagar, Iraj. "The mpeg-dash standard for multimedia streaming over the internet." IEEE multimedia 18, no. 4 (2011): 62-67.

[28] 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.

[29] 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.

[30] Wei, Sheng, and Viswanathan Swaminathan. "Cost effective video streaming using server push over HTTP 2.0." In Multimedia Signal Processing (MMSP), 2014 IEEE 16th International Workshop on, pp. 1-5. IEEE, 2014.

[31] 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.

[32] 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.

[33] Zhao, Shuai, Zhu Li, Deep Medhi, PoLin Lai, and Shan Liu. "Study of user QoE improvement for dynamic adaptive streaming over HTTP (MPEG-DASH)." In 2017 International Conference on Computing, Networking and Communications (ICNC), pp. 566-570. IEEE, 2017.

AUTHOR'S PROFILE

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.