QoE 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
Over recent years there has been a considerable shift, from quality of service (QoS) to quality of experience (QoE), when evaluating video delivery across networks. Hence, we first explore the need for this shift towards user-QoE in the video delivery ecosystem. Further, we investigate major QoE metrics researchers use in the evaluation of DASH users. We point out a huge problem with DASH beginning with its transport layer protocol. DASH utilizes Transmission control protocol (TCP) as the transport layer protocol. Thus, we give an overview of the mechanism of Transmission Control Protocol (TCP) and two mechanisms greatly impacting the streaming process: (1) TCP congestion mechanism and (2) TCP Fast Start. This leads us to investigate the impact of these TCP mechanisms on DASH players and consequently user-QoE.

Keywords - **QoS; user-QoE; video; TCP; transport layer; DASH; congestion; fast start.**

Date of Submission: Jan 26, 2018 Date of Acceptance: Feb 06, 2018

I. INTRODUCTION

The past ten year growth of Internet services creates a need for an increase in multimedia delivery across a standard medium. Consequently, today's Internet is abundant with applications utilizing video. Old-fashioned RTSP [27] streaming eventually evolves to HTTP-based streaming protocols. This shift leads to a better user-perceived QoE. Progressive download utilizes HTTP as a protocol and succeeds traditional streaming. Today, HTTP adaptive video streaming is the de facto standard. Real-time multimedia delivery possesses tight latency constraints, and data arriving too late is essentially useless. To create the illusion of motion, video frames play between 24-30 frames per second (fps). Each client media compression creates interdependence between packet contents and codecs. Thus, packet losses and late arrivals of video data can be detrimental. A combination of the inherent nature of network environments and transport protocol behaviors presents multimedia delivery challenges.
MPEG develops a HAS standard in 2012, Dynamic Adaptive Streaming over HTTP (DASH), [31], [13], [29]. It is also known as MPEG-DASH. DASH aims to provide video streaming service to users with dynamic network conditions and heterogeneous devices, with no interrupts. MPEG-DASH uses an application layer Adaptive BitRate (ABR) algorithm. The main goal of ABR algorithms is to prevent player's playout buffer under-run, while maximizing user-QoE, [34]. DASH ABR algorithms achieve this by adapting to dynamic changes in network conditions. The key differences between DASH and earlier protocols for multimedia streaming are: (a) Unlike earlier UDP-based schemes, DASH is built on top of TCP transport, (b) The player drives the algorithm. Depending on its ABR, the player typically requests video bitrates based on observed network conditions, hence regulating the server's transmission rate and (c) DASH requests and receives video data in terms of multi-second video segments, instead of a continuous stream of video packets. The player algorithm dynamically adapts the quality level in DASH-based systems. This adjustment focuses on the current network conditions. HAS addresses the weaknesses of RTSP, progressive download and regular segmented streaming. However, the server pre-processes media content. The media content is split into several small segments of a certain length, as in segmented streaming. The server now encodes each segment into a number of predefined qualities. The different versions of the segments are then distributed over one or more media servers (HTTP web servers). The video player decodes each representation of the segments individually. Also, the segment length determines the shortest video duration after which a quality adjustment can occur. Usually, this is 2 or 4 seconds in length.
This work consists of four sections. Section II discusses the need for quality of experience (QoE) in adaptive video streaming networks. Section III presents QoE metrics. Section IV gives an outline of TCP congestion and fast start mechanisms. Section V gives experimental evidence supporting the need for QoE improvements in present day DASH. Section VI explains the result and how the TCP mechanisms negatively affect user-QoE. Finally, the conclusion is given in Section VII.

II. THE NEED FOR QUALITY OF EXPERIENCE (QOE)

Current needs for the design of interactive, personal and socially connected platforms makes the video delivery landscape a rapidly evolving ecosystem. The Internet is a ubiquitous high-speed communication network supporting such evolution. However, the Internet does not provide any Quality of Service (QoS) or Quality of Experience (QoE) guarantees. Thus, applications dynamically adapt their requirements to the available QoS or QoE level.

Quality of experience (QoE) quickly emerges as a multidisciplinary field. QoE formulates itself from social psychology, cognitive science, economics and engineering science, cf. Figure 1. It focuses on understanding overall human quality requirements, [12]. The design of QoE culminates human expectations and quality needs. It's traditional counterpart is Quality of Service (QoS), [35], [6], [15], [1]. QoS parameters ensure service quality to end users. However, QoE incorporates aesthetic and parsimonious needs of individuals. QoS lacks these scales. Research in, [33] defines Quality of Experience (QoE): "the overall acceptability of an application or service, as perceived subjectively by the end user."

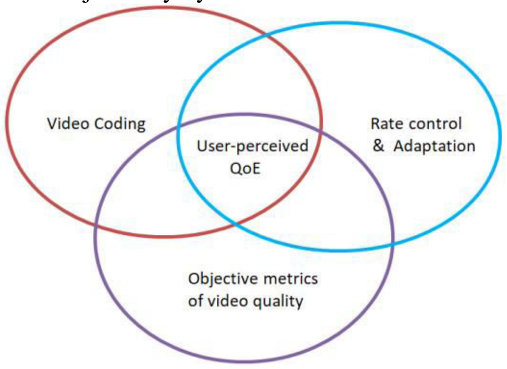


Fig. 1. Quality of Experience in adaptive video streaming

Measurable objective performance of technical system entities and subjective testing with people classifies network service evaluation scales. User-perceived QoS scales are subjective. They gather opinions from participants who rate the quality or change of an experience with an entity. Advantages of this approach are (1) it is very user-centric, (2) tests are well documented, (3) it contains standard repeatable components and (4) it can easily be compared to technical system performance. However, disadvantages, such as possible defects of individual human perceptual traits (example, ear infection during testing) and unconscious psychological factors, [2], may cause difficulties with this approach.

QoE gathers measures from end users, whereas QoS measures technical patterns affecting system performance. Subjective QoE extends beyond user-perceived media quality. It measures user satisfaction, [9]. Subjective user perception mainly focuses on questionnaires and rating scales, [3]. However, user behavior relying only on opinion is not reliable. Usually, objective measures concern the system and not the end users. QoS tools collect and monitor these objective measures. For example, in a network the buffer levels can be monitored. Here, these indirect technical measures infer QoE. Thus, user measures validate the relationship between technical measures and user behavior. However, user-QoE are also affected by other factors, cf. Figure 2.

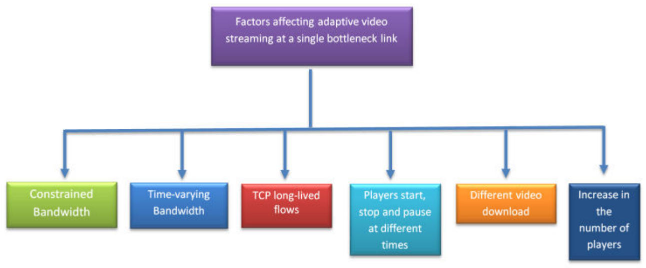

Factors affecting adaptive video streaming at a single bottleneck link
Constrained Bandwidth
Time-varying Bandwidth
TCP long-lived flows
Players start, stop and pause at different times
Different video download
Increase in the number of players

Fig. 2. Factors affecting QoE in adaptive video streaming

Computational models of media quality are objective because they model measurable technical parameters, [22]. They access cumulative variations in technical parameters, including those that affect quality. However, the value of these models reduces, as they must be continually validated against user test data, with the inclusion of new parameters.

Measures of QoE concerns user performance based on actual usage. There are numerous ways to gather objective QoE for laboratory and field tests or different types of services and user scenarios. A key point relates objective measures of QoE to user perception and beyond. This takes QoE into the realm of user experience. However, in the study of QoE, subjective user measure remains an important variable. Thus, a combination of objective and subjective QoE measurements better reflects the complex nature of QoE.

III. Quality Of Experience Metrics

In the context of HAS, TCP throughput reduces with mishaps along the network path, for example, packet losses and reordering. In such circumstances, video playback pauses for new video data. This severely impacts user-QoE. Many other factors affect user-QoE, such as quality of video and smoothness of playback, [16]. A Mean Opinion Score (MOS) of 1 ("Bad") to 5 ("Excellent") expresses QoE, [8]. Subjective or objective measures accumulates to mean opinion scores. However, objective measures, such as Peak-Signal-to-Noise-Ratio (PSNR) and Mean Square Error (MSE), are not suitable for HTTP video streaming. They only evaluate spatial video quality. Some application performance metrics include, [37]:

- Initial buffering time: T_{init} - measures the period between starting time of loading a video and starting time of playing it.
- Mean rebuffering duration: T_{rebuf} - measures the average duration of a rebuffering event.
- Rebuffering frequency: f_{rebuf} - measures rebuffering event frequency. In some cases, the amount of data in the video buffer decreases to low values. Playback pauses. The player enters a rebuffering state.

The development of QoE evaluation methodologies, performance metrics and reporting protocols play a key role in optimizing the delivery of HAS services, [18]. 3GPP DASH defines the following QoE metrics, in specification TS 26.247, [23], [32]:

- HTTP request/response transactions: logs the outcome of each HTTP request and corresponding HTTP response. For every HTTP request/response transaction, the player measures and reports:
    - Type of request (e.g., MPD, initialization segment, media segment)
    - Time of receiving HTTP request and corresponding HTTP response (in wall clock time)
    - HTTP response code
    - Contents in the byte-range-spec part of HTTP range header
    - TCP connection identifier
    - Throughput trace values for successful requests from HTTP request/response transactions. It is also possible to derive more specific performance metrics, such as (a) the fetch durations of the MPD, (b) initialization segment and (c) media segments.
- Representation switch events: reports a list of representation switch events taking place during a measurement interval. A representation switch event signals the player's decision to perform a representation switch to a new representation. As part of each representation switch event, cf. Figure 3, the player reports: (a) the identifier for the new representation, (b) time of the switch event (in wall clock time) and (c) media time of the earliest media sample played out from the new representation.

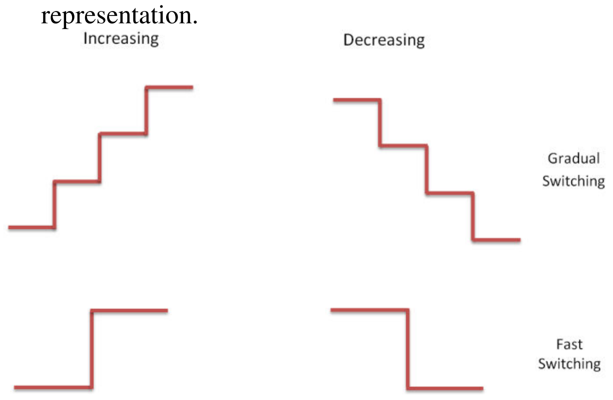


Fig. 3. Different types of switching are recorded during streaming.

- Average throughput: indicates the average throughput the player observes during the measurement interval. As part of the average throughput metric, the player measures and reports:
    - Total number of content bytes (i.e., the total number of bytes in the body of the HTTP responses) received during the measurement interval
    - Activity time during the measurement interval: time during which at least one GET request is still not completed
    - Wall clock time and duration of the measurement interval
    - Access bearer for the TCP connection based on the average throughput report
    - Type of inactivity (e.g., pause of presentation)
- Initial playout delay: signals initial playout delay at the start of the streaming. Initial delay is always present in adaptive video streaming services, as a certain quantity of data must be transferred before decoding and playback can begin. At first, the video playback delays more than necessary in order to fill the playout buffer. The playout buffer is an efficient tool. It handles short term throughput variations. However, the amount of playtime it initially buffers trades-off with actual length of the corresponding delay, [28]. Playout buffer measures the time from when the player requests the first media segment, to the time media is retrieved from the player buffer.
- Buffer level: provides a list of buffer occupancy level measurements carried out during playout, cf. Figure 4. As part of the buffer level metric, the player measures and reports the buffer level indicating the playout duration for which media data is available. This starts from the current playout time along with the time for measurement of the buffer level.

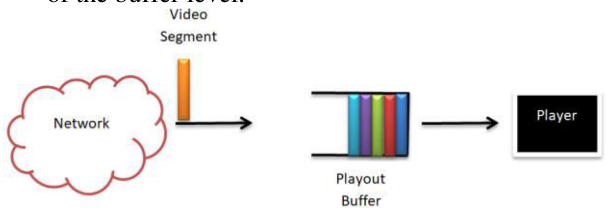


Fig. 4. Playout buffer fills with video segments.

- Play list: log a list of playback periods in the measurement interval. Each playback period is the time interval between a user action and the first occurring among: (a) the next user action, (b) the end of playback or (c) a failure that stops playback. The type of user action triggering playout may include: (a) a new playout request, (b) resume playout from pause or (c) user-requested quality change. For each playback period, the player measures and reports: (a) the identifiers of the representations, (b) rendering times (in media time) and (c) rendering durations, (d) playback speed relative to normal playback speed (e.g., to track trick modes, such as fast forward or rewind) and (e) reasons for interrupts of continuous playback for this representation (e.g., due to representation switch events, rebuffering, user request, end of period, media content and metrics collection period).

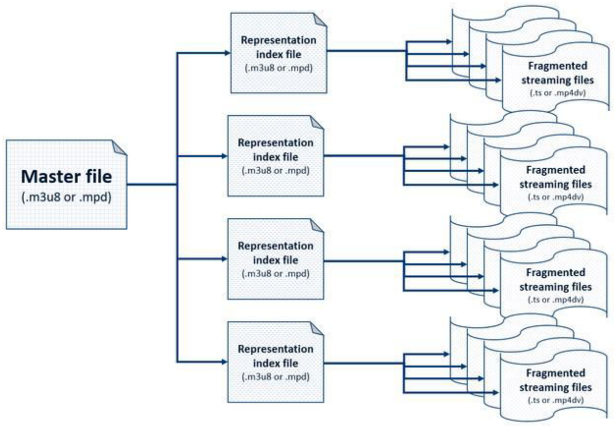
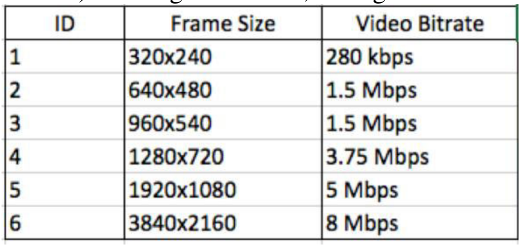


Fig. 5. Adaptive video streaming file fragmentation. [17]

- MPD information: reports information on the media presentations. It obtains information from the MPD, so that servers, without direct access to the MPD, can learn the media characteristics. The player reports media representation attributes on bit rate, resolution, quality ranking and codec-related media information, (including profile and level) utilizing this metric, cf. Figure 5.



ID	Frame Size	Video Bitrate
1	320x240	280 kbps
2	640x480	1.5 Mbps
3	960x540	1.5 Mbps
4	1280x720	3.75 Mbps
5	1920x1080	5 Mbps
6	3840x2160	8 Mbps

Fig. 5. Some information an MPD file contains.

Stalling stops video playback because the playout buffer underruns or becomes low/empty, [28]. During streaming a case may occur, where the video streaming application utilizes more data than incoming video bitrates. The playout buffer becomes smaller. However, the video eventually stops, when there is insufficient data in the playout buffer, for playback to continue. The length of an interruption guides re-buffering. Users experience longer stalling durations, with large playtime buffering. Viewers prefer a scenario with a single long freeze event, when they compare it to one with frequent short freezes, [21]. Research in [7] observes overall video quality decreases as duration of the impairment increased. Further, video stalls are worse than frame rate reduction. They observe video stalls at irregular intervals are worse than those at periodic intervals. Bandwidth fluctuations can cause playout interruptions of video. Packet loss through buffer overruns (buffer is full, so receipt of packets results in drops occurring) lead to retransmission of some packets and consequent delays. Playout interruptions annoy users and delays create bad effects, such as flickering. These events should be taken into consideration for QoE estimation.
Current adaptive streaming approaches, [10] [4] [14], focus on the following five QoE metrics, cf. Figure 6.

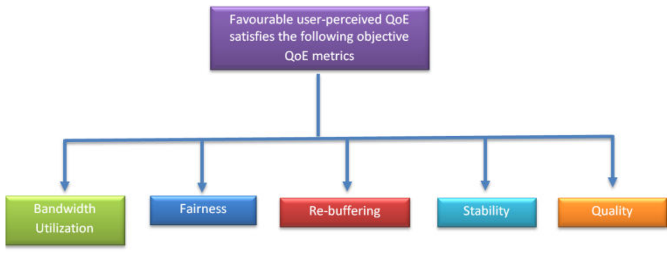

Favourable user-perceived QoE satisfies the following objective QoE metrics
Bandwidth Utilization
Fairness
Re-buffering
Stability
Quality

Fig. 6. Favorable user-perceived QoE for adaptive video streaming.

IV. A Transport Layer Protocol: TCP

Applications requiring data packet delivery guarantees utilize Transmission Control Protocol (TCP) as their transport layer protocol, [20]. It is a sliding window protocol, [11]. TCP handles timeouts and retransmissions. TCP establishes a full duplex virtual connection between two endpoints. An IP address and TCP port number defines each endpoint. The operation of TCP as a finite state machine is made possible in its implementation. The server transfers byte stream segments across one or more network channels. The window size determines the number of data bytes sent, before an acknowledgment (ACK) from the receiver is necessary. We discuss two important algorithms TCP utilizes: (1) Congestion avoidance and (2) Fast start. We select these as they have huge impacts on video streaming. We explain how these algorithms work. This would later enable an understanding of TCP interactions with video flows during streaming.

A. Congestion Avoidance

The TCP Congestion Avoidance algorithm, [25], heavily influences the behavior of Internet traffic dynamics. Congestion occurs when the receiver consumes data at a slower rate than the sender transmits. The buffers at the receiver overflows. This results in lost packets, which the receiver retransmits. Congestion occurs at a router, when the capacity of the output link is less than the sum of all inputs (assume multiple input links and one outgoing link). Congestion avoidance deals with lost packets.
There are two ways of indicating that a packet was lost: (1) a timeout occurs and (3) receipt of duplicate ACKs. Congestion avoidance and slow start are two separate algorithms, each with its own goal. If TCP detects congestion, it slows down transmission packet rates into the network. TCP then call on slow start to resume until congestion occurs, again. Thus, in practice one implements congestion avoidance and slow start together.
A connection maintains two variables, when combining congestion avoidance and slow start: (1) a congestion window, *cwnd* and (2) a slow start threshold size, *ssthresh*. These algorithms operate as follows, [30]:

1. Initialization for a given connection sets *cwnd* to one segment and *ssthresh* to 65535 bytes.
2. The TCP output routine never sends more than the minimum of *cwnd* and window size the receiver advertises.
3. As congestion occurs (a timeout or the reception of duplicate ACKs), *ssthresh* saves one-half of the

current window size (the minimum of *cwnd* and receiver's window advertise size, but at least two segments). Additionally, if a timeout indicates congestion, *cwnd* is set to one segment (slow start).

4. As new data is acknowledged by the other end, increase *cwnd*. However, the way it increases depends on whether TCP is performing slow start or congestion avoidance.

TCP is in slow start if the *cwnd* is less than or equal to *ssthresh*. Otherwise, TCP is using the congestion avoidance algorithm. The slow start algorithm continues as long as TCP is less than half of the point. That is, the previously set value, when congestion occurs. Then, congestion avoidance commences.

The congestion avoidance algorithm increments the *cwnd* by segment size times segment size divided by *cwnd*, every time it receives an ACK, where *cwnd* is the command window size in bytes. The growth rate is linear. Thus, the increase in *cwnd* is at most one segment each round-trip time. This occurs regardless of how many ACKs the algorithm receives in the RTT. However, slow start increases *cwnd* by the number of ACKs it receives in a RTT. Figure 7 depicts the congestion avoidance mechanism. Shown also is fast recovery or start, which is discussed next.

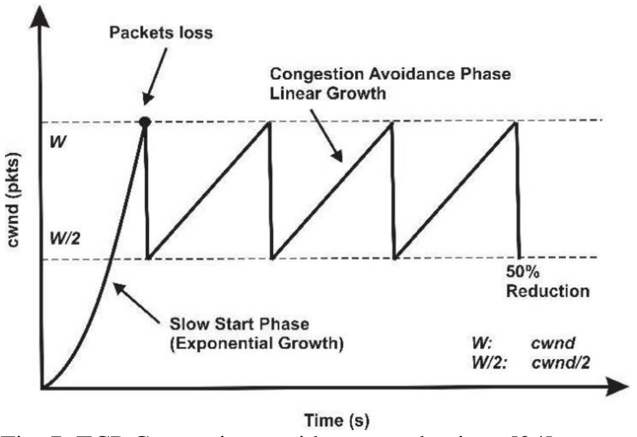


Fig. 7. TCP Congestion avoidance mechanism. [24]

B. Fast Start

TCP fast start [19] speeds up short Web transfers. The sender caches network parameters, which avoids paying a slow start penalty for each download. Stale cache information degrades performance. Fast start enables TCP connections to reuse network information from the recent past. Therefore, it does not repeat the slow start discovery algorithm each time, especially ones transferring small amounts of data between pauses. Cached information includes:

- Congestion window size *(cwnd)*
- Slow start threshold (*ssthresh*)
- Smoothed round-trip time (*srtt*)
- Variance of the smoothed round-trip (*rttvar*)

Two important goals, [19] are:

- If the cached information is still valid, fast start should help improve performance. However, if this information is stale (for instance, because of a sudden surge in network load), fast start should not result in worse performance than if TCP utilizes the standard slow start algorithm in the first place.
- The performance gains of fast start should not be at the expense of other connections. It is okay for other connections to suffer, once fast start tries to use only its share of the bottleneck bandwidth. But, not because the fast start connection is being too aggressive.

It is very difficult to meet these objectives, without increasing the network complexity, for example, connection-to-connection state in routers. However, TCP fast start does its best with a simple drop priority mechanism, which routers implement.

We now present a simple example showing the growth rate of packets sent over a channel using TCP fast start. Initially, the sender transmits one segment and waits for an ACK. Let us assume the ACK is sent successfully from the receiver to the sender. When the sender receives an ACK, it increments the congestion window one, that is, from one to two. Then, two the sender transmits two segments. Again, let us assume that the ACKs for these two segments are sent successfully from the receiver to the sender. The sender increases the ACK count to four and the congestion window to four, see Figure 8 (slow start, but increase rapidly so really fast start as labelled in the figure). This process produces what seems to be an exponential growth rate. However, the receiver may delay its ACKs, generally sending one ACK for every two segments it receives. Thus, it is not exactly exponential. In addition, at a point in transmission, the network, intermediate network device or router reaches their maximum capacity. The intermediate network device starts to discard packets. The sender now realizes the congestion window is too large. A simple way to view TCP: whenever there is congestion, the input rate decreases, and whenever there is under-utilization input rate increases.

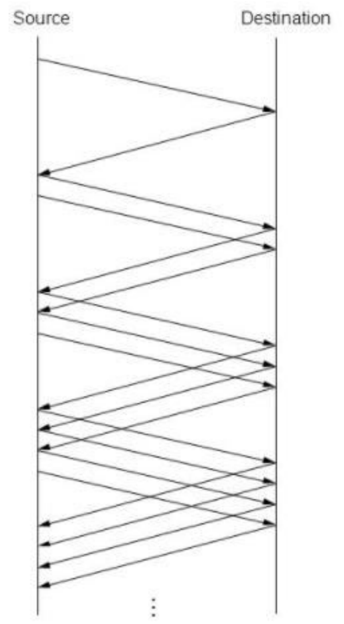


Fig. 8. TCP Fast Start: increase by one packet per ACK

V. Experiments

The following experiments use the conventional DASH controller and measures five DASH-based performance metrics. All experiments are hosted on a Windows 10 machine, with the following specifications: Intel(R) Core(TM) i7-5500U CPU 2.40GHz processor, 16.00 GB physical memory and an Intel(R) HD Graphics processor. A virtual network is setup on the Windows 10 machine for the emulation test bed. Our setup consists of a video server running Ubuntu (HTTP server), a router running FreeBSD (Home Router) and two real network players (HAS client) hosted on Ubuntu, cf. Figure 9.

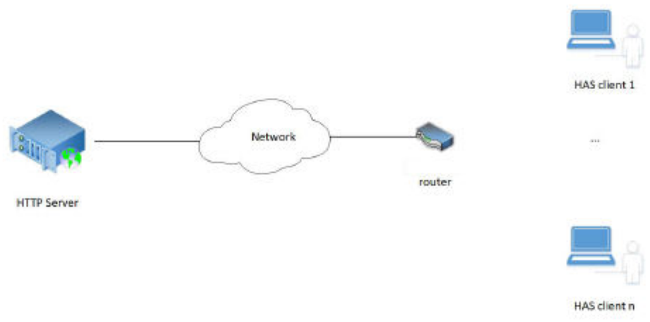


Fig. 9. Experiment testbed setup.

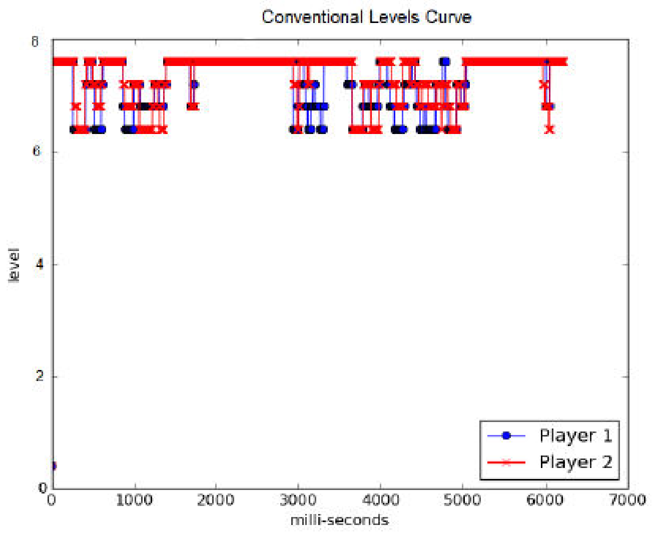


Fig.10. Quality levels: Conventional DASH players.

The result of two conventional players competing at a 5Mbps bottleneck link is shown on Figure 10. The players concurrently download the Elephant's dream video [5]. As shown the players are not able to maintain high quality video and so the user experience degrades at certain time intervals. The reason for this is discussed in section VI.

VI. Discussion: TCP and DASH

Generally, the sender controls the aggressiveness of TCP. However, the receiver throttles traffic in DASH-based systems, cf. Figure 11. TCP tries to increase bandwidth utilization and avoid congestion. However, a DASH player ensures no video playback interruptions by pre-fetching and buffering sufficient video segments. Thus, the DASH player continuously increases its receiver window during ON periods to obtain as much video segments as possible, [4]. The sender transmits as much traffic as possible in bursts. Segments transmissions occur independently of each other. This happens until the player's playout buffer has enough video segments, which either changes the player to an OFF period or the sender incurs TCP packet losses. The connection resets between ON-OFF periods, [10]. This causes extreme bursty traffic (cf. Figure 12) and TCP inefficiency, [38]. The result is unstable video playback quality and unfair sharing of network resources.

VII. Conclusion

We introduce DASH-based video delivery. We show the need for QoE in evaluating DASH-based systems. We highlight the main QoE metrics researchers utilize in evaluating DASH-based systems. We explore the working mechanisms of TCP: (1) Congestion and (2) Fast Start. We present an experiment with competing DASH players. The experiment shows poor QoE for existing users. Further, we explain how the two TCP mechanisms result in this poor user-QoE for DASH users.

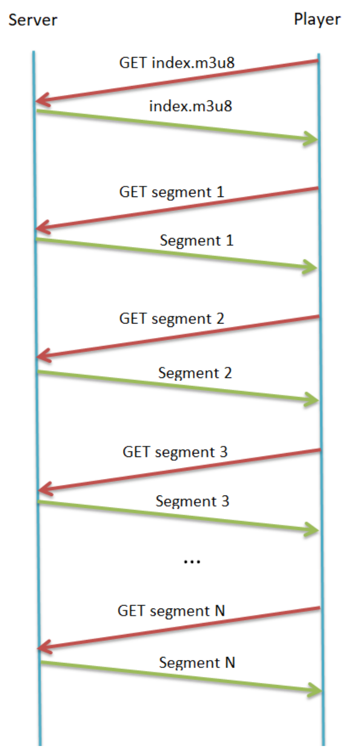


Fig.11. The player first makes a GET request for the MPD file containing data about the video segments it wants to download. After this initiation the player makes requests for successive segments. This is on a segment by segment basis and gives the player the option to select the next segment based on certain "desired" criteria.

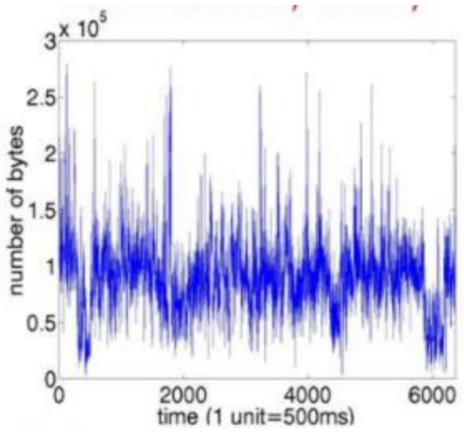


Fig. 12. Bursty network traffic. [26]

REFERENCES

[1] Andrews, M., K. Kumaran, K. Ramanan, A. Stolyar, P. Whiting, and R. Vijayakumar (2001). Providing quality of service over a shared wireless link. IEEE Communications magazine 39 (2), 150-154.

[2] Brooks, P. and B. Hestnes (2010). User measures of quality of experience: why being objective and quantitative is important. IEEE network 24 (2).

[3] Brooks, P. and B. Hestnes (2010). User measures of quality of experience: why being objective and quantitative is important. IEEE network 24 (2).

[4] De Cicco, L., V. Caldaralo, V. Palmisano, and S. Mascolo (2013). Elastic: a client-side controller for dynamic adaptive streaming over http (dash). In 2013 20th International Packet Video Workshop, pp. 1-8. IEEE.

[5] De Cicco, Luca, and Saverio Mascolo. "An experimental investigation of the Akamai adaptive video streaming." In Symposium of the Austrian HCI and Usability Engineering Group, pp. 447-464. Springer, Berlin, Heidelberg, 2010.

[6] Ferguson, P. and G. Huston (1998). Quality of service: delivering QoS on the Internet and in corporate networks, Volume 1. Wiley New York.

[7] Huynh-Thu, Q. and M. Ghanbari (2008). Temporal aspect of perceived quality in mobile video broadcasting. IEEE Transactions on Broadcasting 54 (3), 641-651.

[8] ITU-T RECOMMENDATION, P. (1998). Subjective audiovisual quality assessment methods for multimedia applications.

[9] Jain, L. P., W. J. Scheirer, and T. E. Boult (2004). Quality of experience. In IEEE multimedia. Citeseer.

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

[11] Kontaki, M., A. N. Papadopoulos, and Y. Manolopoulos (2007). Adaptive similarity search in streaming time series with sliding windows. Data & Knowledge Engineering 63 (2), 478-502.

[12] Laghari, K. U. R. and K. Connelly (2012). Toward total quality of experience: A qoe model in a communication ecosystem. IEEE Communications Magazine 50 (4).

[13] Lederer, S., C. Müller, and C. Timmerer (2012). Dynamic adaptive streaming over http dataset. In Proceedings of the 3rd Multimedia Systems Conference, pp. 89-94. ACM.

[14] 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.

[15] McDysan, D. E. (2000). QoS & tra-c management in IP & ATM networks, Volume 18. McGraw-Hill New York, NY, USA.

[16] Mok, R. K., E. W. Chan, and R. K. Chang (2011). Measuring the quality of experience of http video streaming. In Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on, pp. 485-492. IEEE.

[17] Nwamba, C. (2017). How to Implement Smooth Video Buffering for a Better Viewing Experience [Blog post]. Retrieved from https://cloudinary.com/blog/how_to_implement_smooth_video_buffering_for_a_better_viewing_experience .

[18] Oyman, O. and S. Singh (2012a). Quality of experience for http adaptive streaming services. IEEE Communications Magazine 50 (4).

[19] Padmanabhan, V. N. and R. H. Katz (1998). Tcp fast start: A technique for speeding up web transfers.

[20] Postel, J. (1981). Transmission control protocol.

[21] Qi, Y. and M. Dai (2006). The effect of frame freezing and frame skipping on video quality. In Intelligent Information Hiding and Multimedia Signal Processing, 2006. IIH-MSP'06. International Conference on, pp. 423-426. IEEE.

[22] Recommendation, I.-T. (2001). Perceptual evaluation of speech quality (pesq): An objective method for end-to-end speech quality assessment of narrowband

telephone networks and speech codecs. Rec. ITU-T P. 862.

[23] Reznik, Y., E. Asbun, Z. Chen, and R. Vanam (2013, April 23). Method and apparatus for smooth stream switching in mpeg/3gpp-dash. US Patent App. 13/868,968.

[24] Rhee, Injong, Volkan Ozdemir, and Yung Yi. TEAR: TCP emulation at receivers-flow control for multimedia streaming. NCSU Technical Report, 2000.

[25] Ryu, S., C. Rump, and C. Qiao (2003). Advances in internet congestion control. IEEE Communications Surveys & Tutorials 5 (1).

[26] Sarvotham, Shriram, Rudolf Riedi, and Richard Baraniuk. "Connection-level analysis and modeling of network traffic." In Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement, pp. 99-103. ACM, 2001.

[27] Schulzrinne, Henning. "Real time streaming protocol (RTSP)." (1998).

[28] Seufert, M., S. Egger, M. Slanina, T. Zinner, T. Hoÿfeld, and P. Tran-Gia (2015). A survey on quality of experience of http adaptive streaming. IEEE Communications Surveys and Tutorials 17 (1), 469-492.

[29] Sodagar, I. (2011). The mpeg-dash standard for multimedia streaming over the internet. IEEE MultiMedia 18 (4), 62-67.

[30] Stevens, W. R. (1997). Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms.

[31] Stockhammer, T. (2011a). Dynamic adaptive streaming over http: standards and design principles. In Proceedings of the second annual ACM conference on Multimedia systems, pp. 133-144. ACM.

[32] Stockhammer, T. (2011b). Ts 26.247 transparent end-to-end packet-switched streaming service (pss). Progressive Download and Dynamic Adaptive Streaming over HTTP, 3GPP.

[33] Takahashi, A., D. Hands, and V. Barriac (2008). Standardization activities in the itu for a qoe assessment of iptv. IEEE Communications Magazine 46 (2).

[34] Tavakoli, S., J. Gutiérrez, and N. Garcia (2014). Subjective quality study of adaptive streaming of monoscopic and stereoscopic video. IEEE Journal on Selected Areas in Communications 32 (4), 684-692.

[35] Wang, Z. and J. Crowcroft (1996). Quality-of-service routing for supporting multimedia applications. IEEE Journal on selected areas in communications 14 (7), 1228_1234.

[36] Wang, Z., S. Banerjee, and S. Jamin (2003). Studying streaming video quality: from an application point of view. In Proceedings of the eleventh ACM international conference on Multimedia, pp. 327-330. ACM.

[37] Wang, Z., S. Banerjee, and S. Jamin (2003). Studying streaming video quality: from an application point of view. In Proceedings of the eleventh ACM international conference on Multimedia, pp. 327-330. ACM.

[38] Zhou, L., X. Wang, W. Tu, G.-M. Muntean, and B. Geller (2010). Distributed scheduling scheme for video streaming over multi-channel multi-radio multihop wireless networks. IEEE Journal on Selected Areas in Communications 28 (3).

Biographies and Photographs

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.