EPIQ: A NEW ACTIVE QUEUE MANAGEMENT PARADIGM FOR PACKET VIDEO

Hothaifa Al-Qassab and Hayder Radha, IEEE Fellow
Department of Electrical and Computer Engineering
Michigan State University
East Lansing, Michigan 48824
{alqassab, radha}@msu.edu

ABSTRACT

In this paper, we introduce a new Active Queue Management (AQM) paradigm and a corresponding congestion control mechanism for a novel video-streaming framework. We call the proposed framework Erasable Packets within Internet Queues (EPIQ). EPIQ exhibits best-effort complexity within the network, similar to the networking complexity of HTTP-based Adaptive Streaming (HAS); yet EPIQ can provide optimal quality-of-experience performance similar to the performance of high-complexity multi-queue solutions. EPIQ exploits the inherent multi-priority nature of video to deliver optimal quality-of-experience without requiring any additional overhead in terms of redundant packets or network resources such as multiple queues and without the need to maintain state information about flows. The novelty of EPIQ is rooted in its packetization. Each EPIQ packet consists of multiple segments, and where each segment carries different priority video content. Under congestion, a network router can simply partially erase only small portions of the EPIQ packets instead of dropping complete packets. We show through an analytical model and extensive simulations that this new paradigm in video streaming provides significant improvements over HAS solutions in terms of quality-of-experience metrics. These significant improvements are achieved while maintaining fairness, stability, and low-complexity within the network.

***Index Terms*—**Video streaming, network congestion control, HTTP-based adaptive streaming

1. INTRODUCTION

Currently deployed HTTP-based Adaptive Streaming (HAS) [1] solutions employ well-established protocols such as TCP and HTTP, and they use standard best-effort Internet services. Consequently, the shortcomings of traditional streaming approaches in terms of using network resources, and their inherent dependency on the best-effort Internet have been well studied [2][3][4]. These shortcomings have been exaggerated by the tremendous growth of Internet video traffic, and this led to a noticeable increase in the severity and frequency of network congestion events [5][6].

More importantly, HAS based streaming solutions, in general, and Dynamic Adaptive Streaming over HTTP (DASH), in particular, lead to oscillatory behavior that is highly undesirable in terms of users' Quality of Experience (QoE)[6]. Meanwhile, consistent high-quality QoE video can be realized using multi-priority queuing within the networks. This led to the development of the DiffServ[7], which arguably represents state-of-the-art multi-priority video queuing solution. Under DiffServ, the video stream is divided into sub streams that are marked with different dropping precedence. As packets travel in the DiffServ network, low priority packets are dropped during congestion. To reduce queue management and overall network complexity, the number of DiffServ priority queues is limited to only a few. However, despite the broad support for DiffServ within network routers, it has not been utilized in a significant way, and certainly not at the level that HAS based solutions have been employed and used.

In this paper, we introduce a novel Active Queue Management (AQM) paradigm and a corresponding congestion control mechanism for a new packet-video streaming framework. The proposed framework is capable of delivering optimal and consistent quality-of-experience similar to the quality promised by multi-priority video queuing solution while requiring minimal low-complexity network support similar to the support provided by standard and best-effort Internet. We refer to our framework as Erasable Packets within Internet Queues (EPIQ). There are two important aspects of the proposed EPIQ framework. First, EPIQ is based on a novel packetization of the video content in a way that exploits the inherent multi-priority nature of video. In short, each EPIQ packet consists of multiple segments, and where each segment consists of video data of a different priority (see Fig. 1). The highest priority segment is placed next to the packet header whereas the lowest priority segment is placed at the tail of the packet. This novel packetization enables the second major aspect of the proposed EPIQ framework. Under congestion, a simple AQM mechanism can be realized where a network router could drop (or erase) only small portions located toward the tail of EPIQ packets instead of dropping complete packets as traditionally done by current routers. It is important to note that EPIQ does not require multi-priority queues or highly complex marking algorithms.

In this paper, we: (a) develop an analytical model for EPIQ performance; (b) present an AQM algorithm that is based on the popular RED mechanism, and hence, we refer to it as EPIQ-RED; (c) develop an EPIQ-based TCP-Friendly Rate Control (TFRC) algorithm that we call EPIQ-TFRC; (d) conduct an extensive set of simulations using ns-2 to evaluate both network performance parameters and quality-of-experience video metrics for EPIQ. We compare the performance of EPIQ with DASH and with Multi-Priority Queue (MPQ) solution like DiffServ. Our analysis and simulation results validate EPIQ's capability in delivering significant improvements over DASH while maintaining fairness, stability, and low network-complexity and without the burden that DiffServ and similar MPQ solutions require from the network.

The remainder of the paper is organized as follows. Section 2 provides a high level background and a brief outline of related

work. Section 3 presents the EPIQ framework and its analytical models. Section 4 describes the detailed aspects of EPIQ including the AQM and congestion control mechanisms. Section 5 presents our extensive ns-2 based simulation results. Section 6 provides a brief summary of the paper and some concluding remarks.

2. BACKGROUND AND RELATEDWORK

In this section we briefly review relevant literature in the areas of priority video transmission and HTTP Streaming.

2.1. Priority Dropping

Priority-based video transmission is a vast area that emerged from the early days of video streaming. Here, we briefly outline a very short list of key solutions that are relevant to the proposed EPIQ framework. In particular we focus on solutions that are based on Quality-of-Service (QoS) mechanisms that do not require any redundant packets.

One of the first successful efforts to introduce QoS guarantees was DiffServ. Its scalable and distributed mechanism made it the most widely deployed QoS supported technology nowadays [8]. One early example for exploiting unequal priority of video packets using DiffServ was proposed in [7][9]. To take advantage of DiffServ, the packets of a video stream are divided and marked using a rather sophisticated mechanism to adhere to certain ISPs' policies. The authors of [10] proposed a DiffServ, UMTS and DVB cross-technology system solution that provides priority dropping to video streams across all systems. To overcome DiffServ packet marking polices, authors of [11] proposed a simple differentiated service by supporting multi service profiles (e.g. low delay) for video streams.

Other mechanisms were proposed to improve priority-based video transmission using AQM. AQM with priority dropping (AQM-PD) provides the desired video stream priority packet dropping. The authors in [12] proposed PID-PD, a single queue algorithm that employs proportional-integral-derivative (PID) controller to perform AQM and a sorting algorithm that sorts packets according to their dropping precedence. An analytical model of best effort video transmission is proposed in [13]. The model shows the importance of priority dropping over random dropping in layered video streams. In addition, the author proposed a multi queues mechanism that has different dropping precedence and congestion control algorithm to mark the video packets according to network condition.

2.2. HTTP Streaming

One of the most popular video streaming solutions today is based on HAS with an international standard known as MPEG-DASH)[14]. HAS client uses HTTP signals to request a video chunk with a certain bit-rate from the server. The user estimates the TCP throughput, which is used to carry video stream data, to select the corresponding video chunk from the server.

There have been several works that studied the performance of commercial HAS solutions. In [5] the authors summarize the drawbacks of the conventional HAS systems. These drawbacks include: (a) Unfairness among video and non-video flows; (b) Oscillation in the requested video bit rate; and (c) Inefficient video bit-rate selection; and a forth drawback, (d) large delays, especially for live streaming, was studied in [15].

To improve the performance of HAS, several approaches were proposed. The authors of [16] proposed a four-step model (estimate, smooth, quantize and schedule) to design a rate adaptation algorithm. Several studies proposed using control theoretic approaches to govern the rate adaptation process. For example, in [17] proportional integral derivative (PID) controller is used. A stochastic Markov Decision Process (MDP) approach was proposed in[18] to maximize video quality. Scalable Video Coding (SVC) was proposed by [19] to minimize the required storage for video files and to increase the supported video rates. In [20], the authors proposed a scheduling algorithm to deliver different SVC priority packets over HTTP. The authors in [15] suggested using new HTTP/2 to reduce the HTTP delay.

3. EPIQ SYSTEM DESIGN AND ANALYSIS

In this section, we also develop an analytical model that shows the vitality of our approach.

3.1. EPIQ System Overview

EPIQ exploits the well-known dependency among video packets and the inherent multi-priority nature of compressed video content to achieve its goals. EPIQ consists of Partially Erasable Packets (PEP). A PEP structure is conceptually simple, and it is illustrated in Fig. 1. For any set of compressed video frames, we can identify different levels of priority for the content of such set. These levels could correspond to different picture types (i.e., I, P, and B frames) of a non-scalable video stream; or, they could correspond to different layers of a scalable video frame (i.e., a base-layer and multiple enhancement layers). Whatever the case, a video stream can be divided into different priority levels as shown in Fig. 1.

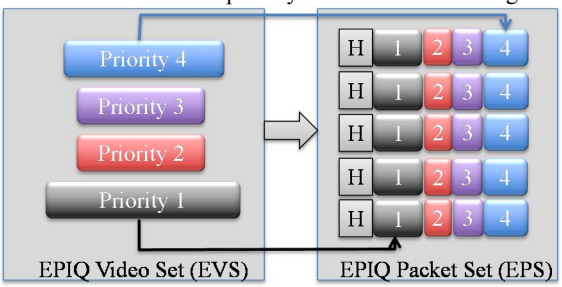

Fig. 1. The mapping of multi-priority video content from an EPIQ Video Set (left) to Partially Erasable Packets of an EPIQ Packet Set (right).

EPIQ Video Set (EVS) represents a set of prioritized video content. In general, EVS could represent any set of video frames that can be partitioned into multiple priorities as shown in Fig. 1. EPIQ Packet Set (EPS) represents the set of PEP packets that carry the content of one EVS. Thus, we divide each EVS priority level into different segments that we distribute among the packets of an EPS. Although not necessarily the case, in Fig. 1, each EVS priority level is divided into equal-size segments, which are uniformly distributed among the EPS packets.

The novelty and power of the proposed EPIQ packet structure is that it enables a simple and best-effort-like queue management mechanism while mimicking and achieving the benefits of MPQ management approaches. During congestion, instead of dropping a complete packet, a router can now simply erase an arbitrary portion of a PEP packet starting from its tail. It is important to note that the router does not have to be aware of the number of priority segments or their sizes within a PEP packet.

3.2. Stochastic Model

In this section, we present our proposed system analytical model. Also, we develop a Multi-Priority Queue (MPQ) analytical model using Best Effort (BE) model. These analytical models are

intended to provide an insight into how our system performs in comparison with traditional best-effort and Multi-Priority Queue systems. More importantly, we further compare the models with the ideal case of achieving channel capacity.

It is important to note that we refer to BE and a MPQ as conventional system because both systems employ packet dropping. Let p be the probability of dropping a packet within the network under traditional best-effort streaming (e.g., using HAS). If the average size of a packet is S bytes, then p is also the probability of dropping S bytes. Under the proposed EPIQ framework, p becomes the probability of erasing the same number (S) bytes from PEP packets in the network queue. Hence, in either case, p represents the average loss rate. Fig. 2 shows an example for mapping a traditional scalable video frame with N_P packets into an EPS using the same number N_P of Partially Erasable Packets (PEPs). Let B_T be the size of one EPIQ packet set in bytes. Since S is the packet size in bytes, for both the traditional and EPIQ packetization. Let B_{lost} be the number of bytes that would be lost in a conventional system due to packet dropping.

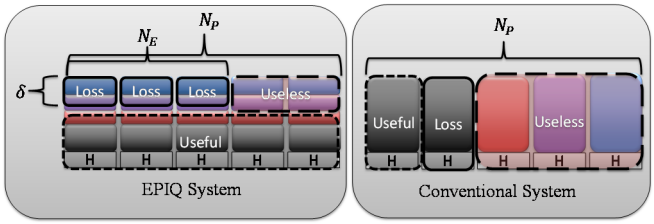

Fig. 2. EPIQ vs. conventional

The expected value of the number of bytes to be lost within an EPIQ packet set can be written as:

$$E[B_{lost}] = p \cdot B_T \tag{1}$$

Similarly, we use B_{lost} to represent the total number of bytes lost from an EPIQ packet set due to partial erasing. Note that in a conventional system, B_{lost} corresponds to the number of bytes lost due to dropping complete packets.

To capture our Partial Erasing (PE) process for distributing the loss B_{lost} bytes over multiple packets, we define two parameters N_E and $\bar{\delta}$. Fig. 2 shows an illustrative example of the Partial Erasing process. Let (N_E) be the number of packets (from a given EPIQ packet set) that encounter partial erasing.

The second parameter in this analysis, $\bar{\delta}$, represents the average number of bytes that are erased from the N_E packets selected for partial erasing. Consequently, for an EPIQ based loss event, on average, we have the following:

$$\bar{\delta} = E\left[\frac{B_{lost}}{N_E}\right] \tag{2}$$

Now, we would like to compute the amount of useful data received by the decoder. Let B_U be the useful data (measured in bytes) recovered from one EPIQ packet set (EPS), which consists of N_P PEP packets. Consequently, we have the following:

$$E[B_U] = B_T - \bar{\delta} \cdot N_P = B_T - E\left[\frac{B_{lost}}{N_E}\right] \cdot N_P \tag{3}$$

Using first order approximation of $E\left[\frac{B_{lost}}{N_E}\right]$ and assuming $E[N_E]$ have uniform distribution. We have the following

$$E[B_U] = B_T\left(1 - \frac{p.N_P}{E[N_E]}\right) = B_T\left(1 - \frac{2.p.N_P}{1+N_P}\right) \tag{4}$$

To compare our model with BE and MPQ video transmission, we must measure the usefulness of the same amount of data in all models. Under EPIQ, $E[B_U]$ represents the useful data measured by the number of bytes from an EPIQ packet set that are used by the video decoder. On the other hand, in BE and MPQ, we measure the usefulness per a corresponding video content and compare the results with our system. The maximum theoretical achievable usefulness, which is the channel capacity, is:

$$E[B_U]_{Capacity} = N_P(1-p) \tag{5}$$

In [13], a model is developed for BE video packets with sequential dependency. The expected useful information for such model is:

$$E[B_U]_{BE} = \frac{1-p}{p}\left(1-(1-p)\right)^{N_P} \tag{6}$$

In MPQ, the system manages a number of queues where each queue has a different packet dropping precedence. Hence, N_P packets are enqueued into MPQ system according to packet importance. In an ideal scenario, only packets that belong to the lowest priority queue are dropped during congestion. In that case, we can consider Eq.(6) to represent the expected useful data of the last queue (least important) of the MPQ system while higher priority queues do not experience any packet dropping. Assuming that the MPQ system has m queues and each queue can accommodate k_i packets. Then, the useful data of the MPQ system $E[B_U]_{MPQ}$ can be expressed as:

$$E[B_U]_{MPQ} = \sum_{i=1}^{m-1} k_i + E[B_U]_{MPQ}^{m} \tag{7}$$

Where $E[B_U]_{MPQ}^{m}$ is the expected usefulness of the m_{th} queue. Eq. (7) can be rewritten as:

$$E[B_U]_{MQ} = (N_p - k_m) + \frac{1-p_m}{p_m}\left(1-(1-p_m)\right)^{k_m} \tag{8}$$

To further normalize the models, we define a utility function U:

$$U = \frac{E[B_U]}{E[B_U]_{Capacity}} \tag{9}$$

The utility function represents the ratio of the useful information received relative to channel capacity. Fig. 3 shows a comparison of the utility as a function of the number of packets in an EPIQ packet set (N_P) for the EPIQ, BE and MPQ models. For the case of MPQ, we show the case where MPQ system consists of three equal size queues. It is obvious that our system delivers higher level of useful data to the receiver when compared with BE and MPQ.

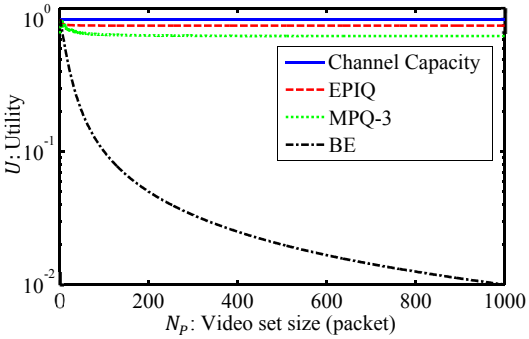

Fig 3. Utility as a function of video-set size ($p = 0.1$)

4. SYSTEM IMPLEMENTION

To implement the EPIQ system, we need a new Active Queue Management (AQM) solution within the network. In this section

we first discuss our proposed AQM algorithm. Also, to complement our AQM solution, we implement a rate control algorithm, which is discussed in the second part of this section.

4.1. Active Queue Management

Our AQM algorithm is built on the popular Random Early Detection (RED) algorithm [21]. In light of efforts that studied, analyzed, and improved RED [22], we designed our algorithm by taking these approaches into consideration; and we call our approach EPIQ Random Early Detection (EPIQ-RED). It is important to note that we use Byte version (Byte mode) and packet marking (Explicit Congestion Notification (ECN)) of the RED algorithm.

One key EPIQ-RED parameter is N, which is related to (but not the same as) the parameter N_E described in Section 3. Here, N represents the number of Partially Erasable Packets (PEPs) that are selected randomly from the queue for partial erasing. These N packets that are selected for partial erasing might belong to multiple EPIQ packet sets that belong to multiple users. However, only one of the N packets is marked as lost using ECN bit in the packet header. Assuming that we have different EPIQ packet sets with different sizes in the queue, under EPIQ-RED, we set N to some maximum value N_{max}. Here, N_{max} can be thought of as the maximum number of packets of an EPIQ packet set. However, in cases when N_{max} is larger than the current number of PEP packets in the queue, N is set to the current number of EPIQ packets in the queue q_{EQ}. Hence,

$$N = min(q_{EQ}, N_{max}) \tag{10}$$

Recall that PEP packets have a size of S bytes. Although it is possible to erase more than one packet size S of bytes at a time, we use the RED standard queue management procedure where one packet is dropped (or erased) at a time. This is also important to achieve fairness with different non-EPIQ flows. The amount of data that are erased from each of the N selected packets is represented by δ:

$$\delta = \frac{S}{N} \tag{11}$$

To accelerate the process of packet erasing, EPIQ-RED has to maintain the location and the number of EPIQ packets in the queue. A virtual list can be used to point to the location of the EPIQ packets in the queue. The minimum number of packets for RED threshold min_{th} is usually set to a number of bytes equivalent to five packets as recommended by [21]. However, since we are targeting to spread the erasing process over as many EPIQ packets as possible, a small queue will affect EPIQ transmitted video quality as shown in Eq.(4). Thus, we set the minimum threshold min_{th} as the maximum between the RED standard recommended threshold and the average value of the EPIQ packet set size $\overline{N}_P$. On the other hand, a large queue will cause long delays. As discussed in [9], limiting the number of packets in the queue reduces packet latency and prevents buffer-bloating. Hence, we set EPIQ-RED maximum threshold max_{th} as the minimum between: (a) The number of packets in the queue $N_Q(\tau)$ that is equivalent to a $\tau = 0.1$ second packet delay; and (b) Twice the maximum size N_{max} of an EPIQ packet set. The motivation for choosing twice the size of N_{max}, is because EPIQ video content will most likely occupy more than half of the queue. Hence, in terms of bytes, we employ the following expressions for min_{th} and max_{th}:

$$min_{th} = S \cdot max(5, 0.5\overline{N}_P). \tag{12}$$

$$max_{th} = S \cdot min(N_Q(\tau), 2 \cdot N_{max}) \tag{13}$$

The algorithm works by evaluating EPIQ-RED packet-dropping probability p_a for every incoming packet. If the packet that is marked to be dropped is a Partially Erased Packet (PEP), then the packet is enqueued and N PEP packets are randomly selected from the queue by using the virtual list and are partially erased; and only one packet is marked as lost. If it is not a PEP packet, the packet is dropped. Thus, we guarantee fairness between the different data types of traffic in the queue.

4.2. Rate and congestion control

The channel between the sender and the receiver should be monitored to ensure high-quality video delivery. In general, video transmission mechanisms can be categorized into: (a) Pull-Based streaming like HAS, which relies on the traditional Transmission Control Protocol (TCP); and (b) Push-Based streaming like RTP [1], which relies on multimedia oriented TCP friendly protocols like TFRC [23], GCC [24] and Kelly's framework [25]. Protocols like Datagram Congestion Control Protocol (DCCP) [26] and TCP-friendly Rate Control Protocol (TFRC) are well known RTP solutions. The congestion algorithm can be implemented as an application that uses UDP as a transport layer. Equation based control protocols use TCP response function [23] to estimate the sending rate. Since our proposed solution does not drop whole packets but partially erase them, using a multimedia oriented rate control protocol is easier to implement. Hence, we propose a modified TCP-Friendly Rate Control protocol (TFRC) and refer to it as EPIQ-TFRC. It is also important to note that we use Explicit Congestion Notification (ECN) mode of the TFRC algorithm and EPIQ-TFRC is built as an application layer protocol.

Another modification to standard TFRC that we included is a video streaming profile control algorithm. EPIQ-TFRC selects the appropriate video transmission rate (video quality profile) by calculating the running average of the sending bitrate using:

$$\hat{R}_{bps} = \begin{cases} \alpha\hat{R}(i-1) + (1-\alpha)R(i) & i > 1 \\ R\,(1) & i = 1 \end{cases} \tag{14}$$

$\hat{R}(i)$ represents the average sending rate at event i where, here, the event duration is 5 seconds. Also, reducing the quality profile of the video is allowed only if the playback buffer size is less than 5 seconds. Hence, based on the running average and the playback buffer size, the appropriate video quality profile is selected.

5. SIMULATION

In this section, we present the simulation results of our EPIQ system video transmission quality, and we compare the performance of EPIQ with DiffServ and DASH.

5.1. Simulation Setup

We used the ns-2 simulator to implement the EPIQ framework. Also, we employed the widely used bar-bell topology with multiple sources connected to a single bottleneck link. In our experiments, the capacity of the bottleneck link and queue size is set to 250 Mbps and 500 packets respectively while the rest of the links are fixed to 350 Mbps.

In our simulations, the traffic is generated by two equal member application groups where each has 10 sources (users) and a random traffic source. The first group is setup as FTP/TCP traffic generator while video streaming applications are setup as the second traffic group. Meanwhile, the random source generates ON/OFF traffic to simulate internet burst streams. For the video content, we used JSVM [27] and OpenSVCDecoder [28] to both generate trace files for our simulations, and to evaluate the received video quality. Eight High Definition (HD) video sequences[29] , with total video duration of 80 seconds, were

encoded as a single video streaming sequence in our simulation. The video was encoded as AVC stream at 30 FPS. Four versions of the video were generated to simulate different video quality profiles. The video supports the following quality profiles: 2, 3.5, 5.1, and 10 Mbps profiles. To evaluate the performance of our system, we compare it with commonly used video streaming methods. In the following subsection, we discuss the configuration related to each system.

5.1.1. EPIQ

Our proposed EPIQ video streaming system was configured according to Section 4. In our simulations, each AVC GoP represents one EPIQ video set (EVS), and it was carried by one EPIQ packet set (EPS). As explained in section 3.1, the highest priority data are placed near the EPS packet header (i.e. I frames data) followed by less priority data (i.e. P and B frames data). EPIQ-RED and EPIQ-TFRC were preconfigured to the ns-2 default parameters.

5.1.2. Multi-Priority Queue

Multi-Priority based video streaming is the prominent traditional solution to benefit from the priority structure of video. Our MPQ comparison model is based on DiffServ. At the sender, the AVC video stream was divided into three different priority streams depending on frame type (I, P and B). The AQM for DiffServ employs three queue sets. The first set is for multimedia traffic and it is divided into three virtual queues (Green, Yellow and Red) with each having different dropping precedence. The most important video traffic, I stream, is queued in the Green queue while P and B streams, the less important streams, are queued in the Yellow and Red queues respectively. The second queue set is for FTP; meanwhile, the third queue is for Burst traffic stream. Also, to control the video sending rate we used standard TCP Friendly Congestion Control (TFRC) protocol. The AVC quality profile was selected in a similar manner to the EPIQ system. The system used the default ns-2 TFRC and DiffServ parameters.

5.1.3. HTTP Streaming

Nowadays, one of the most widely used video streaming algorithm is Dynamic Adaptive Streaming over HTTP (DASH). DASH video traffic is streamed over best effort Internet. Hence, in our simulations, Drop Tail was used as the AQM algorithm in the bottleneck network. At the sender, The H.264 AVC video Quality profiles were used as DASH supported bitrates similar to EPIQ and DiffServ systems. In standard DASH, data are transported using TCP, which provides bandwidth estimate to the application layer to adjust the sending rate accordingly. We used the DASH standard method to control the selected video bitrate.

5.2. Simulation Results

5.2.1. AQM Stability and Congestion Control Fairness

Our EPIQ-RED algorithm shares the core principles of RED. According to Eq. (12)-(13), min_{th} and max_{th} are set automatically to 79 and 412 packets, respectively, while the average queue size is 120 packets. We found that our algorithm is relatively stable in comparison to what is found in [22] and consistent with RED performance. However, due to space limitation we did not include our detailed stability plot in this paper.

Congestion control is necessary to achieve smooth video streaming while maintaining fairness with other flows in the network. For a completed 80 seconds simulation, the average share ratios are, 1.02, 0.98, and 0.94 for EPIQ, DiffServ and DASH respectively. It is clear that DiffServ-TFRC and EPIQ-TFRC connections provide a higher level of fairness toward FTP/TCP than DASH. It is important to note that despite the fact that we introduce changes to the original TFRC work, we find our system operates within original TFRC boundaries.

5.2.2. Video Quality

A key goal of our work is to provide high quality video transmission. The performance metrics used to evaluate the quality of the received video traffic include the average Peak Signal to Noise Ratio (PSNR) of the luminance (Y) component, packet retransmission, delay parameters and video quality profile changes. In our simulations, retransmissions were allowed only for I frame packets of the AVC video for both EPIQ and DiffServ. Meanwhile, the retransmission function was allowed for all DASH packets, which include both control packets and video streaming packets because of the necessity of all packets to DASH operation.

Table (1) Transmission and video quality

	Delay (msec)	**Delay Jitter (msec)**	**Quality Change**	**Packet-Retransmit (%)**	**PSNR (dB)**
EPIQ	28	5	1.8	0	38.5
DiffServ	20	7	2	0.2	38.4
DASH	635	350	3.3	3.9	35.7

We tested our proposed solution and computed the average results for the performance metrics that we stated earlier. All results represent the average per user over 80 seconds, where this average is evaluated over 10 video sessions corresponding to the 10 users in our simulations.

Important performance parameters are application to application data delay parameters (average delay and delay jitter). Table (1) shows the average delay parameters of the three mechanisms as seen by user application layer. EPIQ delay parameters are slightly higher than DiffServ because the EPIQ receiver forwards the received data to the application after receiving all EPS's packets. DASH on the other hand, has very high delay parameters because it relies on TCP to deliver data. Both the packet retransmission process and the TCP buffer contribute to the delay. Regarding retransmission, EPIQ did not retransmit any packet while DiffServ retransmitted 0.2% of the total number of packets transmitted. Because DASH uses HTTP/TCP, it retransmitted 3.9% of the total packets.

Also shown in Table (1), the performance of the different video transmission mechanisms from the user prospective with startup buffering of 5 seconds. It is crucial to highlight that EPIQ achieves 2.8 dB in PSNR improvements over DASH. Further, EPIQ outperforms DiffServ by 0.1 dB. Regarding video quality-profile changes (which correspond to bitrate changes), DASH suffers from almost two times the quality changes when compared to EPIQ while DiffServ has an average of 2 profile changes. Note that neither EPIQ, DiffServ nor DASH suffer from any stalling. It should be clear from these results the importance of quality/bitrate adaptation, Such bitrate changes are necessary to avoid video stalling. (During the stalling time, the video playback stops for buffering.)

6. CONCLUSION

In this paper, we developed a novel approach, Erasable Packets within Internet Queues (EPIQ), for video streaming. EPIQ provides high quality video transmission with minimal network complexity. Under congestion, EPIQ works by erasing portions of multiple video packets within a network queue rather than the conventional packet dropping mechanism. We developed an

analytical model for our proposed solution and verified it using simulation. We further introduced EPIQ-RED AQM, based on the well know RED algorithm, to implement our partial packet erasing. Furthermore, to maintain fairness and to control the rate of competing video streams, we presented EPIQ-TFRC. Our simulation results show that our system can achieve better performance than both MPQ and HTTP-based video streaming. In summary, the proposed EPIQ solution introduces a new paradigm in video streaming by partially erasing packets to control congestion in the network while maintaining the highest video transmission quality.

6. REFERENCES

[1] A. C. Begen, T. Akgul, and M. Baugher, "Watching Video over the Web: Part 1: Streaming Protocols," *IEEE Internet Computing*, vol. 15, no. 2, pp. 54–63, Mar. 2011.

[2] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. 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*, 2012, pp. 9–14.

[3] S. Akhshabi, S. Narayanaswamy, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptive video players over HTTP," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 271–287, Apr. 2012.

[4] R. Kuschnig, I. Kofler, and H. Hellwagner, "An Evaluation of TCP-based Rate-control Algorithms for Adaptive Internet Streaming of H.264/SVC," in *Proceedings of ACM SIGMMs*, New York, NY, USA, 2010, pp. 157–168.

[5] J. Jiang, V. Sekar, and H. Zhang, "Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming With Festive," *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 326–340, Feb. 2014.

[6] M. Nagy, V. Singh, J. Ott, and L. Eggert, "Congestion Control Using FEC for Conversational Multimedia Communication," in *Proceedings of the 5th ACM Multimedia Systems Conference*, NY, USA, 2014, pp. 191–202.

[7] D. D. Clark and W. Fang, "Explicit Allocation of Best-effort Packet Delivery Service," *IEEE/ACM Trans. Netw.*, vol. 6, no. 4, pp. 362–373, Aug. 1998.

[8] R. Adams, "Active Queue Management: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1425–1476, 2013.

[9] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet," *Commun. ACM*, vol. 55, no. 1, pp. 57–65, Jan. 2012.

[10] T. Pliakas, G. Kormentzas, and C. Skianis, "End-to-end QoS issues of MPEG-4 FGS video streaming traffic delivery in an IP/DVB/UMTS network," *European Journal of Operational Research*, vol. 191, no. 3, pp. 1089–1100, Dec. 2008.

[11] V. Firoiu and X. Zhang, "Best Effort Differentiated Services: Trade-off Service Differentiation for Elastic Applications," in *in Proc. IEEE ICT 2001*, 2000.

[12] Y. Xiaogang, L. JiQiang, and L. Ning, " Congestion Control Based on Priority Drop for H.264/SVC," in *International Conference on Multimedia and Ubiquitous Engineering, 2007. MUE '07*, 2007, pp. 585–589.

[13] S.-R. Kang, Y. Zhang, M. Dai, and D. Loguinov, "Multi-layer active queue management and congestion control for scalable video streaming," in *24th International Conference on Distributed Computing Systems, 2004. Proceedings*, 2004, pp. 768–777.

[14] MPEG-DASH. ISO/IEC 23009, Retrieved November 16, 2015 from http://mpeg.chiariglione.org/standards/mpeg-dash

[15] R. Huysegems et al, "HTTP/2-Based Methods to Improve the Live Experience of Adaptive Streaming," in *Proceedings of the 23rd ACM International Conference on Multimedia*, NY, USA, 2015, pp. 541–550.

[16] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719–733, Apr. 2014.

[17] G. Tian and Y. Liu, "Towards Agile and Smooth Video Adaptation in Dynamic HTTP Streaming," in *Proceedings of the 8th International Conference on Emerging Netw. Exper. and Tech.*, New York, NY, USA, 2012, pp. 109–120.

[18] D. Jarnikov and T. Ozcelebi, "Client intelligence for adaptive streaming solutions," in *2010 IEEE ICME*, 2010,

[19] Y. Sánchez de la Fuente et al, "iDASH: improved dynamic adaptive streaming over HTTP using scalable video coding," in *Proceedings of the second annual ACM conference on Multimedia systems*, 2011, pp. 257–264.

[20] R. Kuschnig, I. Kofler, and H. Hellwagner, "An Evaluation of TCP-based Rate-control Algorithms for Adaptive Internet Streaming of H.264/SVC," in *Proceedings of the First Annual ACM SIGMM Conference on Multimedia Systems*, New York, NY, USA, 2010, pp. 157–168.

[21] V. Jacobson, K. Nichols, K. Poduri, and C. Systems, "RED in a Different Light,", *Technical report,* 1999.

[22] C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong, "A control theoretic analysis of RED," in *IEEE INFOCOM ,* 2001, vol. 3, pp. 1510–1519 vol.3.

[23] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based Congestion Control for Unicast Applications," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, New York, NY, USA, 2000, pp. 43–56.

[24] L. De Cicco, G. Carlucci, and S. Mascolo, "Experimental investigation of the google congestion control for real-time flows," in *Proceedings of the 2013 ACM SIGCOMM*, 2013, pp. 21–26.

[25] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability," *The Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, 1998.

[26] S. Floyd, M. Handley, and E. Kohler, "Datagram Congestion Control Protocol (DCCP)." March 2006, RFC 4340 .

[27] Joint Video Team, "Reference software," http://iphome.hhi.de/suehring/tml/

[28] M. Blestel and M. Raulet, "Open SVC decoder: a flexible SVC library," in *Proceedings of the international conference on Multimedia*, 2010, pp. 1463–1466.

[29] Derf's short video collection." http://media.xiph.org".