EPIQ VIDEO STREAMING: PACKET DESIGN AND PERFORMANCE ANALYSIS

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 video-streaming framework that is based on a novel packet design in conjunction with low-complexity active queue management within the network. Our proposed framework, which we call Erasable Packets within Internet Queues (EPIQ), exploits the inherent multi-priority nature of video to deliver optimal quality-of-experience to users, yet without requiring any additional overhead in terms of redundant packets or network resources such as multiple queues or maintaining state information about flows. The novelty of EPIQ is rooted at its packetization. Each EPIQ packet consists of multiple segments and where each segment carries different priority video content. Under congestion, a network node (ideally a 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 improvements over multi-queue systems. These improvements are achieved while maintaining low-complexity within the network.

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

1. INTRODUCTION

Current widely-deployed Internet video applications and services employ well-established protocols such as TCP and HTTP. In particular, HTTP-based Adaptive Streaming (HAS) [1], uses 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].

In addition to severe congestion events, traditional streaming solutions lead to oscillatory behavior that is highly undesirable in terms of users' Quality of Experience (QoE). 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 video streaming framework that 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 Active Queue Management (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; the model takes into consideration an abstract structure for dependencies among video packets; (b) conduct an extensive set of simulations using ns-2 to compare the performance of EPIQ with a multi-queue system. Our analysis and simulation results validate EPIQ's capability in delivering competing performance when compared with a multi-priority system in PSNR metric while maintaining low network-complexity and without the burden that DiffServ and similar multi-priority queue solutions require from the network.

2. BACKGROUND AND RELATEDWORK

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

This work was supported in part by an unrestricted gift from Google, Inc. under a Google Faculty Research Award.

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. It is worth noting that beside QoS there are other priority-based mechanisms, most notably ones that require the transmission of end-to-end redundant data in the form of Forward Error Correction (FEC) packets, such as Priority Encoding Transmission (PET) and Unequal Error Protection (UEP) mechanisms. Unlike PET and other FEC-based solutions, EPIQ does not require any redundant packets, and hence it is significantly more efficient in terms of network bandwidth utilization.

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. However, all of the proposed solutions that use multi queues to support video priority dropping have a limited number of queues. If the number of the video priority levels is more than the number of queues then the usefulness (or utility) of the packets delivered to the video decoder cannot be guaranteed.

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.

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.

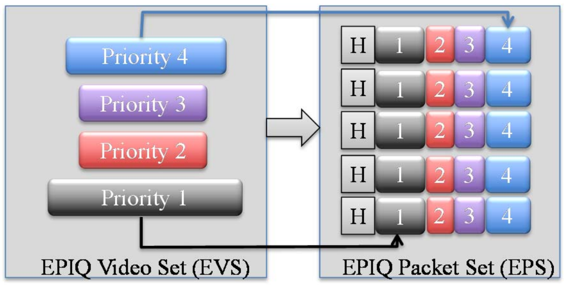

Fig. 1. The mapping of multi-priority video content from an EPIQ Video Set (left) to Partially Erasable Packets of an EPIQ Packet Set (right).

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 multi-priority queue 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 develop an analytical model for the performance of the proposed system. This analytical model is intended to provide an insight into how our system performs in comparison with traditional best-effort streaming. More importantly, we further compare our system with the ideal case of achieving channel capacity.

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 EPIQ Packet Set (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 we have the following:

$$B_T = N_P \cdot S \tag{1}$$

Note that Eq. (1) is valid for both our proposed solution and a conventional system. Let B_{lost} be the number of bytes that would be lost in a conventional system due to packet dropping. The expected value of the number of bytes to be lost within an EPIQ packet set can be written as:

$$E[B_{lost}] = p \cdot B_T \qquad (2)$$

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. In general, the selected packets in the PE process are not necessarily contiguous and may encounter a different level of erasing. Let (N_E) be the number of packets (from a given EPIQ packet set) that encounter partial erasing. When congestion occurs at the network, then at the receiver, we may observe anywhere from a single packet to a maximum number of N_P packets that have suffered from partial erasing for a given EPIQ packet set. Hence, under congestion, N_E is a uniform random variable with a probability mass function $\frac{1}{N_P}$

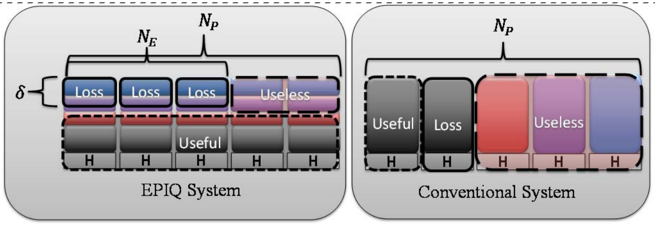

Fig. 2. EPIQ vs. conventional

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:

$$\bar{\delta} = E\left[\frac{B_{lost}}{N_E}\right] \qquad (3)$$

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:

$$E[B_U] = B_T - \bar{\delta} \cdot N_P = B_T - E\left[\frac{B_{lost}}{N_E}\right] \cdot N_P \qquad (4)$$

$$E\left[\frac{B_{lost}}{N_E}\right] \cong \frac{E[B_{lost}]}{E[N_E]} - \frac{cov(B_{lost}, N_E)}{E^2[N_E]} + \frac{var(N_E)E[B_{lost}]}{E^3[N_E]} \qquad (5)$$

Here, we assume that: (a) the variation in N_E value is rather small relative to its mean; and (b) the correlation between B_{lost} and N_E is also small relative to the expected value of N_E. Our simulation results confirm that these assumptions are quite reasonable. This leads to the first order approximation of Eq. (5).

$$E\left[\frac{B_{lost}}{N_E}\right] \cong \frac{E[B_{lost}]}{E[N_E]} \qquad (6)$$

Hence,

$$E[B_U] = B_T - \frac{E[B_{lost}]}{E[N_E]} \cdot N_P \qquad (7)$$

Or,

$$E[B_U] = B_T\left(1 - \frac{p.N_P}{E[N_E]}\right) = B_T\left(1 - \frac{2.p.N_P}{1+N_P}\right) \qquad (8)$$

To compare our model with Best Effort (BE) video transmission, we must measure the usefulness of the same amount of data in both 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, 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:

$$E[B_U]_{Capacity} = N_P(1-p) \qquad (9)$$

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

$$E[B_U]_{BE} = \frac{1-p}{p}\left(1-(1-p)\right)^{N_P} \qquad (10)$$

To further normalize our model, we define a utility function U:

$$U = \frac{E[B_U]}{E[B_U]_{Capacity}} \qquad (11)$$

The utility function represents the ratio of the useful information received relative to channel capacity. For our EPIQ model, the utility function is

$$U_{EPIQ} = \frac{\left(1 - \frac{2.p.N_P}{1+N_P}\right)}{1-p} \qquad (12)$$

The utility function for the Best Effort model becomes:

$$U_{BE} = \frac{1-(1-p)^{N_P}}{N_P.p} \qquad (13)$$

Fig. 3 shows a comparison of the utility as a function of the number of packets in an EPIQ packet set (N_P) for both the EPIQ and BE models. It is obvious that our system delivers significantly higher level of useful data to the receiver when compared with BE. For example, EPIQ at $N_P = 20$ achieves 90% utility while BE only achieves 40% utility. Another example is when $N_P = 100$, EPIQ achieves 89% while BE could only maintain 10%. Our extensive simulations will further validate the results of this model.

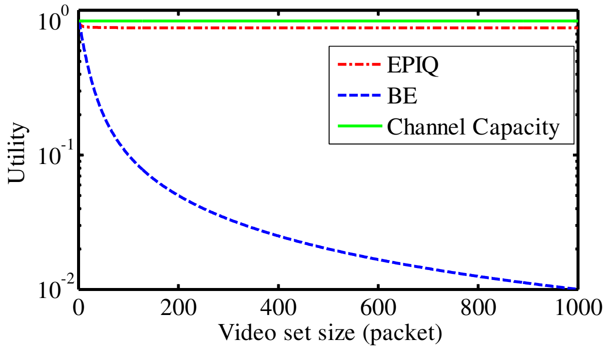

Fig 3. Utility as a function of video-set size ($p = 0.1$)

4. SIMULATION

In this section, we present the simulation results of our EPIQ system video transmission quality, and we compare the performance of EPIQ with multi-priority queue system.

4.1. Simulation Setup

We used the ns-2 simulator to implement the EPIQ framework. The simulation topology consisted of a sender, receiver and middle box that represents the congested network. The video content was generated and evaluated using JVT reference software [14], SAEF framework [15] and Open SVC Decoder [16]. We simulate both AVC and SVC [17] video stream of Park Run video sequence encoded at 50 FPS and GoP of size 8. The SVC stream consisted of one base layer and two enhancement layers and GOP of size 8. Both streams have Peak Signal to Noise Ratio (PSNR) quality of 36.6 dB. We evaluated the performance of our proposed method and multi-priority systems at packet loss probability p of 1%, 5%, 10%, 20% and 30%.

4.1.1. EPIQ

To configure EPIQ, we need to determine the content of our EPS and the number of packets to be erased during congestion. For the AVC video stream, we specified that each EPIQ video set (EPS) to carry the content of one GoP, which consists, in our simulations, of 8 frames. In the SVC case, we have chosen two different configurations; in the first configuration; the EPS carried the content of one SVC frame (3 layers), while in the second configuration, the EPS carried the content of one SVC GoP frames. As stated earlier in the analysis section, any number of EPS packets can be selected during congestion.

4.1.2. Multi-priority

In the multi-priority system case, one queue is assigned for each priority in the congested node so that during congestion the lowest priority queue drops its packets. For AVC stream, we decided to use the most common number of priority queues which is three based on frame types (I, P and B) classification. In a second scenario, we decided to use the maximum number of possible priorities that the video supports to capture frame's dependency. In such ideal case, the number of priorities is the GoP size, which is eight in our configuration. For SVC video, first we decided to use three priority queues and assign the highest priority to the base layer while the 2nd and 3rd (lowest priority) are assigned to the first and second enhancement layers respectively. In another SVC scenario, similar to AVC 2nd scenario, we set the number of priorities to the maximum number of priorities that can be supported by the SVC stream. In such case, the number of priorities is set to the GoP size multiplied by the number of video layers (i.e., the total number of priorities is 24 in the ideal SVC case).

4.2. Simulation Results

Fig. 4 shows the average PSNR of the received video of both EPIQ and the multi-priority solution for the AVC stream case. Our proposed method (EPIQ-GoP), on average, is both 6dB better than the system supporting three-priority queues (Pri-3) within the network, and only 0.8 dB less than the significantly more complex (ideal) eight-priority system (Pri-8). The reason behind the poor performance of the three-priority system is that although it gives highest priority to I and P frames, it assigns the same lowest priority to all B frames without taking into consideration the dependability among B frames. Meanwhile, the eight-priority system takes into consideration all frame dependencies with added overhead of managing eight queues rather than three. On the other hand, our system employs a single queue without complex management or significant overhead, and its performance is independent of the number of priority levels that are carried by the video packets.

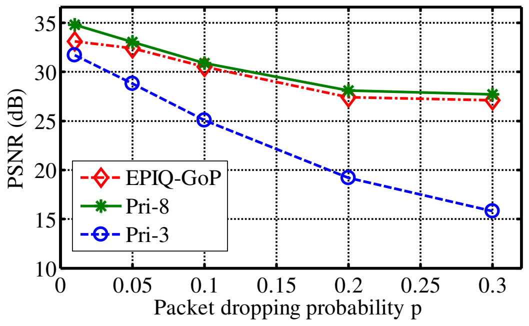

Fig. 4. PSNR of EPIQ and Multi-priority for AVC Stream

Fig. 5 shows the results of SVC video streaming for both EPIQ and the multi-priority system. For the three-priority queues (Pri-3), it is noticeable that the PSNR values do not change even when the number of dropped packets increases. The reason for such behavior is that the decoder is only decoding the first enhancement layer. Similar to the AVC case, not all of layer two (enhancement layer) packets have the same priority. Our proposed system performance with EPS carrying only one frame content (EPIQ-Frame) is 3 dB better than the three-priority systems (Pri-3). It is important to note that EPIQ with EPS carrying GoP (EPIQ-GoP) achieves, on average, virtually the same performance of the ideal high-complexity 24 -priority system (Pri-24); the difference in performance is only 0.3dB.

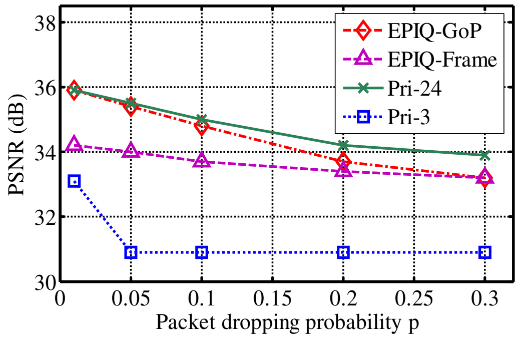

Fig. 5. PSNR of EPIQ and Multi-priority for SVC stream

5. 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 of our proposed solution and verified it using simulation. Our simulation results show that our system can achieve significantly better performance than three priorities multi-queue system and achieves a comparable performance with high order multi-queue system. 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 the First Annual ACM SIGMM Conference on Multimedia Systems*, 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*, New York, 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] Joint Video Team, "Reference software," http://iphome.hhi.de/suehring/tml/

[15] A. Detti, G. Bianchi, C. Pisa, F. S. Proto, P. Loreti, W. Kellerer, S. Thakolsri, and J. Widmer, "SVEF: an open-source experimental evaluation framework for H.264 scalable video streaming," in *IEEE Symposium on Computers and Communications, 2009. ISCC 2009*, 2009, pp. 36–41.

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

[17] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.