# Hard-Deadline-based Frame Filtering Mechanism Supporting the Delivery of Real-Time Video Streams

Jun Liu (jliu@cs.und.edu)
Computer Science Department, University of North Dakota

***Abstract*— This paper describes a cross-layer filtering mechanism which facilitates real-time video frames to meet their stringent decoding deadlines in the existence of network congestion. The basic idea is to remove the dysfunctional video frames, which have missed their decoding deadlines, from transmission as early as possible, since they no longer serve for the functioning of a real-time media streaming application. The filtering mechanism consists of a pair of components which operate at the encoder and the decoder, respectively. The decoder-side component identifies the dysfunctional frames and sends the notifications to the encoder. The encoder-side component removes the identified dysfunctional frames from transmission. By removing dysfunctional frames from transmission, the video frames that are behind the dysfunctional frames are eligible for transmission at an earlier time and are made likely to meet their decoding deadlines. Meanwhile, removing dysfunctional frames from transmission also serves to maintain a stable and low queueing delay. The filtering mechanism relies on a user-space transport stack which enables the application-controlled transmission of data segments. The effectiveness of the filtering mechanism has been demonstrated through experiments in emulated networks.**

***Index Terms*— Real-Time Media Streaming, Deadlines, Frame Cancellation, Cross-Layer Design, User-Space Transport Support**

## I. Introduction

Real-time video streaming applications have stringent requirements of meeting the latency constraints, such as the live or interactive video streaming applications. In these applications, video frames that are delivered from an encoder to a decoder are only valid within a tight time frame. If a video frame can not be decoded before the end of its valid time frame, then it can not contribute to the functioning of an application. Hence, maintaining the perceived quality requires that a steady transmission delay is maintained along the end-to-end path used for video stream delivery. However, the bit rate of a video stream can largely vary over short and long timescales due to the inherently varying spatio-temporal complexity of video content [1]. The unsteady bit rates of video streams easily lead to fluctuations of end-to-end delays and packet losses, which, in turn, cause many video frames to miss their decoding deadlines on the decoder side. Effective mechanisms are needed for supporting real-time video streaming to achieve an acceptable rendering quality.

Many rate-adaptive video encoding methods [2], [3], [4], [5], [6] have been proposed for regulating the unsteady bit rates of video streams in the existence of dynamic bandwidth, delays, and packet losses. These methods can be roughly categorized into multi-file/multi-rate switching methods [4], [5], the adaptive single-layer encoding methods [7], [8], and the scalable multi-layer encoding methods [9], [3], [10]. These methods require either special encoding mechanisms, or redundant storage of the video content, or special encoding on video content, or special filtering mechanism.

In this paper, we describe a cross-layer filtering mechanism which fits into the category of adaptive single-layer encoding methods. This filtering mechanism aims to facilitate real-time video frames to meet their stringent decoding deadlines by removing *defunct frames* from transmission. A defunct frame is a frame which either has missed its decoding deadline or can not be properly decoded because of a corrupted dependency. The filtering mechanism operates on top of a user-space transport stack which runs completely in user space. The user-space transport stack enables the application-controlled transmission of data segments. The filtering mechanism consists of a pair of components which operate at the encoder and the decoder, respectively. The decoder-side component identifies the defunct frames and sends the notifications to the encoder. The encoder-side component removes the identified defunct frames from transmission through interacting with the user-space transport stack. By removing the defunct frames from transmission, the frames that sit behind the defunct frames are made likely to meet their decoding deadlines, since they are eligible for transmission at an earlier time.

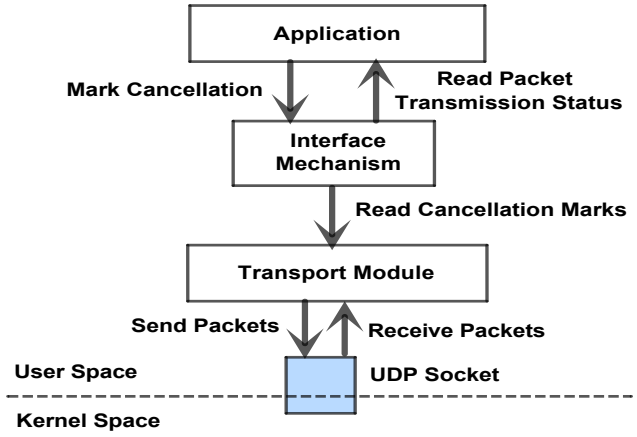



Fig. 1. The architecture of the interactions between an application and the user-space transport stack.

The cross-layer filtering mechanism relies on the support of an interface mechanism which facilitates the interaction between an application and a user-space transport stack as illustrated in Figure 1. The user-space transport stack provides a flexible user-space transport support and enables application-controlled flow control and feedback by allowing application-specific customizations [11]. A real-time video streaming application can express its control instructions over the data segments to the user-space transport stack through an interface. The interaction can be viewed as a user-space implementation of the kernel-space transport-layer buffer holding the unacknowledged data segments, *e.g.*, the congestion window used in TCP. The interface is only needed on the sender side. Through the interface mechanism, an application can also read the transmission status of the data segments that are under transmission. Enabling the interaction between a real-time video streaming application and the user-space transport stack is essential to make the rate of the video stream to be adapted to the available bandwidth [12], [13], [14].

Our contributions in this work are briefed as follows. First, an application-controlled flexible transmission arrangement is made possible through the interaction between a real-time video streaming application and a user-space transport stack. Second, our filtering mechanism does not rely on prioritized scheduler or multiple queues on the encoder side. It is highly computationally expensive in operating prioritized scheduler or maintaining multiple queues. Only simple data structures are used in our mechanism for identifying and removing the defunct frames. Third, no special and sophisticated encoding scheme is required in our case. Our mechanism only needs to know the inherent dependency structure among video frames. Fourth, the defunct frames are more effectively identified in our filtering mechanism, because the identification is made by the decoder which has a better knowledge than the encoder about which frames no longer serve for the functioning of a real-time video streaming application. In many other designs, the scheduling decisions on transmission are made on the encoder side. In general, the encoder-side scheduling decisions on transmission are less effective because the encoder has to infer the need of the decoder.

The effectiveness of our filtering mechanism has been demonstrated through experiments in emulated networks. The experimental results have demonstrated the following benefits of adopting our filtering mechanism. First, removing the defunct frames from transmission serves to stabilize the end-to-end queueing delays. Second, the prioritized video frames (I frames) are relatively favored for continued transmission than the less prioritized frames (P or B frames), when the network is in congestion. Third, removing defunct frames from transmission based on the identification made by the decoder allows the play-out quality to be gracefully downgraded when the network is in congestion. Blindly removing frames from transmission will inevitably worsen the play-out quality.

In the rest of this paper, the related work is presented in Section II. The structure of the interface mechanism is briefed in Section III. The structure of our cross-layer filtering mechanism is described in Section IV. The empirical evaluation of the filtering mechanism is described in Section V. Our work is summarized in Section VI.

## II. Related Work

Rate adaptation can be achieved through joint source/channel encodings which combine the transport control and the video encoding mechanisms. The video encoding can be altered in order to adapt the bitstream rate to the available bandwidth [15]. A transport control mechanism reacts to network congestion by adjusting the rate at which a bit stream is injected into the network [12], [13], [14].

The rate-adaptive video encoding methods can be roughly categorized into the multi-file/multi-rate switching methods [4], [5], the adaptive single-layer encoding methods [7], and the scalable multi-layer encoding methods [9], [3], [6], [10].

Our filtering mechanism belongs to the adaptive single-layer encoding methods. Although a video stream can be encoded into multiple layers, our filtering mechanism treats a layer-encoded video stream as a single layer. Rate adaptation is achieved through dropping the defunct frames, many of which belong to the enrichment layer and are with a lower priority, in the existence of network congestion.

### A. Semi-Reliable Transport Control Protocols

Adaptive Automatic Repeat reQuest (ARQ) [16] with conditional frame skipping, reference frame selecting and intra-frame refreshing techniques has been adopted for H.26L real-time video streaming over WLAN. Semi-reliable multicast [17] is designed to support the distributed multimedia applications based on groups. In this scheme, data are classified before being transmitted, establishing different importance or priority levels for error recovery (retransmission). A transcoding-based frame-rate control scheme is proposed in [18] for frame-rate reduction to improve picture quality. This scheme dynamically adjusts the number of skipped frames according to the incoming motion vectors and re-encoding errors due to transcoding such that the decoded sequence can have a smooth motion as well as better transcoded pictures. Selective frame discard [8] is a transport-level rate shaping, which estimates the minimum number of frames that must be discarded in order to meet transport constraints.

## III. The Structure of the Interface Mechanism

The interface mechanism enables the interaction between an application and the user-space transport stack. On one hand, an application can pass its transmission control instructions to the transport stack. On the other hand, the transport stack can report the status of the transmission of data segments to an application. The main component of the interface is a circular queue which maintains a transport-oriented record for each data segment, as well as the cancellation flag. An interface mechanism is maintained for each stream to hold the not-yet-acknowledged data segments of a stream. The user-space transport stack can support multiple streams and can access the interface mechanisms of all the active streams.

The user-space transport stack enables applications to perform application-controlled semi-reliable transmission of data segments. An application can remove some data segments from continued transmission after these segments have been submitted to the transport stack for transmission. An application expresses its cancellation instructions through the interface to the transport stack which physically removes the data segments from its storage. The user-space transport stack is developed based on DCCP/TP [19] which is a user-space implementation of the DCCP protocol [20], [21]. In our work, we have introduced the cancellation mechanism into the DCCP/TP implementation. The cancellation flag of each data segment is assigned with a value of "NOT CANCELLED" by default when it is submitted to the transport stack by an application. Through the interface mechanism, an application can change the cancellation flag of a data segment from "NOT CANCELLED" to "CANCELLED." Hence, the data segments which have not been cancelled by an application are allowed for reliable transmissions. In contrast to our user-space transport stack, the traditional transport stack does not allow an application to cancel the transmission of data segments that are under its transport control.

## IV. The Cross-Layer Filtering Mechanism

The cross-layer filtering mechanism aims to facilitate real-time video frames to meet their stringent decoding deadlines by removing the defunct frames from transmission. The filtering mechanism is composed of two components which are deployed at the encoder and at the decoder, respectively. The decoder-side component identifies the defunct frames using a dependency structure among frames, and it also notifies the encoder about the identified defunct frames through embedding the notifications in the acknowledgment packets that are sent to the encoder. Then, the encoder-side component removes the identified defunct frames from transmission.

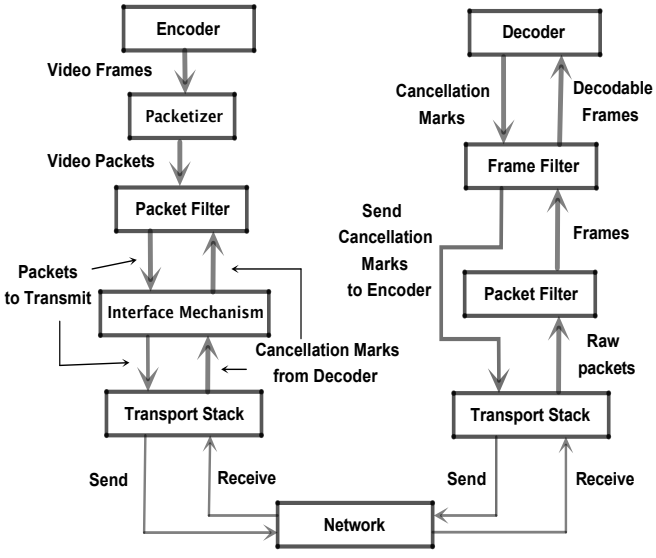



Fig. 2. The architecture of the filtering mechanism.

### A. The General Structure of A Filtering Component

The architecture of the filtering mechanism is shown in Figure 2. The filtering mechanism is composed of two filtering components that are at the encoder and at the decoder, respectively. The decoder-side component identifies the defunct frames using the decoding deadlines and the dependency structures among video frames. The decoder-side component also notifies the encoder-side component about the identified defunct frames. The encoder-side component updates its filtering rules using the notifications sent by the decoder-side component. The filtering components in our work are constructed to support MPEG encoded video streams. The methodology of the filter construction can be applied to other encoding methods.

### B. The Filtering Component on the Encoder Side

The encoder-side filtering component is illustrated in Figure 3 (a). A list of reference frames is maintained in the packet filter for keeping track of the dependency structures among frames. Once any of the reference frames in the list is notified by the decoder as a defunct frame, the encoder marks the record of the same frame in the packet filter as *defunct*. The future packetized frames generated by the packetizer are ignored if any one of their reference frames has been marked as *defunct* in the list of reference frames. Meanwhile, the encoder also changes the cancellation flags of these packetized frames in the transport stack into "CANCELLED."

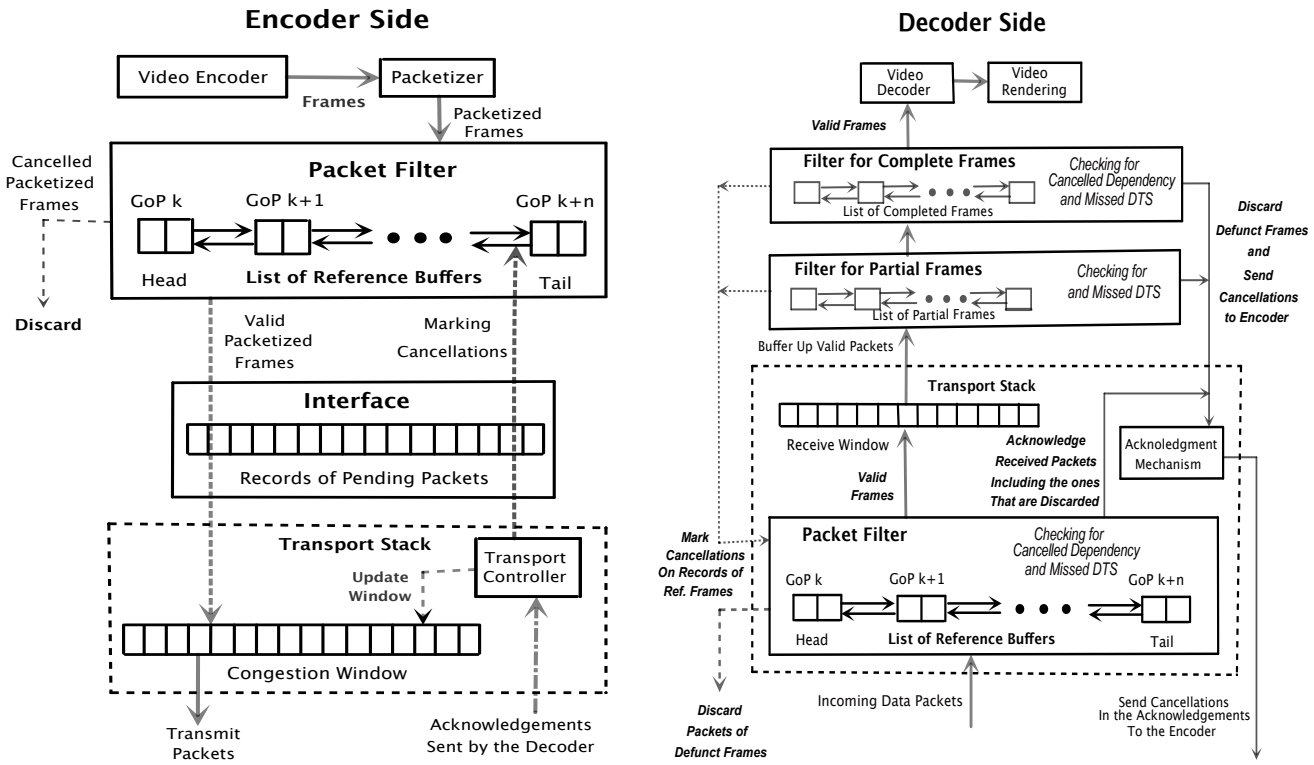
### C. The Filtering Component on the Decoder Side

The decoder-side filtering component is illustrated in Figure 3 (b), which checks for the expiration of the decoding deadlines for either the partial or completed frames. This filtering component accepts the valid frames for decoding. A valid frame is one which is received before the frame's decoding deadline and satisfies its decoding dependency. The decoding deadline for a MPEG frame is specified by a Decode Time-Stamp (DTS). All deadline-expired frames are discarded. If a deadline-expired frame is a reference frame, then it has to be labeled as *defunct* in the list of reference frames. The decoder also notifies the encoder with the identified defunct frames by embedding the notification in an ACK packet.

## V. Empirical Performance Evaluation

The effectiveness of the cross-layer filtering mechanism is demonstrated through experiments that are conducted in a lab-scale emulated network with the network topology shown in Figure 4. These experiments aim to demonstrate that the following effects can be achieved by adopting the cross-layer filtering mechanism.

(1) Maintaining a stable and low end-to-end queueing delay;
(2) Favoring the more important video frames (I frames) for continued transmissions when the network is in congestion;
(3) Gracefully downgrading the play-out quality when the network become congested.




(a) On the Encoder Side (b) On the Decoder Side

Fig. 3. The structures of the encoder-side and the decoder-side components.

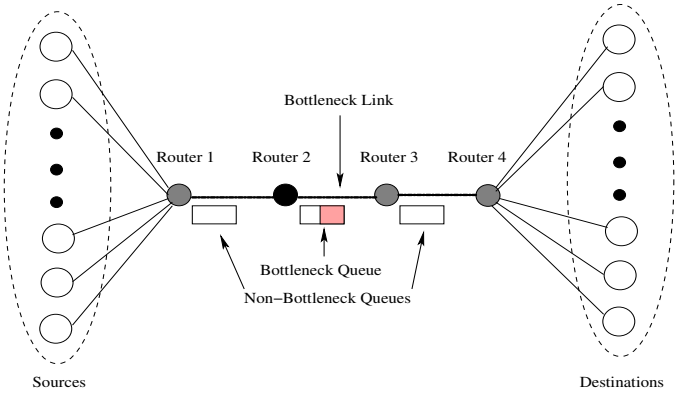



Fig. 4. The Topology of the Sample Network Used in the Experiments.

### A. Experiment Settings

In a lab-scale emulated network with its topology shown in Figure 4, the backbone path is made up of four emulated routers. The link between router 2 and 3 is made into a bottleneck link, and router 2 is treated as the bottleneck router. The link capacity, buffer capacity, propagation delays, and packet dropping policies of router 2 are configurable by running the NISTnet [22] emulation toolbox. The NISTnet emulator emulates the handling by a "real router" on the traffic flow from router 2 to router 3. The NISTnet emulator can drop packets according a dropping policy, or can hold up packets for a prescribed time interval for emulating a propagation delay. The NISTnet emulator forwards packets onto the bottleneck link at the speed of the prescribed link bandwidth. Since the emulated bandwidth of the bottleneck link is configured as 4 Mbps as compared to 100 Mbps bandwidth on other links, almost all the queueing delays happen at the outgoing queue of router 2. The one-way propagation latency from a source to a destination is set to be 50ms.

In the experiments, 15 flows deliver the same video stream from sources (video encoders) to their corresponding destinations (video decoders) as shown in Figure 4. The video stream is called `xmen2` which contains a total of 3448 frames and lasts approximately 140 seconds. The video signal is encoded in MPEG-4 format with a video encoding rate of 250Kb/s (or 25 frames/second) and an audio encoding rate of 96Kb/s. There are 297 GoPs with an average of 11.6 frames per GoP in the video clip. Among the 15 flows, 5 flows are treated as the real-time video streams, and the rest are treated as the cross traffic. By enabling or disabling the 10 cross traffic flows, we demonstrate the performance of the 5 real-time video streams with or without cross traffic along the backbone path. Each video frame is associated with a fixed 55ms decoding deadline which is slightly larger than the one-way propagation delay. This means that each video frame can tolerate minor queueing delays. The same value of the deadline applies to all the evaluation results shown in this section.

The play-out quality is evaluated using the metric of the peak signal-to-noise ratio (PSNR). The computation of the PSNR metric is based on the `EvalVid` package [23] and the *ViTooKi* package [24]. The `EvalVid` package measures the video quality evaluation of the received video based on the

frame-by-frame PSNR calculation. The `ViTooKi` package is a high-level multimedia library which supports a large number of multimedia features.

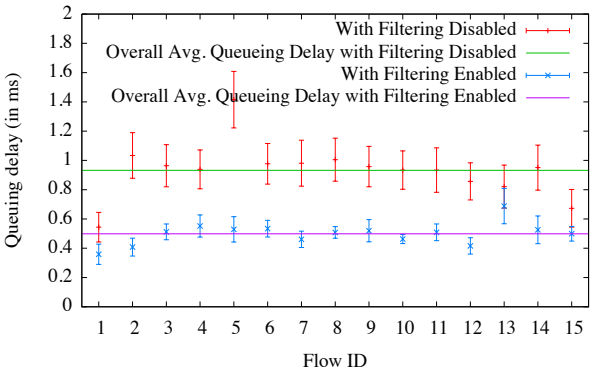



Fig. 5. Comparison of the end-to-end queueing delays with and without cross traffic. The comparison is demonstrated using the 95% confidence interval of each metric.

### B. Stabilizing the Latencies

By enabling and disabling the cross traffic, we compare the end-to-end queueing delays and round-trip times (RTTs) for the flows. All these metrics are measured on the encoder side. The queueing delays and RTTs are measured on a per-packet basis. The end-to-end queueing delay is the difference between an RTT of a packet and the round-trip propagation latency. The comparison is demonstrated by the 95% confidence interval of each metric as shown in Figure 5. When the cross traffic is enabled, the end-to-end queueing delays and RTTs are still kept low and are constrained in very narrow ranges.

### C. Favoring I Frames for Continued Transmissions

| Flow IDs | With Cross Traffic | | Without Cross Traffic | |
|---|---|---|---|---|
| | I Frames | P/B Frames | I Frames | P/B Frames |
| 1 | 60 | 1657 | 42 | 633 |
| 2 | 73 | 2239 | 51 | 771 |
| 3 | 42 | 2000 | 44 | 795 |
| 4 | 61 | 2290 | 50 | 766 |
| 5 | 56 | 1965 | 42 | 709 |

TABLE I
COMPARISON OF THE CANCELLED FRAMES BY FRAME TYPES.

Table I shows the comparison of the cancelled video frames by types for with and without cross traffic. Due to the stringent setting of the deadline (55ms), missing deadlines is possible when without cross traffic. By treating the numbers of cancelled frames by types without cross traffic as the baseline, we examine the cancelled I frames under cross traffic. We can see that the numbers of cancelled I frames under cross traffic are only slightly larger than the corresponding baseline figures, while a large number of the defunct P or B frames are removed from the transmissions. Thus, the frames sitting behind the defunct frames are eligible for transmission at an earlier time.

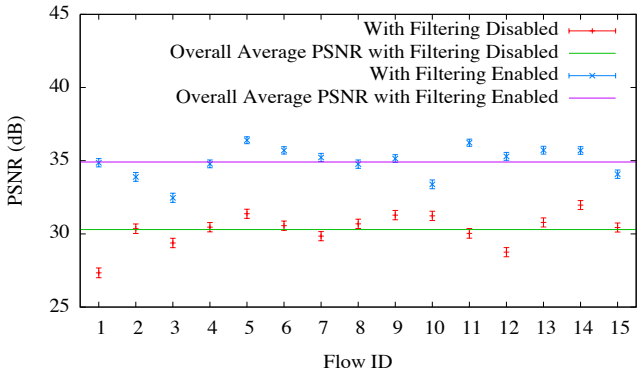



Fig. 7. Comparison of the play-out quality using the 95% confidence intervals of the per-frame PSNR.

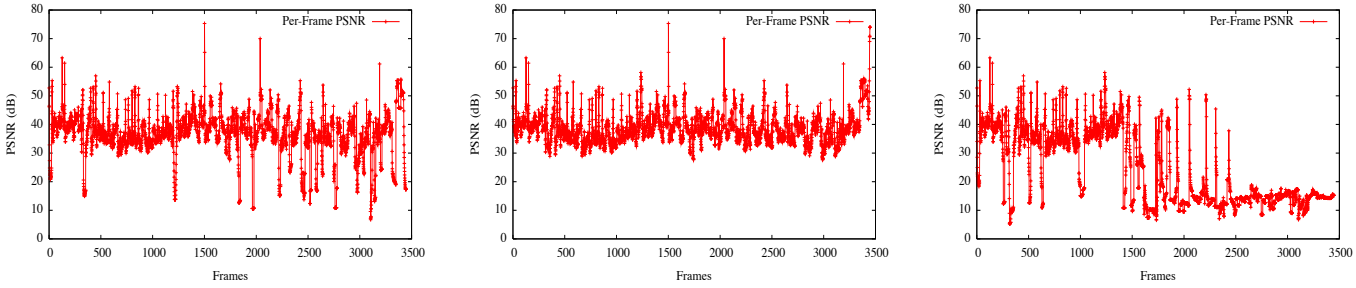
### D. Gracefully Downgrading the Play-out Quality

The comparison of the PSNRs of the encoded video streams with and without cross traffic is shown in Figure 7. The comparison of the PSNRs is demonstrated by the 95% confidence interval of the PSNRs for every video stream. Due to the image compression, a decoder can not restore the original images from the MPEG encoded video frames. The PSNRs of a MPEG encoded image characterize the distortion between the original image before encoding and the corresponding restored image after decoding. The PSNRs of a MPEG encoded video stream exhibit decay in the playout quality even when transmission is not involved, *i.e.*, without any extra transmission delays, or frame losses. When transmission of frames is involved, some frames can become expired. Hence, the play-out quality after transmission should be further decayed in addition to the decays caused by the compressions.

When without cross traffic, our filtering mechanism serves to restrict the additional decay in the play-out quality and make the average PSNRs to be approximately 38.5 dB which is the original decay of the video clip (ref. Figure 7). When with the cross traffic, the PSNR will inevitably further decay due to higher queueing delays. In this case, our filtering mechanism can help to make the further decay to be low. The further decay is at about 10 dB (ref. Figure 7). The frame-by-frame PSNR is also shown for one flow in Figure 6. When without cross traffic, the PSNRs of most frames after transmission are made very close to the original PSNRs, expect for some frames whose play-out quality is further decayed after transmission (ref. Figure 6 (a) and (b)). When with cross traffic, the quality of many frames after transmission is further decayed (ref. Figure 6 (b) and (c)). Without adopting our filtering mechanism, the further decay in paly-out quality can only be much worse.

## VI. CONCLUSIONS

This paper described a cross-layer filtering mechanism which cancels the continued transmission of the frames that can not be properly decoded before their decoding deadlines. This filtering mechanism aims to stabilize the network delay in existence of network congestion by dropping the defunct frames. A well-controlled network congestion facilitates the




(a) Without cross traffic (b) The original PSNR (c) Without cross traffic

Fig. 6. Comparison of the per-frame PSNR of one flow for with and without cross traffic.

frames that sit behind the defunct frames to meet their decoding deadlines. The effectiveness of this filtering mechanism has been examined through experiments in emulated network environments. The experimental results show that the filtering mechanism serve to achieve the following effects. First, removing the defunct frames from transmission serves to stabilize the end-to-end queueing delays. Second, the prioritized video frames (I frames) are relatively favored for continued transmissions than the less prioritized frames (P or B frames), when the network is in congestion. Third, the play-out quality is gracefully downgraded when the network is in congestion.

## REFERENCES


[1] Damir Isovic, Gerhard Fohler, and Liesbeth Steffens. Real-time issues of MPEG-2 playout in resource constrained systems. In *Journal of Embedded Computing*, 1(2):239–256, 2005.

[2] Kavitha Chandra and Amy R Reibman. Modeling one-and two-layer variable bit rate video. In *IEEE/ACM Transactions on Networking*, 7(3):398-413, 1999.

[3] Mehdi Alasti, Kamran Sayrafian-Pour, Anthony Ephremides, and Nariman Farvardin. Multiple Description Coding in Networks with Congestion Problem. In *IEEE Transactions on Information Theory*, 47(3):891-902, Apr 2001.

[4] Gregory J. Conklin, Gary S. Greenbaum, Karl O. Lillevold, Alan F. Lippman, and Yuriy A. Reznik. Video coding for streaming media delivery on the Internet. In *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3):269-281, 2001.

[5] Patrick Seeling, Martin Reisslein, and Frank H.P Fitzek. Layered video coding offset distortion traces for trace-based evaluation of video quality after network transport. In *Proceedings of the International Conference on Computer Communications and Networks (ICCCN)*, San Diego, CA, October 17-19 2005.

[6] Giuseppe Bianchi, Andrea Detti, Pierpaolo Loreti, Claudio Pisa, Francesco S Proto, Wolfgang Kellerer, Srisakul Thakolsri, and Joerg Widmer. Application-aware H. 264 Scalable Video Coding delivery over Wireless LAN: experimental assessment. In *Proceedings of the The International Workshop on Cross Layer Design (IWCLD)*, Palma de Mallorca, Spain, June 11-12 2009.

[7] John D McCarthy, M Angela Sasse, and Dimitrios Miras. Sharp or smooth? Comparing the effects of quantization vs. frame rate for streamed video. In *Proceedings of the ACM conference on Human factors in computing systems (CHI)*, page 542, Vienna, Austria, April 24-29 2004.

[8] Zhi-Li Zhang, Srihari Nelakuditi, Rahul Aggarwal, and Rose P Tsang. Efficient selective frame discard algorithms for stored video delivery across resource constrained networks. In *Real-Time Imaging*, 7(3):255–273, 2001.

[9] Maureen Chesire, Alec Wolman, Geoffrey M Voelker, and Henry M Levy. Measurement and analysis of a streaming media workload. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS)*, San Francisco, CA, March 26-28 2001.

[10] John G Apostolopoulos. Reliable video communication over lossy packet networks using multiple state encoding and path diversity. In *Proceedings of the SPIE Visual Communications and Image Processing Conference (VCIP)*, volume 1, San Jose, CA, January 2001.

[11] Thorsten von Eicken, Anindya Basu, Vineet Buch, and Werner Vogels. U-Net: a user-level network interface for parallel and distributed computing. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, pages 40–53, Copper Mountain, CO, December 3-6 1995.

[12] Nirwan Ansari, Hai Liu, Yun Q Shi, and Hong Zhao. On modeling MPEG video traffics. In *IEEE Transactions on Broadcasting*, 48(4):337–347, December 2002.

[13] Min Dai and Dmitri Loguinov. A unified traffic model for MPEG-4 and H. 264 video traces. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, Miami, FL, March 13-17 2005.

[14] Xiao-Dong Huang, Yuan-Hua Zhou, and Rong-Fu Zhang. A multiscale model for MPEG-4 varied bit rate video traffic. In *IEEE Transactions on Broadcasting*, 50(3):323–334, January 2004.

[15] Dapeng Wu, Yiwei Thomas Hou, W Zhu, Ya-Qin Zhang, and Jon M Peha. Streaming video over the Internet: approaches and directions. In *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3):282–300, 2001.

[16] Min Chen and Gang Wei. Multi-stages hybrid ARQ with conditional frame skipping and reference frame selecting Scheme for Real-Time Video Transport Over Wireless LAN. In *IEEE Transactions On Consumer Electronics*, 50(1):158–167, January 2004.

[17] Christiane Montenegro Bortoleto, Lau Cheuk Lung, Frank A Siqueira, Alysson Neves Bessani, and Joni da Silva Fraga. A Semi-Reliable Multicast Protocol for Distributed Multimedia Applications in Large Scale Networks. In *Springer Lecture Notes in Computer Science (LNCS)*, 3754:109–120, 2005.

[18] Kai-Tat Fung, Yui-Lam Chan, and Wan-Chi Siu. New architecture for dynamic frame-skipping transcoder. In *IEEE Transactions on Image Processing*, 11(8):886–900, January 2002.

[19] Tom Phelan. DCCP-TP—a fresh-start implementation of the Datagram Congestion Control Protocol (DCCP). http://www.phelan-4.com/dccp-tp/tiki-index.php.

[20] Eddie Kohler, Mark Handley, and Sally Floyd. Datagram Congestion Control Protocol (DCCP). RFC 4340 (Proposed Standard), 4340, IETF, March 2006. Updated by RFCs 5595, 5596.

[21] Eddie Kohler, Mark Handley, and Sally Floyd. Designing DCCP: Congestion Control Without Reliability. In *Proceedings of the ACM International Conference on Communications Architecture and Protocols (SIGCOMM)*, pages 27–38, Piza, Italy, September 11-15 2006.

[22] Mark Carson and Darrin Santay. NIST Net–A Linux-based Network Emulation Tool. In *SIGCOMM Computer Communications Review (CCR)*, 33(3):111–126, July 2004.

[23] Jirka Klaue, Berthold Rathke, and Adam Wolisz. Evalvid-a framework for video transmission and quality evaluation. In *Springer Lecture Notes in Computer Science (LNCS)*, 2794:255–272, January 2003.

[24] Michael Kropfberger, Laszlo Boszormenyi, Hermann Hellwagner, and Peter Schojer. ViTooKi: the video toolkit. http://vitooki.sourceforge.net/, 2004.