

Layered Video Coding with Minimum Delay over Cable Modems

Barry Haskell, Glenn Cash, Reha Civanlar

AT&T Labs

Middletown, NJ 07748

bgh@research.att.com, glenn@research.att.com, civanlar@research.att.com

Abstract

It is well known that layered video coding can be employed to optimize the use of guaranteed QoS bandwidth in loss resilient video transport over IP networks. An effective layering technique for this purpose is based on the use of *multiple threads* as defined in ITU-T H.263 standard where some video frames are defined as high priority and the rest as low priority. A problem with a simplistic implementation of this approach, particularly for CATV broadband access where a single cable may be shared by many end points; however, is that several encoders may attempt to transmit high priority base layer frames simultaneously, thus overloading the network with high priority data. This usually results in excessive delays creating problems with interactive applications. This paper presents an analysis of the problem followed by several solution techniques applicable to various implementations.

Introduction

In traditional layered video encoders, part of the video information (the *base* layer) is sent on a high priority transmission channel and part (the *enhancement* layer) is sent on a low priority transmission channel. The network endeavors to send all of the high priority base layer information, while some of the low priority enhancement information may be lost. The base layer can typically be decoded and displayed without any data from the enhancement layer, albeit with reduced quality. One convenient and simple method of layered coding is to designate some video frames as high priority and the rest as low priority. For example, every fifth frame may be sent as high priority with the rest being sent as low priority.

The ITU-T H.263 coding standard [1] in its Annexes N and U provides for a compression using *multiple threads*, wherein the base layer frames are coded independently using only previous base layer frames for motion compensation, and the enhancement layer frames are coded normally using both base and enhancement layer frames for motion compensation. If in the above example, frames 1,6,11, ... are high priority and frame 4 is lost, then the decoder cannot display frames 4 and 5 since frame 5 information depends on correctly receiving frame 4. However, it can resume decoding upon receiving frame 6.

Dealing with Network Congestion via Feedback

A problem with this simple approach, particularly for CATV broadband access where a single cable may be shared by many end points, is that several encoders may attempt to transmit high priority base layer frames simultaneously, thus overloading the network with high priority data. In such a case, either base layer frames may be lost causing drastic quality reduction, or they must be buffered causing excessive delay.

In this paper we propose alleviating the above degradation by using feedback from the network to indicate whether or not capacity is available for sending the *high priority* frames.

In our system, the encoder does not decide *a priori* which frames are in the base layer. Instead, shortly before a candidate base layer frame is to be encoded, the encoder requests from the network permission to send high priority data. If permission is granted, the high priority frame is encoded and sent. If permission is denied, then the encoder could either code and send that frame

as a low priority frame, or delete the frame from transmission and request permission to send the next frame as a high priority frame.

The effect of this mechanism is to randomize the arrival of high priority data into the network, thus minimizing the likelihood of overload, while at the same time maintaining low-delay transmission. An additional benefit is to avoid the buffering of large amounts of data and the ensuing delay that this causes.

Alternative implementations of our approach include the following techniques:

- I. The encoder actually begins to encode and buffer the high priority frame as it requests permission to send. If permission is granted, then buffered data can be sent. If permission is denied the buffered data is deleted, and the encoder waits for the next frame in order to repeat the process.
- II. The encoder attempts to code all frames at high priority. If the network grants permission, then the frame is transmitted as high priority. Otherwise, it is sent as low priority. The advantage of this is that if traffic is low, then the network can award a higher grade of service without additional cost. Clearly, if a frame has to be sent as low priority, it should not be used as a reference for the future high priority frames.
- III. The encoder may be told by the network how much bandwidth is allocated for the high priority frames. It may also know by network feedback or other means, e.g., RTCP [2], if low priority frames are being lost. Then if many low priority frames are lost, the encoder may choose to send more high priority frames, but code them at lower quality while staying within the allocated high priority bandwidth. The net effect will in some cases result in better perceived picture quality.

In all the above cases, the high priority data may be multiplexed onto a guaranteed Quality of Service (QoS) “trunk” that can be established using “differentiated services” [3] and transmitted separately from the low priority data, which would be sent on a best-effort “trunk” over the backbone network.

Layered Video Coding for Cable Modems

If a cable modem has Docsis 1.1 [4] capability, then for layered video coding the network is requested to reserve periodic high priority QoS capacity for the high priority frames. For example if the video frame rate is 15 frames per second and every fifth frame is sent at high priority QoS, then a network service would be requested such that high priority capacity is made available every 1/3 second (333ms) until the reservation is cancelled. In some modes of Docsis 1.1 any unused capacity could be reassigned to other uses.

The above scenario works well if the video encoder has “intimate” connection to the cable modem, i.e., if the video (and audio) encoders are *integrated* with the cable modem. Then any signals available to the cable modem can also be made available to the video encoder.

However, in many implementations the cable modem may be *external* to the video codec. In this case the “network permission signals” may not be available to the video coder. With present-day external cable modems, a high priority QoS video frame would be passed from the encoder to the cable modem where it would be held until the next opportunity arrives for high priority QoS transmission, which in the above example could be as long as 333ms. See Fig. 1. This delay is much too long for high quality interactive conversational services. The purpose of this paper is to enable low-delay operation in the case of *external cable modems* without a need to access the “network permission signals.”

A much lower delay time can be achieved if the video encoder times its high priori QoS frames to coincide with the associated transmit opportunities.

With many transport techniques e.g., RTP [2], the video and audio data packets are time stamped, i.e., data is sent along with each packet indicating with sub-millisecond accuracy the time of day of a system clock running at the encoder. Thus the receiver can examine the time stamps of arriving high priori QoS video packets and compare them with the time stamps of arriving audio packets. Since the audio packets are smaller and sent much more frequently over the QoS channel (considering the relative importance of the audio data, this is a natural assumption), they are assigned much more frequent transmit opportunities and hence do not suffer long delays. If the video packets are suffering excessive delay, then a signal is sent to the video encoder indicating an “excessive delay condition.” The video encoder then redefines the high priori QoS frame times to be one frame later and waits to see if the decoder sends another excessive-delay signal. If it doesn’t, then the high priority QoS frame times remain unchanged. If it does then the encoder lets the high priori QoS frame times slip another frame period later, and so on until the high priority frames occur shortly before the network predefined transmit opportunities and minimum transmission delay is attained. See Fig. 2.

In an alternative approach, the receiver sends to the transmitter the actual measured delay between audio and video packets. The video encoder can then adjust the definition of QoS frame times in one step. Such a measurement can be accomplished using the timestamp relations provided in the sender reports of real-time transport control protocol (RTCP) [2].

Another mechanism is used to *maintain* the synchronization between high priority frames and transmit opportunities. For this the receiver uses a predefined *nominal delay* D between audio and QoS video packets. D is typically a few dozen milliseconds or so depending on the jitter of the network. Denote by P the period between frames, $1/15$ second in the above example. See Fig. 3. During normal operation, if the receiver notices that the QoS frames are arriving more than $D + P$ after the audio packets, it sends a signal to the video encoder indicating a *positive delay*. The video encoder then redefines the QoS frame times to occur one frame period later, thus reducing the delay to approximately D . This is exactly the mechanism described above.

However, if the receiver notices that the QoS frames are arriving less than D after the audio packets, it sends a signal to the video encoder indicating a *negative delay*. The video encoder then redefines the QoS frame times to occur one frame period earlier, thus increasing the delay to approximately $D+P$. Note that if the video frames are actually produced by a standard video camera operating at a higher rate than the *transmitted* video frame rate, then a smaller effective value for P may be used, e.g. $1/60$ second.

Implementation without timestamps

An implementation without using the timestamp processing is also possible and may be preferred by simpler system implementations. In this approach, small flag packets are utilized in minimizing delay. As shown in Fig. 4, here delay synchronization is independent of network congestion as long as most of the QoS packets are received.

In this approach, a small packet, called *Flag1* in Fig. 4, is transmitted to the sending cable modem a short time, say D , after each QoS frame. A second small packet, called *Flag2* in Fig. 4, is transmitted to the sending cable modem one frame period P plus a jitter allowance J after *Flag1*.

During normal operation, shown in Fig. 4, the receiver receives a concatenation of a Flag packet, a QoS frame, a Flag packet and on rare occasion (depending on jitter) another Flag packet. In this case, i.e. when the QoS frame is preceded by one flag, the receiver sends no synchronization signal back to the encoder.

However, if the transmit opportunities drift too far to the right in Fig. 4, then the receiver receives a concatenation of a QoS frame, a Flag packet and usually (depending on jitter) a second Flag packet. In this case, i.e. when zero flags precede the QoS frame, the receiver sends a positive delay signal back to the encoder. The video encoder then redefines the QoS frame times to occur one frame period later, thus decreasing the delay to approximately $D+J$.

If the transmit opportunities drift too far to the left in Fig. 4 then the receiver receives a concatenation of two Flag packets, a QoS frame, and on rare occasion (depending on jitter) another Flag packet. In this case, i.e. when two flags precede the QoS frame, the receiver sends a negative delay signal back to the encoder. The video encoder then redefines the QoS frame times to occur one frame period earlier, thus increasing the delay to approximately $D+P$.

This approach may have its performance improved if additional flag packets are sent to the cable modem at known intervals. Then by examining which flag packets arrive prepended to the QoS frame the receiver can estimate with high accuracy the delay between the QoS frame and its transmit opportunity. This estimate would then be sent back to the encoder to enable it to better synchronize with the network as above.

Further increasing the loss resilience

The loss resilience of the BE enhancement layer may be increased by using other “loss resilient” coding techniques such as Fine Granular Scalability (FGS) of MPEG-4 [5]. These methods decompose the BE enhancement layer information into two or more layers of enhancement information which can be packetized separately. This reduces the effects of packet losses on the received video quality [6].

Conclusions

Use of layered coding to optimize the use of QoS bandwidth in video transport over IP networks is well established. QoS usually guarantees packet delivery but not the end-to-end delay. This may be problematic in interactive applications. We presented a particular case of this when cable modems using Docsis 1.1 standard for QoS are employed. We also showed several solutions for this problem that can be used for various implementation scenarios.

References

1. ITU-T Recommendation H.263, (*Video Coding for Low Bit Rate Communication*).
2. IETF RFC 1889, *RTP: A Transport Protocol for Real-Time Applications*. Audio-Video Transport Working Group, H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. January 1996.
3. IETF RFC 2475, *An Architecture for Differentiated Services*. Network Working Group, S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss. December 1998.
4. *Data Over Cable Service Interface Specifications (Docsis)* Cable Labs. <http://www.packetcable.com/>
5. Information Technology –Generic Coding of Audio-Visual Objects ISO/IEC 14496, International Standards Organization, <http://mpeg.telecomitalia.com/>
6. “Scalable Video Coding,” Chapter 11 of *Video Processing and Communication*, Y. Wang, J. Ostermann, Y.-Q. Zhang, Prentice Hall, 2001.

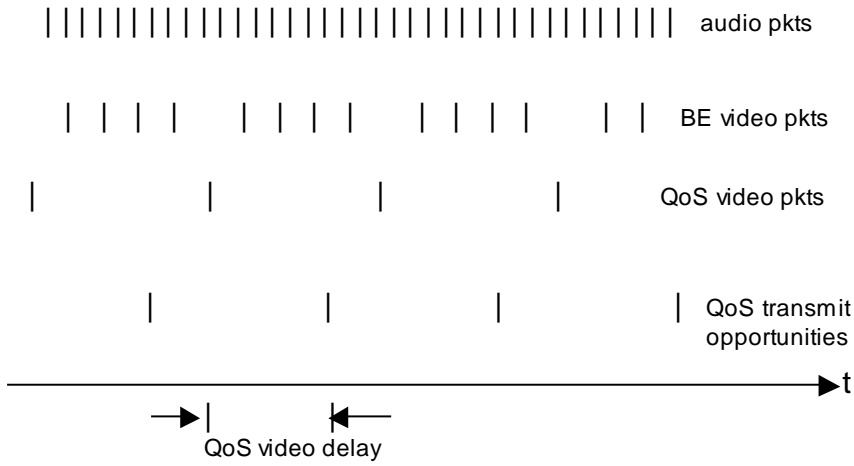


Fig. 1 Large QoS video delay

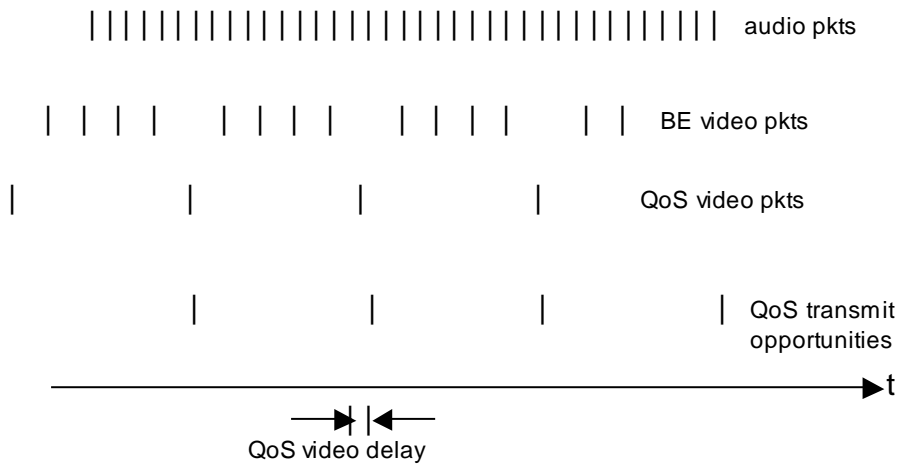


Fig. 2 Minimized QoS video delay

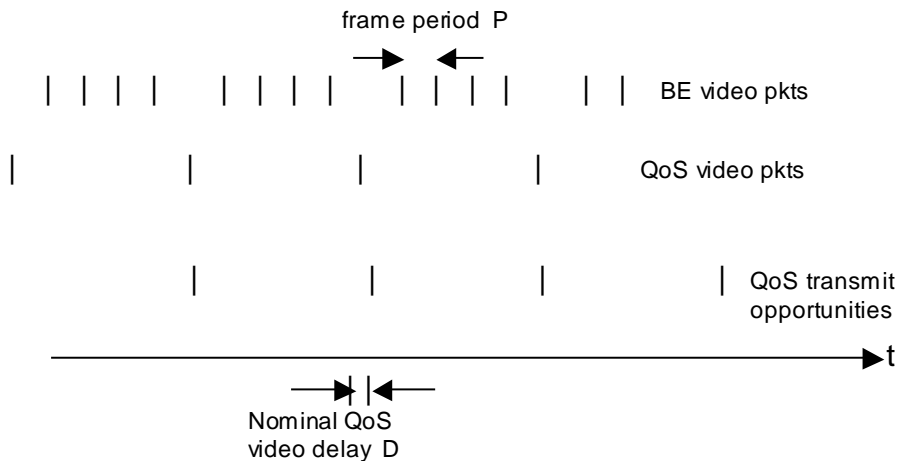


Fig. 3 Maintaining small QoS video delay

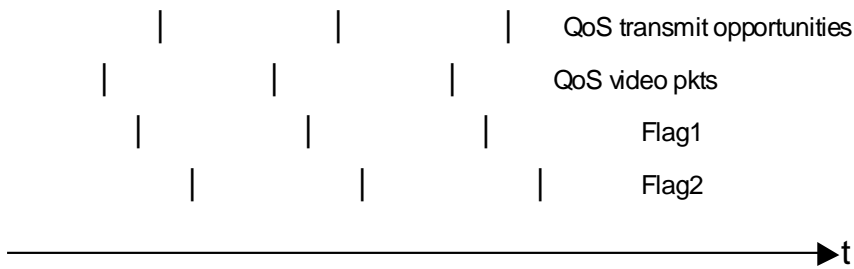


Fig. 4 Maintaining small QoS video delay via flags

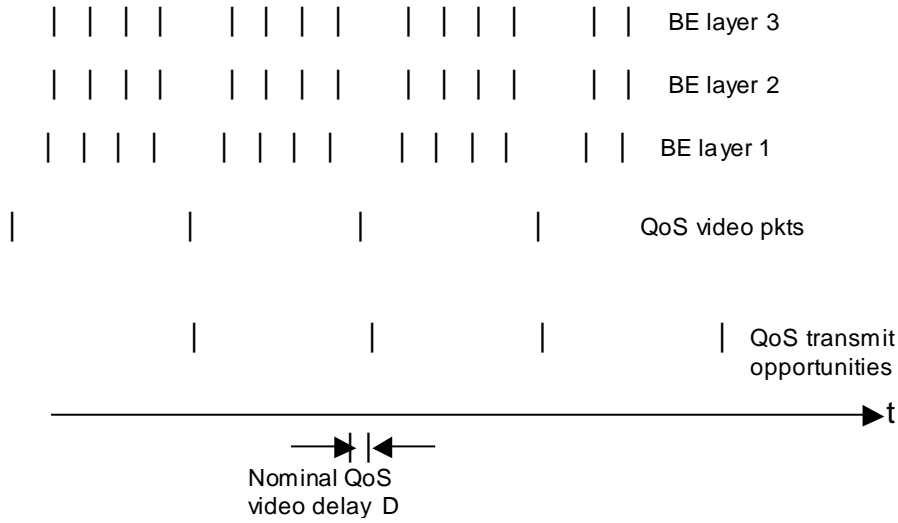


Fig. 5 More than two layers of coding