Adaptive Layered Video Coding for Multi-time Scale Bandwidth Fluctuations

Yuan-Chi Chang\textsuperscript{1} and David G. Messerschmitt\textsuperscript{2}

\textsuperscript{1}IBM T. J. Watson Research Center, P. O. Box, 704, Yorktown Heights, NY 10598
\textsuperscript{2}Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720

October 4, 1999

Abstract

Quality of service of network video is determined by both the amount of available bandwidth and its temporal fluctuations. Except for the premium service, other proposed DiffServ traffic classes such as assured and best effort services are not immune to temporal bandwidth fluctuations. We observe that fluctuations in shared packet switching networks occur in different time scales, from less than a microsecond up to the order of minutes and longer. In the fast time scale (microseconds), bandwidth fluctuations are caused by bit errors, cell loss, and packet corruption, which are associated with noise in physical devices and communication channels. In the medium time scale (milliseconds to seconds), fluctuations are often originated from link layer operations such as routing, queueing, multiplexing and scheduling of communication links. In the slow time scale (seconds to minutes), fluctuations are caused by connection migration among networks with different speeds (e.g. LAN vs. wireless). While temporal fluctuations in the DiffServ model mainly happen at the medium time scale, in this paper we broaden the scope to investigate layered video coding techniques which adapt to multi-time scale fluctuations of connection bandwidth, with emphasis on the medium time scale. We review prior work in dealing with fast and slow time scale fluctuations and argue additional work is required for medium time scale in order to provide a complete solution. We propose delay cognizant video coding and demonstrate that it adapts effectively to medium term bandwidth fluctuations. Delay cognizant video coding segments and compresses video information into layers with variable delay tolerance. When a network link is congested, delay sensitive layers are prioritized over delay tolerant layers, whose packets are queued. Delay tolerant layers resume their transmission as soon as the congestion is cleared. Late arrivals from delay tolerant layers are displayed, according to decoding rules, along with delay sensitive data at the next picture refresh. We found the amount of tolerable delay exceeds 400 milliseconds for videoconferencing type sequences. Delay cognizant video coding complements prior work in layered coding for bandwidth adaptation. With our contribution, layered video now offers a complete solution for adapting to multi-time scale fluctuations.

Keywords: bandwidth fluctuation, layered video coding, network video

1 Introduction

In the proposed IETF DiffServ model, only the premium service may be immune from temporal bandwidth fluctuations. Other traffic classes such assured and best effort services compete
within class of each own for transmission capacity. Temporal fluctuations of connection bandwidth exhibit great impact on quality of service of network applications. Bandwidth fluctuations, however, occur in different time scales, whose durations range from less than a microsecond to the order of minutes and longer. Various factors contributing to the cause of fluctuations are depicted in Figure 1. In the slowest time scale on the order of seconds to minutes, fluctuations are caused by connection migrations among differential networks (e.g. LAN, wireless LAN, PCS, satellite, dialup) or within neighboring links in the same type of networks (e.g. PCS handoffs). The dynamic range of available bandwidth at the slow time scale is the largest, from kilo to giga bits per second. Figure 1 presents the slow term fluctuations by marking link types on segments associated with activities and connection modes such as “indoor, wireless”, “indoor, wired”, “outdoor, fast moving”, etc. Link layer fluctuations, which are reflected to the medium time scale on the order of milliseconds, have much milder variations. Such fluctuations due to routing, queueing, multiplexing and scheduling of communication links often change within twenty percent of the average rate. They are shown in Figure 1 with small but frequent ups and downs on top of big, slow fluctuations. Fluctuations in the DiffServ model mainly happen at this time scale. Finally, at the fastest time scale on the order of microseconds or less, physical layer disturbance such as noise contributes to bit errors and packet corruptions at the probability of $10^{-5}$ or less. The magnitude of fluctuations at this time scale is much smaller so they are not shown in Figure 1.

This paper examines the impact of multi-time scale bandwidth fluctuations on network
video services and discusses a specific technique to address medium time scale fluctuation. It is long recognized that the coding and delivery of video and other media content must be adaptive. Adaptive video coding has been a major focus of research in recent years. Rate control techniques have been developed to follow fluctuations of available bandwidth. For point-to-point connections, online measurements and feedback enable closed loop bit rate control to match link bandwidth closely. The effectiveness of closed loop control relies on the timeliness and accuracy of the feedback, which may be sufficient for slow to medium time scale fluctuations. Fast time scale fluctuations are remedied by error control mechanisms such as forward error correction. When a video distribution server is not equipped with the above capabilities, a separate proxy server can be set up to transcode the content, which is essentially a concatenation of decompression followed by adaptive compression [5] [12]. Although such per-connection rate control solution is effective and immediately available, it is not scalable to broadcast or multicast scenarios because of the solution’s dependence on sender-initiated per-connection control.

Adaptive layered video coding resolves the scalability issue through receiver-initiated open loop rate control, where receiving devices determine the proper bit rate/quality based on bandwidth availability, device capability, and possibly usage policies. Figure 2 depicts an example in which receiving devices range from high definition TV (HDTV) to laptop computers. They are connected to the Internet through links at various speeds. From this picture, it is clear that traditional video compression model of generating a single bit stream cannot satisfy differential
quality demands nor meet connection characteristics of the wide variety of devices. Fixes do exist, however, in today's Internet. One option is to provide multiple versions of the same video at the sender site (e.g. 28K, 56K, ISDN). Another option is to install transcoding proxies at subnetworks, like cellular networks. Both approaches work only for a small, limited number of access links. For video deliveries with fine-grained bit rate allocations, potential choices are too many to be satisfied with a few pre-specified versions.

One exemplary usage paradigm of layered video coding can be best described in Figure 3. Raw image sequences are compressed and packetized into multiple bit streams. In general, these streams have an order of dependence which require a decoder to uncompressed an ordered subset starting from the base stream. For the ease of description, Figure 3 depicts three bit streams from the video source. The base stream is labeled $S_1$ and is delivered to all three devices. $S_1$ contains the most essential video information at the lowest spatial and temporal resolution acceptable. Information carried by $S_2$ is built upon $S_1$ to enhance resolution and quality. $S_3$ adds more information in additional to $S_1$ and $S_2$ to further enhance quality. Depending on their device capabilities and connectivity, receivers adjust the number of bit streams they subscribe. The more bit streams they receive, the more bandwidth and computing power they need. The number of bit streams can be fairly large (> 10) to provide sufficient choices [13].

Notice in Figure 3, bit rate control is open loop and driven by receivers. The video source only generates the bit streams and is relieved from monitoring and controlling bit rate for each connection, thereby making the paradigm scalable. This usage paradigm was first proposed in [8], whose wavelet based compression algorithm is coupled with multicast mechanisms to distribute streaming video on the Internet.

Having established the receiver-driven, open loop rate control model, we next examine related work in layered video to see if a solution already exists for multi-time scale bandwidth fluctuations. The aforementioned type of layered video coding proposed in [8] and [13] mainly addresses rate control at the slow time scale. While the number of receiving layers can be adjusted dynamically, adding or removing layers does not happen instantly. A receiver needs to wait for the next synchronization point in bit streams, which may be inserted at the regular intervals of seconds for compression efficiency reasons. Bit streams need to be perfectly synchronized at the receiving end or they are considered lost. These algorithms were not proposed with unequal error protection features. This type of layered video is named rate scalable video coding since the emphasis is on the long term, average rate control.

The second type of layered video coding seen in the literature focuses on error resilience, which is named error resilient video coding [17] [16] [18]. Error resilient coding generates bit

\footnote{In the cited publications, certain error resilience mechanisms can be introduced to increase the robustness of bit streams.}
streams which can be decoded even with occasional bit errors or cell loss. It is known that in compressed video streams, certain bits belonging to motion vectors and low spatial frequency components are more visually significant than others. These visually significant bits, when lost or corrupted, cause more displeasing effects to viewers. An important subject of error resilient coding is to identify and allocate redundant bits to protect the significant bits. Although prior work in error resilience does not always use the term layered coding, through unequal error protection to subsets of bits in the compressed stream, two or more layers are formed logically, if not physically.

Since error resilient video coding is most effective in combating cell loss and bit errors, it is suited for adapting to the fast time scale fluctuations of a link. Rate scalable and error resilient video coding are complementary and one is not replaceable by the other. A joint design can make the compressed bit streams to be adaptive to both slow and fast time scales bandwidth fluctuations.

Neither of the two aforementioned layered coding techniques address medium time scale fluctuations adequately. For example, when a link becomes temporarily congested, either because of bursty incoming traffic or because of physical channel degradation, the packet scheduler may have to buffer packets. The available bandwidth is temporally reduced. When the normal transmission resumes, buffered packets arrive at the receiver late. Normally, these late packets are discarded and treated as packet loss. In the case of rate scalable bit streams, the decoder would have to wait for the next synchronization point of the discarded layer. In the case of error resilient bit streams, the decoder might not be able to recover from consecutive packet losses at all, since built-in redundancy is not usually robustly enough against clustered errors.
A different layered video coding is needed to adapt medium time scale fluctuations. Since bandwidth fluctuations at this time scale create delay jitter to packet arrivals, the new technique must be able to take advantage of the information carried by those packets which arrive late. For normal video playback rate at 30 frames per second, jitter over 33 milliseconds is unacceptable to conventional video coding. However, on packet switching networks, accumulated queueing jitter does occur at the scale of tens of milliseconds. Since jitter is difficult to control, we need to seek an alternative solution that does not always discard late packets and the information they carry. This alternative solution begins by reexamining the traditional synchronous frame display model, in which all pixels are refreshed simultaneously at regular intervals. Should we relax this synchronization constraints and assign pixels which do not change significantly to late packets, it is then possible to make use of all packets regardless of early or late arrivals.

This paper proposes the third type of layered video coding to address the issues of packet delay jitter. We named it “delay cognizant video coding” to emphasize the delay aspect it addresses. Differing from two other layered video, delay cognizant video coding segments and compresses video information into layers with variable delay tolerance. When a network link is congested, delay sensitive layers are prioritized over delay tolerant layers, whose packets are queued \(^2\). Delay tolerant layers resume their transmission as soon as the congestion is cleared. The decoder at a receiver does not refresh all pixels at regular intervals. Instead, it refreshes delay-sensitive video information regularly and refreshes tolerant video layers less frequently. We found delay cognizant video is capable of combating jitter exceeding 400 milliseconds for videoconferencing type sequences. It is ideal to be used for adapting medium time scale fluctuations.

Having motivated our work, in the rest of the paper, we first review related work in layered video coding, including techniques to achieve rate scalability and error resilience. We then present the encoding and decoding algorithms of delay cognizant video coding in Section 3. Its performance characterizations are presented in Section 4. Having established delay cognizant video, we then discuss how these three types of layered video may be mapped to a connection exhibiting multi-time scale fluctuations in Section 5. Section 6 concludes the paper with our vision of new challenges that adaptive layered video coding brings to the networking research community.

\(^2\)We assume two or more levels of prioritized transmissions are supported by the network. Packets are tagged by the sender before they are ingested.
2 Related Work

Figure 3 depicts the basic principle of layered video as a partition of video information into multiple layers, with dependence among them. This principle covers all three types of layered video discussed in this paper although each way of partitioning has its own goals. While the emphasis of this paper is on delay cognizant video and its adaptability to medium time scale fluctuations, techniques commonly used in rate scalable and error resilient layered video are summarized in this section to address slow and fast time scale fluctuations.

Rate scalable layered video partitions video information into layers with differential bit rate requirements. In MPEG-2 terminology, scalability includes the following four types: temporal scalability, spatial scalability, signal-to-noise ratio (SNR) scalability, and data partitioning [9]. Temporal scalability may be achieved by having the base layer carrying information with a lower frame rate and enhancement layers carrying incremental frames. The granule of temporal scalability rarely attains partitions of more than four layers. Spatial scalability is obtained by having the base layer carrying coarse, spatially subsampled images and enhancement layers carrying higher resolution data. From HDTV quality down to thumb nail images, spatial scalability can easily generate ten or more layers. SNR scalability seeks for a progressive quantization of signal amplitudes, where the base layer carries the most significant bits and enhancement layers carry fine details. Lastly, in transform or sub-band coding techniques, low frequency bands are known to be more visually significant. Their coefficients can be put into the base layer while coefficients of high frequency bands are put into enhancement layers. It has been demonstrated that a proper mix of the four scalability options enables rate scalable video to generate tens of layers with little overheads [10] [13] [14].

Rate scalable layered video is best suited for slow time scale fluctuations, where bandwidth variations are large but happen infrequently. Dependence coding between and within layers is the main reason that makes rate scalable video unsuitable for two other time scales. If a packet from layer $N-k$ is lost or delayed, the decompression of layers $N-k$ to $N$ will not be able to continue. Furthermore, the decompression only resumes at the next synchronization point of layer $N-k$, because synchronization points are inserted infrequently for efficiency reasons. For lossy links, error resilient coding comes to rescue.

Error resilient video coding has a long research history dated back to early inventions of video compression and error control [6] [17] [16]. It is long recognized that unlike the 100% accuracy requirement on text or numerical data, multimedia like audio, image, and video can suffer serious loss and still convey carried information at the expense of quality degradation. It is the quality, not the fidelity, that characterizes performance. Therefore media error control aims at protecting visually significant information to achieve high quality when losses occur.
The study of human visual systems reveals that not all video information affects visual quality equally [15]. Image and video processing has been exploiting differential visual sensitivities to remove redundancy and improve compression efficiency. For example, it is known that human vision is more sensitive to luminance than chrominance changes. Bit losses in chrominance planes are thus remedied by interpolating lost values from neighboring pixels temporally or spatially.

Error in communication channels can be categorized into bit corruptions and losses. Corruptions refer to bit errors happened randomly which damages the integrity of a packet. Forward error detection and correction techniques are commonly used to detect and recover corruptions. In addition, bit interleaving in time or channels is effective against clustered errors, provided that the interleaving interval is greater than the duration of error clusters. If the video information is completely lost, automatic retransmission request (ARQ) may be applied end-to-end by requesting the same packet to be resent to the receiver \(^3\). Finally, error resilience can be achieved by simply keeping some visual or statistical redundancy in the compressed bit stream. For example, inter-coded frames (difference frames, P or B frames) can be replaced by intra-coded frames (I frame) so block errors are limited locally rather than propagating from frame to frame. Keeping redundancies, however, adversely affects compression efficiency at the expense of robustness.

Error resilient layered video is best suited for fast time scale fluctuations because techniques mentioned above do not respond well to wide bandwidth swings occurred in slow and medium time scale. Integrating rate scalable and error resilient layered video coding is possible. One can add various amounts of redundancy directly to bit streams generated by a rate scalable coder to provide unequal protection. Another option is to further decompose these streams and offer levels of redundancy. Should SNR scalability and data partitioning be used, the bit streams already exhibit different resilient levels, with the base layer being least robust. In a service prioritized network, the base layer shall be assigned with the highest priority to transfer reliably.

Since neither rate scalable nor error resilient layered video are effective solutions for medium time scale fluctuations, we recognized the need for delay cognizant layered video, which is detailed next.

\(^3\)Simple ARQ is not a scalable solution in multicast or broadcast environments. Reliable multicast often relies on a combination of ARQ and request aggregation.
3 Delay Cognizant Video Coding

Delay cognizant video coding (DCVC) generates multiple layers which are differentiated by their delay requirements. The amount of jitter between the least and most delay tolerant layers is around several hundred milliseconds. This makes DCVC suitable for dealing with medium time scale fluctuations. We assume layers are prioritized from the least to the most tolerant and networks buffer low priority layers when congestion happen. The least delay tolerant layer is given the highest priority while the most tolerant layer is given the lowest priority.

In order to widen jitter tolerance between DCVC layers, we must abolish the synchronous display model commonly assumed in traditional video processing. The synchronous display model refers to the approach that the capturing, compression, decompression, and rendering of video information all happen at fixed time intervals (frames). Pixels captured in one frame are always displayed at the receiver in the same frame. It is the synchronous display model which forces the delivery of video data to be jitter free. However, the *jitter-free* requirement is at odds with *jitter-prone* packet switching networks.

DCVC assumes the *asynchronous* display model in which pixels captured at one frame may be separately displayed at several frames, which lifts the jitter-free constraint. Similar to computer graphics, image regions are asynchronously rendered like graphical objects. There is no longer an objective measure of latency, since a one-to-one mapping does not exist between a captured frame at the sender and a composite frame at the receiver. The focus is thus on perceptual delay - the delay perceived by end users ⁴ One goal of DCVC is to put delay sensitive video information into the lowest delay layer such that the perceptual delay will be primarily determined by the latency experienced by the lowest delay layer.

Although the number of differential delay layers is arbitrary, there is a tradeoff between the layer-incurred overhead and the benefits brought by multiple layers. In this paper, we focus on two layers with finite delays, low- and high-delay layers, recognizing that this can be easily generalized. At the network side, video layers may be mapped to flows proposed in Internet Protocol version 6 [1]. Since there is a logical mapping between a layer and a flow which carries the video layer, in the rest of the paper, we use these two terms interchangeably.

Ideally, a DCVC algorithm achieves two objectives. The first objective is to reduce traffic in the low-delay layer so that minimal network resources are required to support its tight delay and jitter requirements. Traffic in the high delay layer has relaxed delay bounds, which gives the network the most flexibility in transmission prioritization and scheduling. The second objective

---

⁴ Perceptual delay is a subjective measure of delay in video communications. It can be quantified by the end-to-end latency of perceiving a motion event, such as nodding or gestures, happening at the sender side. It can also be characterized by the relative delay to the associated audio signal for maintaining lip sync.
is to maximize the delay offset (difference in latency) allowed between the two layers while visual quality is still acceptable. Visual artifacts are expected to occur with this asynchronous display model since pixels from several frames are composed into one. However, if pixels in the low-delay layer were chosen intelligently, quality degradation could be reduced. These two objectives were concluded from an argument based on the convexity of rate-distortion curves and conjectures on video quality in the asynchronous display model. The argument is detailed in [4]. During the course of the study, we were fortunate to collaborate with experts in human vision to conduct experiments and improve the design. However, unlike studies in the synchronous display model, prior vision research has revealed little about the kind of video information that has the most impact on quality when delayed. It is an important issue in need of more research.

The partition (segmentation) of video information into delay flows thus has been explored through heuristics rather than rigorous mathematical formulation. This methodology is the result of insufficient understanding of human vision to characterize video quality, especially in the case of asynchronous display. Our remedy for the situation, also taken by standard-issuing committees, is to conduct subjective quality evaluation. Details of prior video segmentation schemes are reported in [2] [11]. Results of subjective quality measurements are reported in [3]. In the following we discuss the architectures of encoding and decoding algorithms that are considered to have the most success in achieving the aforementioned two objectives.

3.1 DCVC Encoder

The DCVC encoder shown in Figure 4 is divided into two stages: segmentation and compression, each of which is framed in the figure. A video frame is first processed by the segmentation stage to extract the low-delay information. High-delay data is obtained by subtracting low-delay data from the video frame. Extracted flow data is then passed to the compression stage to remove spatial and temporal redundancy. We describe the algorithm in the order of the processing: segmentation first and then compression.

We applied a block-based segmentation amid its low addressing overhead. A captured video frame is first divided into blocks of size 8 by 8. Each block is then independently assigned to either the low- or high-delay flow, based on the spatio-temporal frequency variation in the block. The flow diagram of this stage is marked and shown in the left frame in Figure 4.

The segmentation algorithm identifies visually significant blocks by measuring the similarity of a block and its predecessor in time. A block of size 8 by 8 is first transformed to the frequency domain. Each frequency coefficient is compared against its corresponding threshold and there are a total of 128 test conditions, all of which must be satisfied for a block to be assigned to the high-delay layer. The 128 conditions are composed of two conditions each for every Discrete Cosine Transform (DCT) coefficient of the tested block, which has 64 coefficients. Since each
coefficient is independently tested, it suffices to look at just one pair of such conditions:

Condition 1: \( |P_{i,j,n,t} - P_{i,j,n,t-1}| < V_{i,j} \)

Condition 2: \( |P_{i,j,n,t} - P_{i,j,n,update}| < V_{i,j} \)

In the above expressions, \( P_{i,j,n,t} \) is the \((i,j)\)th DCT coefficient for block \( n \) at time \( t \); \( P_{i,j,n,update} \) is the \((i,j)\)th coefficient of block \( n \) stored in a buffer for the latest update; \( V_{i,j} \) is a fixed preset threshold for the \((i,j)\)th coefficient. The 8x8 threshold table of used in all the reported experiments is listed in Table 1 (for 8 bit pixels).

The first condition is to limit the variation of spatial frequencies in two consecutive frames. The subtraction operation can be viewed as a two-tap high-pass Haar filter operating in the temporal dimension. The second condition is to limit the variation relative to the latest update that is the last block assigned to the low-delay layer. The two threshold blocks in Figure 4 are marked as Condition 1 (C1) and Condition 2 (C2), respectively.

We found temporal variations of a block consist of steep changes as well as small perturbations as depicted in Figure 5. Steep changes are typically originated from movements of objects with sharp contrast while small perturbations may come from slow variations of textures. To minimize visible artifacts, steep changes cannot be ignored and DCVC must act immediately by updating the block (region) with the low-delay layer. What DCVC can take advantage of are the small perturbations, which can be delayed without causing strong artifacts. The first condition (C1) is to monitor steep changes of a coefficient in consecutive frames. This condition
Table 1: The 8x8 table of thresholds for detecting DCT coefficient changes; DC value is at the upper-left corner.

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>15</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>45</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>15</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>45</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>15</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>45</td>
<td>45</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>45</td>
<td>45</td>
<td>45</td>
<td>45</td>
<td>45</td>
<td>45</td>
</tr>
</tbody>
</table>

Figure 5: An illustration of the two threshold conditions; the curve represents a time-varying frequency coefficient.

only, however, is insufficient, as we found in experiments. In some scenes, the changes are more mild and gradual, which may not violate C1 that is set to detect steep changes. These changes, when accumulated over time, become more significant to the extent that they cannot be delayed. The second condition (C2) is designed to capture these gradual changes (or drifts). It limits the size of perturbations with respect to the latest update in time (block assigned to the low-delay layer).

The choice of detection thresholds $V_{i,j}$ directly affects the performance of DCVC. More blocks will be assigned to the high-delay layer, which is our objective, by simply increasing $V_{i,j}$. However, this would cause significant visible artifacts especially for low spatial frequencies. Through informal viewing experiments, we noticed higher spatial frequency components could tolerate a greater temporal variation without adversely affecting quality. Our observation is consistent with human vision studies on the roll-off of contrast sensitivity function at high spatial frequencies. The $V_{i,j}$ matrix we applied was based on subjective quality experiments
and could be further refined.

As indicated in the block diagram of the segmentation stage in Figure 4, two threshold units control the switch to selectively update blocks through the low-delay layer. Those blocks are then inverse transformed back to the spatial (pixel) domain and are put into the low-delay image plane for compression. An example is shown in Figure 6, where three frames of the Suzie test sequence are shown with low- and high-delay images. Areas in black have no pixel values. The low- and high-delay images can be added to obtain the original. It is not surprising to see most of the background resides in the high-delay portion. It is also worth noting that although the person in the picture is moving, some of the high-textured regions, such as hair area, are segmented to the high-delay layer. High-textured areas have high spatial frequency and mild movements are not very visually sensitive.

For head-and-shoulder scenes appeared in videoconferencing applications, the low-delay image plane typically consists of 10 to 20 percent of the total blocks. The percentage is observed to vary significantly depending on the motion content of the sequence. In the case of scene changes, the whole frame is sent to the low-delay layer. The corresponding high-delay image plane consists of blocks going to the high-delay flow. The generation of the high-delay plane is related to the compression stage, which is described next.

The compression stage as shown at the right frame of Figure 4, removes spatial and temporal redundancies from the segmented video layers. Since redundancy removal creates data dependency, compressing the low-delay layer with reference to data in the high-delay layer is not allowed. Should the data dependency constraint be violated, low-delay data cannot be decompressed ahead of high-delay data, leading back to the synchronous decoding. The compression of the high-delay layer, however, can reference data in the low-delay layer. The current encoder is chosen to keep the compression of two layers independent because we did not found significant compression gain to justify the additional complexity of establishing cross-layer reference. Cross-layer reference may be justified when more layers are generated.

The compression stage complies with the dependency constraint by using two separate en-
coding loops. Our design adopts the motion estimation (ME) with discrete cosine transform (DCT) architecture [9]. The upper (lower) ME loop in the diagram corresponds to the compression of the high- (low-) delay layer. The reconstructed low-delay image plane in the ME loop is fed back to the segmentation stage. The feedback is stored as the latest update for the second segmentation condition described previously. The high-delay image plane is obtained by subtracting the reconstructed low-delay image plane from the input video frame. This allows quantization errors in the low-delay ME loop to be passed as a part of the input to the high-delay ME loop.

Motion-compensated video coding forms the basic architecture of all existing and some in-developing video compression standards, which include MPEG1, MPEG2, part of MPEG4, H.261, and H.263. Rather than treating video as a three-dimensional signal, it is often found more effective to describe scenes in successive video frames with simple transformations such as translation, zooms and pans. While the first video frame is compressed using image coding techniques, its following frames can often be deduced by indicating how image regions move. The presence of motion structures suggests ways to achieve high compression by using the motion estimation and compensation.

To apply motion estimation, the compression algorithm first divides a frame into $N$ by $N$ blocks (typically $N = 8$ or 16). It then performs a search for the best match from the previous reconstructed frame. The difference block (prediction error) between the current block and its best match is compressed using transform coding. To increase matching precision, a half-pixel interpolation is often performed on the reconstructed frame first. Motion characterization then extends the precision from pixel to half pixel. Depending on targeted applications, the aforementioned video compression standards define different modes of motion estimation, such as forward, backward, and bi-directional prediction. Different code tables are applied to compress the motion vector, which records the relative location of the best matched block. The decoder uses the motion vector to find the same prediction and add it to the decoded prediction error.

Transform coding of the difference block often applies discrete cosine transform (DCT) for its near optimality in compacting energy. The DCT coefficients are then quantized using a uniform quantizer with a zero zone to minimize the number of nonzero coefficients. Quantized coefficients are then encoded by run level encoding, which records for each nonzero coefficient, the number of zeros before it and its quantized value.

In our implementation, we applied the code tables of the H.263 standard for motion vectors and run level coding. This is because videoconferencing is one of our target applications and some of the test sequences we acquired were H.263 test clips. With the same quantization step sizes applied to both layers, the compression efficiency is listed in Table 2. Four test sequences of head-and-shoulder scenes were encoded, all of which except the Carphone sequence
Table 2: Average bits per pixel for the low- and high-delay layers; As a comparison, H.263 outputs are listed at the last column.

<table>
<thead>
<tr>
<th>Video sequence</th>
<th>Low-delay layer</th>
<th>High-delay layer</th>
<th>H.263</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suzie</td>
<td>0.063</td>
<td>0.032</td>
<td>0.073</td>
</tr>
<tr>
<td>Salesman</td>
<td>0.035</td>
<td>0.018</td>
<td>0.043</td>
</tr>
<tr>
<td>Carphone</td>
<td>0.132</td>
<td>0.031</td>
<td>0.153</td>
</tr>
<tr>
<td>Miss America</td>
<td>0.028</td>
<td>0.024</td>
<td>0.034</td>
</tr>
</tbody>
</table>

have still backgrounds. The low-delay layer is approximately 50 to 80 percent of the total compressed traffic although it carries less than 20 percent of the blocks. This is mainly because the compression of the high-delay layer is far more effective. We expect further optimization in compression to reduce the total bit rate as well as the low-delay portion.

A direct compression on the image planes as shown in Figure 6 does not turn in competitive compression efficiency because artificial block boundaries create many high frequency residues in prediction errors. An effective solution, as we found, is to fill the empty regions (indicated by black areas) with the same blocks from the reference frame in the ME loop. Pixel values are copied from the reference frame to the no-value regions. Since the first frame is always intra-coded, area filling guarantees all successive frames have no empty regions. Area filling operations are performed ahead of compression, as indicated by the AF blocks marked in Figure 4. Area filling improves compression efficiency by preserving the shapes of image objects and smoothing artificial block boundaries. Blocks used in filling the empty area are not compressed for they are copied from the reference. A one-bit indicator per block is used to inform the decoder if the coded block belongs to the low-delay flow. The addressing overhead before entropy coding is one bit for every block of 64 pixels, or 0.016 bpp.

At the outputs of both ME loops, rate control modules monitor output rates to ensure compliance with the target rate. The quantization step sizes of the two ME loops need not be the same and this flexibility can be exploited to improve video quality. One possibility is to control the number of high-delay blocks in a frame. With a fixed bit rate, decreasing the number of blocks to be compressed increases their coded quality. In the high-delay layer, blocks are selected based on their past history with the assumption of delay inertia. We presume those blocks that are less frequently updated through the low-delay flow tend to stay that way. These blocks stay on the receivers screen for a long time and thus need a higher visual quality. The rate control module keeps an age record of blocks and prioritizes the selection in the descendent order of block ages. The age of a block, incremented at each frame, is reset to zero when it is either updated as a low-delay block or selected for the high-delay layer. The rate controlled
encoding of blocks can be viewed as the video extension of progressive image transmission. A
block is updated through the low-delay layer to establish a coarse initial representation. It is
then replenished through the high-delay flow with a finer version. Unlike image coding, video
does not allow progressive coding for the whole frame. However, progressive coding on video
regions is possible as demonstrated by DCVC. For high texture, slowly varying regions, the rate
controlled compression works very well.

3.2 DCVC Decoder

The DCVC decoder block diagram shown in Figure 7 is also divided into two stages: the
independent decompression of layers and their video composition. We do not elaborate further
on the decompression stage since it mirrors the compression loops in the encoder. Instead we
focus on describing how the two video layers are composed for final rendering.

The DCVC decoder follows a simple set of rules to display received blocks. Compressed bit
streams from both layers are tagged with temporal references (frame numbers). The decoder
maintains one temporal reference table for each layer, in which each entry stores the frame
number of the received block at the coordinates. The tables are initiated to zero and blocks
from earlier frames are replaced with those from later frames. By comparing $TR_{n,L}$, temporal
reference of the $n$th block from the low-delay flow, and $TR_{n,H}$, temporal reference of the $n$th
block from the high-delay flow, the decoder makes the following decision:

1. $TR_{n,L} > TR_{n,H}$, display the block from the low-delay layer;
2. $TR_{n,L} = TR_{n,H}$, display the sum of two blocks;
3. $TR_{n,L} < TR_{n,H}$, display the block from the high-delay layer.

Due to a nonzero delay offset between the two flows, which is caused by queueing and
scheduling of the prioritized flows, it is possible that a block from a later frame may be trans-
mitted through the low-delay flow but it arrives at the decoder earlier than its precedents in the
high-delay flow. By the decoding rules, this block preempts its precedents and it is rendered
upon arrival. The occurrence of preemption is due to significant changes of some spatial fre-
quency components, as the segmentation is designed to detect. Movements of objects and scene
changes typically cause those changes. Block preemption can, under certain conditions, be used
to further improve the efficiency of the network by annihilating obsolete blocks.

4 Delay Tolerance and Video Quality

The layers of delay cognizant layered video need to have jitter tolerance on the order of tens to
hundreds of milliseconds to adapt to medium time scale bandwidth fluctuations. We discovered
that except for scenes containing large motion, information carried by the high-delay layer contributes to video quality. This demonstrates that DCVC indeed adapts to medium term fluctuations well. The performance characterization was executed in two ways. First, we applied the objective measure peak-signal-to-noise ratio (PSNR) and compare the composed frame at the decoder with the original, uncoded frame at the sender. Second, we conducted subjective quality evaluation by inviting naive human viewers to rate video quality. In the second case, we varied the jitter tolerance measured by the delay offset between the two flows from zero to four hundred milliseconds.

In the first experiment, we assume a single node link where multiple video sessions are ingested. DCVC is compared against the single bit stream output of a H.263 encoder. Quantization step sizes are fixed, which causes the output bit streams to be variable bit rate. We applied a two-priority-class scheduling policy to DCVC traffic. The low-delay flow is always prioritized over the high-delay flow. The H.263 single bit stream is scheduled by the first-in, first-out FIFO policy. The link speed of the node was simulated as a T1 line at 1.5 Mbps. Since both DCVC and H.263 streams are not equipped with error resilient layers, we assume a small loss probability of $10^{-6}$. Our simulation added the number of H.263 bit streams gradually until loss started to occur. At that point, we fixed the simulated number of network video connections. At the average bit rate around 30 Kbps, we found the simulated link could only support ten to eleven concurrent connections partly because the high peak-to-average rate ratio is very high (around 20).

The quantization step sizes were different in H.263 and DCVC to make their long term average rates approximately equal. The H.263 compression applied a step size equal to 16. The low-delay layer of DCVC applied a step size equal to 20 and the size was set to 10 in the high-delay layer. There is a nonlinear relation between quantization step size and bit rate. The larger the step size, the lower the bit rate.

We simulated eleven video connections based on the above setting and recorded the output

---

**Figure 7:** Schematic block diagram of the DCVC decoder.
Figure 8: The figure plots PSNR of each frame in a 15-sec salesman sequence. The solid line plots measured values of DCVC. The dotted line plots measured values of H.263. Except for the first second, the two-layer DCVC delivers better quality by taking advantage of medium time scale fluctuations.

video trace. Outputs from both H.263 and DCVC of one representative trace are plotted in Figure 8. We applied the PSNR measure and found DCVC delivered better quality by taking advantage of the *spare* capacity released when the high-priority queue is empty. The jitter tolerance is as much as 330 milliseconds. The 330 millisecond delay also explained the quality degradation at the first second shown in this figure. In the first half second, the low-delay layer, which was compressed at lower quality, was the only information arrived at the decoder. As the high-quality high-delay data started to arrive, overall quality started to improve.

It is known that mean-square error based PSNR measure does not fully reflect human perceived quality. We feared the above measurement might mislead us. We thus conducted subjective quality evaluations, under the instruction of experts in human vision science. In subjective evaluations, we measured the quality impact of asynchronous rendering by varying the jitter tolerance. While human viewers only gave one rating for a 10-sec long video instead of one rating per frame depicted in Figure 8, we found the ratings were statistically insignificant to
differentiate video clips made with small jitter tolerance. We found that jitter tolerance as much as 400 milliseconds was acceptable. The detailed procedure and analysis of subjective quality evaluations are described in [3].

While the experimental results are encouraging, we are optimistically cautious about drawing this conclusion to other types of video clips since we only conducted experiments on videoconferencing sequences. For music video or action films which contain frequent scene changes and a lot of motion, we expect very little video information can be put into the high-delay flow. In those cases, video compression techniques are also less effective since there is very low temporal redundancy to be removed.

5 Mapping of Video Layers

We have shown delay cognizant video coding detailed in Section 3 generates layers differentiated by their delay requirements. Experimental results suggest maximum jitter tolerance can be as much as 400 milliseconds, which makes delay cognizant video suitable for adapting to medium time scale bandwidth fluctuations. Since fluctuations at two other time scales are effectively solved by error resilient (fast time scale) and rate scalable (slow time scale) layered video, in this section we investigate how an integrated layered video coding can adapt to multi-time scale fluctuations. We shall clarify that there is no known integrated algorithm. However we believe with our contribution to the delay cognizant aspect, such algorithms will soon appear. The following discussions are based on properties of individual layered video coding techniques, which shall be equally applicable to an integrated solution.

Given the set of layers generated by such integrated coding algorithm, one main objective for a receiver is to select the subset of layers to receive, based on its connection bandwidth and device capability. The subset selection is equivalent to a mapping of layers to time scales. Figure 9 presents a 3D view of the whole set of layers when a two-level decomposition is applied to each dimension. The rate dimension corresponds to a partition of layers by bit rates; the delay dimension to a partition by delay tolerance; and the loss dimension to a partition by error resilience. In this figure, a cube represents a video layer and characterizes its QoS triplet (rate, delay, loss) by its location. The cube nearest to the origin represents the core video information, the most visually significant data requiring the least bandwidth, least error resilient and lowest delay. This core layer may only carry key frames with the most aggressive compression and has no adaptability to any fluctuation. Adaptability is increased by adding layers along one or more dimensions.

The addition of layers, however, needs to follow data dependency criterion which is not depicted in Figure 9. Data dependency is created because of the removal of both visual and
statistical redundancy. Redundancy removal is necessary for compression efficiency. It is common that compressed video data in a higher bit rate layer depends on data in a lower bit rate layer. A high error resilient layer (such as those carrying high frequency transform coefficients) depends on a low error resilient layer (such as those carrying DC and low frequency coefficients). And should one choose, higher delay layers would depend on lower delay layers.

Figure 10 illustrates an example of the selection of layers. When the available bandwidth is low in the long term, only the four layers associated with low bit rate are selected. Medium and fast time scale fluctuations are adapted by the three layers on top and right of the core layer in this figure. When the available bandwidth is high, all eight layers are selected. In most cases, slow time scale fluctuations exhibit the largest dynamic variation and therefore, layer selection begins at the rate dimension. Bandwidth variation is smallest at the loss dimension and thus layer selection for error resilience happens last.

6 New Challenges to Networking Research

In this paper, we presented a new, delay cognizant perspective for layered video coding which effectively adapts to medium time scale bandwidth fluctuations caused by queueing and scheduling at the link layer. Jointly with two other layered coding techniques, rate scalable and error resilient, they provide a scalable, adaptive solution. While this paper mainly addresses the problem from the view of network video applications, the development of adaptive video or source coding in general could have a profound impact on networking operations from network planning, routing to admission control. For example, when a reservation is passed through several networks, admission decisions are no longer as simple as “accept” or “reject” since requested con-
Figure 10: Different connection environments carry different sets of layers. Following the legend in Figure 9, cubes represent layers assigned to the connection at a given time. When the available bandwidth is high, all eight layers are received. When it is low, only layers corresponding to low rate are received.

...connection rate is not a rigid target. Several iterations of message passing among sender, receivers, and participating network entities may be required to negotiate an all-acceptable arrangement. Mechanisms and algorithms of multi-party QoS negotiation problems are needed [7]. Moreover, with delay tolerant layers, networks can achieve better load balancing on the links and routes by carrying those layers at less congested but much longer paths. The additional flexibility creates new challenges to dynamic routing. Similarly at the physical layer, error resilient packets decrease the need of per-link error control. A programmable socket interface to the physical layer is desirable to toggle its error control mechanisms. While these examples do not fall in the capability portfolio of today’s Internet, we hope new networking technologies will emerge to take advantage of the enabled functions.

References


